Notice

Comments concerning this specification can be sent to the editor via e-mail. The e-mail address is villone@sdd.hp.com. If you find parts of this specification to be contradictory, unclear, or incorrect, please notify the editor. Ideas on how to improve the specification are encouraged.

The PCL Implementor's Guide

Version 6.0 (4/01/95)

Table of Contents

Chapter 4: Escape Command Syntax

Chapter 5: Text Processing

Chapter 6: Printer Control

Chapter 7: Page Control

Chapter 1: PCL Goals and Guidelines

Contents of this Chapter

1.1 Introduction

Historically, printers were developed without a computer industry standard for feature access. Since features differed from device to device, applications for one printer had to be modified to work with another, requiring months or years of software development. Users were reluctant to upgrade to new technologies because applications required extensive modification to support a new printer.

To improve this situation, HP developed the PCL printer language to standardize access to printer features. The PCL language provides the highest level of communication between the system and the printer. It is independent of the host system, device drivers, I/O interface, and network communications. Its purpose is to bring all HP printers together under a common control structure. The feature compatibility provided by the PCL language protects the user’s investment in applications and driver software and provides a vehicle for exploiting new features, capabilities, and technologies.

NOTE: "PCL" is a registered trademark of Hewlett-Packard. "PCL" should be used only as an adjective in literature (e.g., PCL language).

1.2 The PCL Committee

The purpose of the PCL Committee is to maintain a PCL language feature architecture that reduces product development effort, provides a system approach for PCL devices, and facilitates the development of robust device drivers. The objectives of the PCL committee are:

1. Review and approve proposals for extensions and changes to the PCL language:

- Make recommendations for the future
- Discuss backward compatibility implications
- Create guidelines
- Provide an expert forum for discussion and feedback on proposals for new features

2. Document and distribute extensions and changes to the PCL language.

3. Inform and influence development teams:

- Document and distribute responses to proposals
- Informally train implementors
- Provide information for future product definition

4. Maintain the PCL language:

- Obsolete unused features
- Update feature partitions

Committee Members

Committee members as of 11/94 are:

Dean Anderson, SIO (Chairperson)
Ben Brezinski, VPR
Jordi Gonzalez, BCD
Melinda Grant, VCD
John Haney, SDD
Angela Hanson, BPR
Laura Mansfield, SPR
Claude Nichols, VPR
Bob Pentecost, NPR
Ted Podelnyk, ALO
Suzanne Richmond, SPO
Linda Rodda, ALO
Jerry Villone, SIO

1.3 The PCL Implementor’s Guide

***The PCL Implementor’s Guide* should only be distributed internally**. It is not intended as a reference for Independent Software Vendors (ISVs) developing device drivers. Individual product teams will provide technical reference manuals and cookbooks to assist in driver development.

The PCL Implementor's Guide contains information for Hewlett-Packard design engineers and licensees of PCL technology. It includes all the control codes and escape sequences supported in Hewlett-Packard or licensed printers. By defining PCL structure and syntax, it describes how printer features may be controlled by user or system application programs.

The information in this document is subject to change without notice. For the latest revision of *The PCL Implementor's Guide,* contact a committee member. Comments concerning this specification can be sent to villone@sdd.hp.com. If you find parts of this specification to be contradictory, unclear, or incorrect, please notify the editor. Ideas on how to improve the specification are encouraged.

1.4 The Proposal Process

As technology or market direction changes, developers of PCL products may propose language modifications.

Pre-Proposal Planning

The first step in the process is to discuss the problem with a committee member.

Those presenting a proposal should make sure the committee understands the problem that the proposal is addressing. Committee deadlocks often occur because solutions are presented before the problem is understood. The committee may have encountered the same problem before and can recommend a preferred solution.

Proposal Presentation

Each proposal should have a PCL committee sponsor to assist with the form and content. The three forms of proposals and their respective procedures are:

1. OBVIOUS omissions, errors, or ambiguities in the *PCL Implementor's Guide*:

   The submitter should give a marked up copy of the erroneous page with the exact wording of the correction to a committee member. Approvals will be reflected in the next release of the *PCL Implementor's Guide*.

2. MINOR changes to the PCL language. The proposal must contain:

- The exact wording of the change.
- Possible effects on other parts of the PCL language.
- The first products implementing the change and their time frames.

Approvals will be reflected in the next release of the *PCL Implementor's Guide*.

3. MAJOR changes to the PCL language. The proposal must contain:

- The exact wording of the change.
- Possible effects on current and future products.
- Possible effects on other parts of the PCL language.
- The first products implementing the change.
- The time frame for the implementation of the change.

A hard copy of the proposal should be given to the PCL Committee chairperson. The submitter may need to attend committee reviews of the proposal. Submissions should be made early in a project, since the approval process takes two or three committee meetings (90 - 120 days) for complex proposals. Upon committee approval, the submitter should provide a hard and soft copy (preferably in Microsoft Word for Windows 6.0) to the Committee. Approvals will be reflected in the next release of *The PCL Implementor's Guide*.

Proposal Format

PCL command definitions should follow Implementor's Guide format and include the following in order:

1. Functional name of the command followed by the escape sequence.
2. Brief one-sentence functional description of command.
3. List of defined value field values.
4. Default value.
5. Range of possible values.
6. Detailed command description with possible interactions, exceptions, etc.

Feature Analysis

A feature analysis should be submitted along with the command definition. The feature analysis should describe all two-way interactions between the proposed command and other PCL, HP-GL/2, PJL, and PML commands. To obtain a templates in Microsoft Word, FrameMaker, and Interleaf, ask a PCL Committee member.

Process Review

To make the status of proposals brought to the PCL Committee more apparent, the following checkpoints will be tracked at the meetings and in the minutes.

State	Deliverables for next State
Investigation	1. Committee identifies changes, suggestions, and modifications needed to transition to the next phase. 2. Written proposal. 3. Schedule. 4. Concept approval required to transition to the next phase.
Feature Developement	1. Committee identifies changes, suggestions, and modifications needed to transition to the next phase. 2. Feature analysis completed.
Final Editing	1. Committee identifies changes, suggestions, and modifications needed to transition to the next phase. 2. If it has not been previously done, a file is delivered to Jerry Villone.
Approved	Implementor's Guide wording is approved.

Shelved Proposals go back to the Investigation phase. State transition can be bidirectional if significant new data comes to light.

1.5 Language Objectives

The ***primary*** objective of the PCL language is to supply customers with the right print solution at a competitive price.

In addition, internal HP objectives include:

- Reduce system support costs (driver development).
- Reduce R&D investment (leverage).
- Provide compatibility in PCL-based products.
- Provide automatic name familiarity for the PCL language in HP products.

1.6 Historical Perspective

The PCL language has evolved through five major levels of functionality driven by a combination of printer technology, user needs, and application software. Higher levels are supersets of lower levels.

Print and Space is the base function set for single-user output.

Electronic Data Processing adds general-purpose multi-user system printing.

Office Word Processing adds high-quality office documentation (DeskJet 5xx line of products).

Page Formatting adds advanced page printing capabilities (LaserJet II series).

Office Publishing includes cababilities such as font scaling and HP-GL/2 graphics (LaserJet III and 4).

The PCL model has been successful because

- HP printers are generally consistent in their implementation of the PCL language.
- HP printers implement the above language groups in cost-effective formatters.
- HP printers ignore unsupported commands.

1.7 Language Design Guidelines

A general guideline to remember is: *PCL is designed to provide "tools" for ISVs, not total solutions for customers.* Define new escape sequences only if they add new tools, not when they duplicate functionality available through other avenues. An example of such restraint is the scalable type functionality that was added to PCL without adding any new escape sequences.

1. **Escape sequences should have only a single function.** Conflict may occur when future devices attempt to support two or more features simultaneously. The Fill Rectangular Area command is an example that violates this rule. This command supports two very different types of area fill. Similarly, Print Mode Selection allows specification of several different features.

2. **Escape sequences should not duplicate functions.** Do not use two or more escape sequences to provide a single feature. An example that violates this rule is the availability of both PCL Unit moves and decipoint moves.

3. **Preserve data.** Given the option to print or discard, data should be preserved for the customer. An example is the Font Select algorithm.

   Another example is *Esc%-1B*, which switches to stand-alone HP-GL/2. On a DeskJet 1200C, the printed graphic may be clipped if it is in portrait rather than landscape orientation; however, data usable for debugging is still delivered to the customer.

4. **Conserve paper.** Implementors should avoid printing blank or useless pages that are not specifically requested by the data stream (e.g., multiple form feeds). If an *EscE* is received before a mark is made on the paper, the device should reset without ejecting the page. On special paper devices using expensive media, like typesetters and film recorders, it may even be appropriate to ignore multiple consecutive form feeds.

   One appropriate application is the clipping of raster data overflowing the bottom page boundary. Attempts to favor guideline 3 over guideline 4 resulted in a "joke page" containing the "leftover" raster graphics and nothing else. For example, on DeskJets prior to 540, data sent to the unprintable area with perforation skip off was printed on the next page, followed by a form feed. This implementation is not recommended because it wastes paper. Implementors should decide which guideline is most important for their project.

5. **Use a single sequence with a binary header for functionality requiring multiple interdependent sequences.** The Configure Image Data command *Esc*v#W[binary data]* follows this guideline. Font selection could benefit from consolidation.

6. **Avoid implicit actions.** When raster graphics capability was first added to PCL, an implicit start/end of raster graphics "mode" was defined. This has caused a great deal of difficulty as the definition of raster graphics has expanded.

7. **Obsolescence of functionality is a lengthy and expensive procedure.** First, the obsolete functionality must be moved to the obsolete section of the Implementor's Guide. Next, products that contain the obsolete functionality must stop documenting that functionality. Finally, after two or three product generations, when most software supporting the product line has stopped utilizing the obsolete functionality, it can be removed from future devices.

8. **Avoid undocumented underware escape sequences.** These sequences, which may expedite manufacturing processes, perform diagnostics, or provide functionality in a controlled environment, are very difficult to track and support. See Chapter 6 for the recommended underware escape sequence format.

9. **Handle error conditions consistently.** Extreme error conditions can be handled differently across a range of devices without jeopardizing the language standard, but consistency is recommended.

10. **Remember that previous implementations are not always perfect**. Language definition may need to change or be refined at the expense of backward compatibility. Two examples are the "joke page" and the new End Raster command (*Esc*rC*). Another example is the admission for LaserJet III that HMI should not have changed when print direction changed. There will always be creative tension because a software application base is built around the first products released.

Guidelines for Specific Escape Sequences

1. PCL parameter ranges were originally defined for a 16-bit environment. Future parsers may need to accept 32 bit values as they implement features whose value fields extend beyond 32,767. "Movement" is one place where limits may be encountered. Decipoint moves, which are limited to 45.5", may not be adequate for large format devices greater than E-size. Other examples are large graphics transfer, font downloads, and user-defined patterns.

2. When the contents of a value field identified by a state variable are out of range, the previous value should be retained rather than resetting to the default (e.g., Render Alorithm). When the contents of a value field identifying a physical range are exceeded, the value field should be clamped to the associated limit (e.g., Move CAP commands).

3. Use 0, if possible, as a default value. Default values should be determined partially by the impact of the escape sequence on devices that will not implement the functionality.

4. Signs encountered in an escape sequence should be considered and interpreted, not ignored. Examples are horizontal and vertical CAP positioning (e.g., *Esc&a#H* and *Esc&a#V*), and font stroke weight (*Esc(s#B* and *Esc)s#B*).

5. A "W" or "w" terminator on an escape sequence should be reserved for binary data transfers.

6. Group characters and terminating characters may be chosen for their mnemonic value. However, since this is often impossible, nothing can be inferred from the meaning of any specific selection.

7. The "Q" symbol sets should be used only for special character sets and not documented for broad use.

8. Metric compatibility for at least some portion of font products is an *excellent* idea from a customer standpoint. It should be an objective for all devices.

1.8 References

The HP-GL/2 Implementor's Guide (villone@sdd.hp.com)

The Hewlett-Packard Book of Characters (HP BPR; HP Publication No. 5091-5154E)

The PCL 5 Printer Language Technical Reference Manual (Copyright 1992, PN 5961-0509)

PCL 5 Comparison Guide (Copyright 1994, PN 5961-0702)

Printer Job Language Technical Reference Manual (Copyright 1994, PN 5961-0704)

Printer Job Language ERS (HP BPR)

PCL 5 Color Technical Reference Manual (Copyright 1994, PN 5961-0635)

PML Protocol Specification (pml-editor@hpbs987.boi.hp.com)

Chapter 2: Job Setup

Contents of this Chapter

This chapter contains information on job setup performed outside of a PCL job (for information on job setup within PCL, see Chapter 6).

2.1 Introduction

At the beginning of every print job, some basic setup commands must be issued before sending print data to HP printers. This is especially true if the printer is connected to the network. The printer language may need to be changed (e.g., PCL to PostScript) or the user defaults modified from the last print job. Manual control panel setup is often inconvenient and may even be impossible as control panels become more simplified. Some job control commands, such as a user-default reset (*EscE*) may be sent within the print data itself, but this method significantly limits system management on a network. To satisfy these system control needs, HP printers are now supporting job control languages like PJL (Printer Job Language) and PML (Peripheral Management Language), which reside logically above the PCL and PostScript printer languages.

This chapter is a general description of job control, both for printers that support PJL or PML and those that do not.

2.2 Defaults

Defaults are feature settings that are programmed at the factory or selected at the control panel. Feature settings may be changed from their defaults when a job is printed; but then these job-specific settings may need to be changed by the next job. Each new job must be able to set the device to a known state. For example, if the number of copies is set to 65 copies by the previous application, the new application would also print 65 copies. Starting with the user default environment at the beginning of each job eliminates the need to set each feature individually.

The printer's current feature settings are collectively called the ***Print Environment.*** Some devices may maintain all five of the print environments described below, with only one active at a time. Not all PCL devices have all five environments, and the feature set in each environment may vary between devices.

Factory Default Environment

Factory defaults are set at the factory. Factory defaults become active when the printer is first powered on before control panel settings are changed or any printer commands are sent from an application.

Generally, a simple power recycle or reset does not restore the factory environment. (although this is device-dependent); a special sequence of button presses or commands (e.g., the PJL INITIALIZE) must be used.

User Default Environment

The User Default environment contains the control panel settings. The User Default Environment may be changed at the control panel or by PJL or PML commands. Printer language commands override user defaults. A control panel, printer command reset, or power recycle (this is device-dependent) restores the user default environment.

PJL Current Environment (PJL printers only)

The PJL Current Environment is the same as the User Default Environment at the beginning of a PJL job. The PJL SET command modifies the Current Environment for the duration of a PJL job (between JOB and EOJ, or until the next UEL if JOB and EOJ commands are not used). Within a PJL job, Current Environment settings override User Default settings.

Modified Print Environment

At the beginning of a PCL job, a PCL reset (*EscE*) loads the current environment (PJL printers) or user default environment (non-PJL printers) into the Modified Print Environment. Modified print enviroment settings are changed using printer language commands (such as PCL escape sequences).

The Modified Print Environment is saved during a macro call or overlay and restored upon completion.

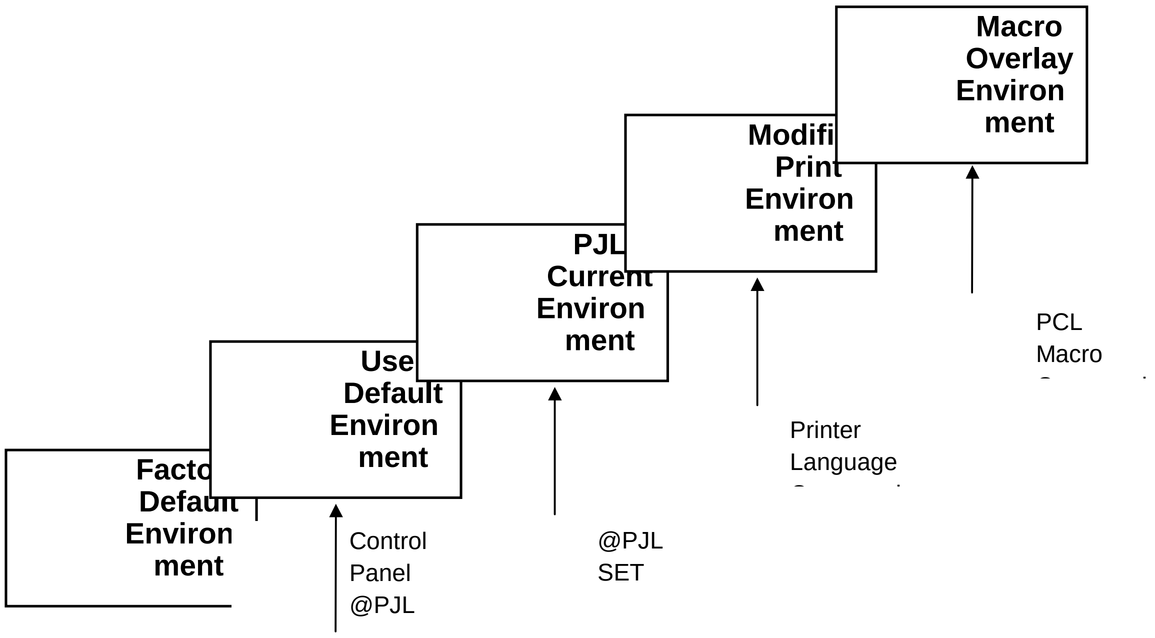
Overlay Print Environment

The Overlay Print Environment becomes active whenever a macro overlay is enabled; the modified print environment is saved. Upon completion of the macro, the modified print environment is restored.

Print Environment Hierarchy

The print environment is a hierarchical structure. Factory defaults are overridden by control panel settings, PJL DEFAULT commands, or PML commands — creating the User Default Environment. On PJL printers, PJL SET commands override user defaults — creating the PJL Current Environment. Printer commands from an application override the user default and current environments — creating the Modified Print Environment. A macro overlay creates the Overlay Environment, which overrides the modifed print environment.

Any feature not set by the active environment defaults to the value set in the next lower environment in the hierarchy.



2.3 Setup without a Job Control Language

The job setup procedure for printers that do not support a job control language such as PJL ~~or PML~~ consists of the following four steps:

1. Reset the printer at the beginning of the jobs using *EscE*.
2. Send the printer commands that set the printer to a desired state and override any environment settings that could have been set using the control panel. The following are examples of environmental variables that may be set with PCL commands:

Number of Copies	*Esc&l#X*
Duplex printing	*Esc&l#S*
Page Size	*Esc&l#A*
Tray/Manual Feed (paper source)	*Esc&l#H*
Orientation (portrait, landscape)	*Esc&l#O*
Number of Lines per Page	*Esc&l#C*
Symbol Set	*Esc(ID*
Selected Font	*Esc(s#p#h#v#s#b#T*

3. Send the print data.
4. Reset (*EscE*) the printer at the end of the job.

2.4 Printer Job Language (PJL)

PJL resides logically above the PCL and PostScript language parsers. PJL provides language switching plus a broad range of features including status readback and the manipulation of control panel settings.

All PJL devices support at least the following three commands:

Universal Exit Language (*UEL*)
Enter Language
Comment

These commands are inserted in the data stream between print jobs in the sequence shown below:

Universal Exit Language/Start of PJL

Comment

Enter Language

(Begin PCL or Postscript Print Job)

...

(End of Print Job)

Universal Exit Language/Start of PJL

PJL contains many other commands for status readback and modifying printer enviroment settings. Some of these are:

CO	INQUIRE
DE	INITIALIZE
DI	JOB
EC	RESET

EN	SET	
US	UST	
		A
		T
		U
		S
		C
		F
		F
INF	OP	
		M
		S
		G
RD	ST	
		M
		S
		G
EO	UEL	

For a complete description, see *The Printer Job Language External Reference Specification*.

All PJL commands except UEL must begin with the uppercase "@PJL" prefix. The portion of the PJL command following the prefix may be either upper or lower case. All PJL commands except UEL must end with a LF, which may be preceded by an optional CR. Spaces in PJL commands depend on their location; some spacing is required and some is optional. Spacing requirements are shown below:

```
    @PJL<WS>Command<WS>Option[<WS>]=[<WS>]Value[<CR>]<LF><WS>
```

where <ws> indicates white space resulting from one or more of either <SP> or <HT>, and [ ] indicates optional fields.

Universal Exit Language (UEL) *Esc%-12345X*

The UEL command, which is used at the beginning and end of every PJL job, transfers control to PJL. At the beginning of a job, the X at the end of the command must be immediately followed by another PJL command; there can be no spaces or characters between the X and the @ that begins the next command. On the other hand, the UEL at the end of the job is the final part of the job and does not require anything to follow it.

This command performs the following actions:

- Prints all data received before this command.
- Performs a reset: *EscE* in PCL, *IN* in HP-GL/2, or *Cntrl-D* (end of job) in PostScript.
- Turns control over to PJL.

In the PCL and HP-GL/2 contexts, this command is always recognized except in HP-GL/2 label mode with TD = 1, during a PCL binary transfer, or in Text Parsing Method 2 (*Esc&t#P*).

This command is also a valid HP-GL/2 terminator.

NOTE: UEL must be immediately followed by "`@PJL`". Characters or control codes (e.g., `CR` or `LF`) other than `@PJL` enable the default language and process the job in that language.

NOTE: All jobs should start and end with UEL. In addition to entering PJL, UEL has the same effect as the *EscE*; but *EscE* should always be included for backward compatibility.

DEVICE NOTE: DJ3xx's, DJ5xx's, and LJ's prior to LJIIISi do not support PJL. They ignore and print all PJL commands.

Enter Language @PJL ENTER LANGUAGE=PERSONALITY

```
@PJL ENTER LANGUAGE = {PCL/POSTSCRIPT}[<CR>]<LF>
```

If the printer does not receive this command, it will try to determine the appropriate language to use. If automatic language switching is supported and enabled, the printer chooses from the available languages; otherwise, the default language is used. This increases the probability that applications not supporting PJL switching will correctly print.

Comment @PJL COMMENT words

This command designates the current line as a comment, which is ignored.

```
  @PJL COMMENT <Words>[<CR>]<LF>
```

2.5 Job Setup for PJL Printers

The job setup procedure for printers that support PJL consists of the following fours steps:

1. Send a UEL command (*Esc%-12345X*) at the beginnning of the job to give control to PJL.
2. Use the PJL ENTER LANGUAGE command to enter the desired printer language.
3. Reset the printer using *EscE* reset before sending PCL commands.
4. Send printer commands to set the printer to a desired state and to override any environmental settings that could have been set using the control panel.
5. Send the print data.
6. Reset the printer at the end of the job using *EscE* followed by the UEL command.

For printers that support status readback, the PJL INQUIRE command may be used after UEL at the beginning of the job to request the status of environment variables. PJL status readback commands can provide the name of the printer, its configuration, and a list of environment settings. Environment feature settings may be set using PJL SET or PCL commands.

2.6 Peripheral Management Language (PML)

PJL is best suited for job level status and control that is synchronous with the PCL or PostScript data stream. PML provides asynchronous status readback and remote printer control commands by using multiple physical channels or multiple logical channels; commands and responses can be received immediately on a channel different than the data without waiting until all the data is sent. PML permits prompt printer control; for example, a network administrator could use a remote device reset to troubleshoot a clogged spooler.

The implementation of MLC (Multiple Logical Channels), and MIO 6.0 allows asynchronous printer monitoring and control on a separate channel. PML utilizes multiple-channel capability to provide asynchronous status and control of the entire printer, allowing device configuration and error messages to occur asynchronously to the data stream. By utilizing this separate control channel capability, PML can provide asynchronous control that supplements PJL.

To take advantage of this asynchronous capability, PML requires an I/O with separate channels for data and control/monitoring. Extended capabilities ports (ECP) like IEEE-P1284 and MIO 6.0 provide this functionality. Parallel half-duplex bi-directional ports allow implementation of a multiple logical channels (MLC) packet protocol that provides simultaneous control and data transmission.

Devices without an ECP port or MIO 6.0 port may still use "encapsulated" PML. The Peripheral Configuration command (*Esc&b#W*) provides a way for PCL to send PML commands. This method may be used even for devices with an ECP port if synchronization between the PCL data stream and the asynchronous control of PML is desired.

PML provides unsolicited status, called *traps*, to hosts that request it. Trapped objects may relate to information on:

Device status — changes: paper jam, out-of-paper, out-of-ink, etc.
Job status — job starts and ends.
Page status — the job to which the page belongs, and the page number within the job.

Chapter 3: The PCL Page

Contents of this Chapter

3.1 Introduction

The Physical Page

The ***physical page*** is the actual sheet of media. The term refers to the size of the media installed in the printer. The ***printable area*** is the maximum area on the physical page in which the printer is able to place a dot. This is an absolute, device-dependent limitation, which is determined by the technology of the printing device.

The Logical Page

The ***logical page*** is the current addressable area on the physical page; it defines the area in which the PCL cursor can be positioned. The PCL cursor, called ***CAP*** (Current Active Position), is the position on the logical page where the next character will be positioned.

As shown on the next page, the logical page extends the length of the physical page — it may, in fact, extend into the unprintable area of a device. On most HP printers, there is a gap between the logical page and the unprintable region.

CAP may be moved anywhere on the logical page using Move CAP commands (e.g., *Esc&a#H*, *Esc&a#V*); but CAP cannot be moved outside the logical page bounds.

The Logical Page

The PCL Coordinate System

The PCL coordinate system is shown below. Relative to the PCL coordinate system the Y direction is always downward and the X direction is to the right. The point (0,0) is the intersection of the left edge of the logical page and the current top margin. Usually the top margin defaults to 1/2", but it may be changed by the Top Margin command (*Esc&l#E*). The point (0,0) moves with the top margin and will rotate around the page if the orientation is changed.

The PCL Coordinate System

The HP-GL/2 Coordinate System

The HP-GL/2 coordinate system is shown below. Both PCL and HP-GL/2 use a Cartesian coordinate system with perpendicular X and Y axes. However, they differ in unit dimension and orientation. Positive Y values are plotted below the origin in PCL and above the origin in HP-GL/2.

The HP-GL/2 Coordinate System

* The default "text area" of an HP-GL/2 graphic that is imported into PCL is coincident with the right and left edges of the logical page, one-half inch down from the top of the logical page, and one-half inch up from the bottom of the logical page.

3.2 Margins

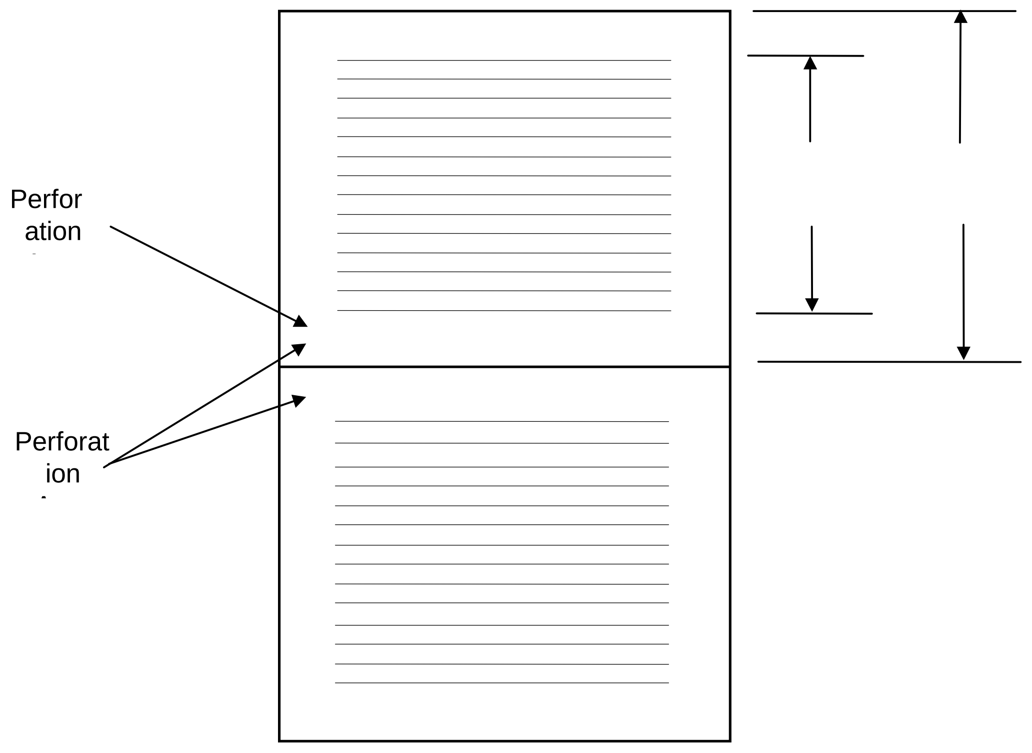
The logical page represents the current addressable area on the physical page; but the actual text area within the logical page is restricted by text margins, and the raster graphics area is restricted by graphics margins.

Text Margins

The text margins define the ***text area***, the area on the physical page where text may be printed. The defaults for the left and right margins are the left and right logical page boundaries. The default for the top margin is 1/2" below the top of the logical page.

The left, right, and top margins may be explicitly set by PCL commands (*Esc&a#L*, *Esc&a#M*, *Esc&l#E*). The bottom margin is set indirectly by the Text Length command (*Esc&l#F*) when the perforation skip region is enabled (default). The perforation skip region extends from the bottom of the text area on one page to the top of the text area on the next page. Disabling perforation skip (*Esc&l0L*) causes text to be printed to the end of the page.

DEVICE NOTE: LJs and SPR products include only the bottom margin in the perforation area. With perforation skip enabled, a linefeed within the top margin moves CAP down one line. DJs below 1200 include both the bottom margin of the current page and the top margin of the next page in the perforation skip area. With perforation skip enabled, a linefeed within the top margin moves CAP to the top margin.



Printing Outside Text Margins

Characters are usually printed only in the text area between the margins. However, characters can be printed outside the margins by using Move CAP commands (e.g., *Esc&a#H*, *Esc&a#V*). Once outside the left and right margins, all features function normally to the edge of the logical page. That is, if CAP is to the right of the right margin, characters are printed and CAP is updated accordingly

Disabling perforation skip mode (*Esc&l0L*) allows characters to be printed in the perforation region between the bottom margin of one page and top margin of the next.

Even with perforation skip enabled, printing may occur outside the top and bottom margins by using Move CAP commands (e.g., *Esc&a#H*, *Esc&a#V*).

DEVICE NOTE: LJs and SPR products include only the bottom margin in the perforation area. With perforation skip enabled, a linefeed within the top margin moves CAP down one line. DJs below 1200 include both the bottom margin of the current page and the top margin of the next page in the perforation skip area. With perforation skip enabled, a linefeed within the top margin moves CAP to the top margin. Therefore, with perforation skip enabled, DJs below 1200 must use Move CAP commands rather than linefeeds if more than one line is to be printed within the top margin

Generally, characters are never printed outside the logical page. However, some characters may extend outside the left and right boundaries of the logical page; for example, the italic ***f*** starts to the left of CAP. In most HP printers, there is a gap between the left and right boundaries of the logical page and the unprintable area.

Characters are not clipped unless they extend into the unprintable area. Some printers allow *pixel clipping*, where parts of characters are clipped. Without pixel clipping, printers clip the entire character if it extends into the unprintable area.

Raster Graphics Margins

Raster mode is different than text mode. Raster mode may be started explicitly by a Start Raster command (*Esc*r#A*), or implicitly by sending data with a Transfer Raster command (*Esc*b#V* or *Esc*b#W*).

The PCL raster graphics system uses the concept of a raster graphics picture, or *raster area*. The raster area represents a boundary within which the printer will zero-fill missing and short. The raster area can be started at CAP anywhere on the logical page.

Raster margins may be different than text margins. Raster height and width can be set explicitly; otherwise they are clipped at the logical page or unprintable area.

Raster graphics never causes a page eject. At the end of the physical page, CAP is set to the logical page boundary and the rest of the graphic is clipped and discarded.

Vector Graphics Margins

In addition to text and raster graphics, PCL5 printers can merge HP-GL/2 vector graphics with text on the logical page. The simplest procedure is to enter the HP-GL/2 context by *Esc%#B*, create the drawing using HP-GL/2 commands, and re-enter the PCL context by *Esc%#A*.

The HP-GL/2 graphic is placed on the logical page within a rectangle called the ***picture frame***, which effectively defines the margins for vector graphics. The default picture frame position or *anchor point* is 0,0 and the default size is the default PCL text area (the logical page minus the top and bottom margins); but the picture frame size and position may be modified by PCL commands called *picture frame directives* prior to entering the HP-GL/2 context.

The Picture Frame

When an HP-GL/2 graphic is placed on the page, it is limited by both HP-GL/2 and PCL restrictions. The HP-GL/2 hard clip and soft clip boundaries limit the original plot; the picture frame limits the transferred plot; and the logical page may clip the picture frame.

Several strategies may be used to handle these restrictions. The size and position of the picture frame may be adjusted to exactly match HP-GL/2 and PCL coordinates, with no picture frame scaling. Of course, if only user-units were used to create the plot, it will be automatically scaled to fit any size picture frame. The plot can also be horizontally and vertically scaled with respect to the picture frame by any desired scale factor — even if it was originally created with absolute units — by using PCL picture frame directives.

3.3 Orientation

Orientation defines the position of logical page with respect to the physical page. Orientation refers to the viewing aspect, and has no relation to the print scan or paper loading directions.

Portrait orientation means the logical page origin (0,0) is toward the top left corner of the physical page, with the X-direction to the right and the Y-direction downward. *Landscape* orientation means the origin is rotated counterclockwise by 90 degrees toward the lower left corner of the physical page, then the X direction is upward and the Y direction is to the right.

The Orientation command (*Esc&l#O*) can be used to change logical page orientation only once per physical page because it ejects the page. The Print Direction command (*Esc&a#P*), which rotates the logical page coordinate system **with respect to the current orientation**, does not eject the page and may be used change orientation multiple times on the same page.

Raster graphics do not track logical page orientation (although Raster Presentation (*Esc*r#F*) can be used change raster orientation). HP-GL/2 graphics do track changes caused by the Orientation command (*Esc&lO*), but not those caused by the Print Direction command (*Esc&aP*) unless the Enter HP-GL/2 command has been sent with a value field of 2 or 3 (*Esc%2B*, *Esc%3B*). It is also possible to alter the HP-GL/2 picture orientation within the logical page by using the HP-GL/2 RO command.

Fonts track logical page orientation.

3.4 The Current Active Position (CAP)

The PCL cursor, which is called the ***Current Active Position (CAP)***, is analogous to the blinking character that identifies the current position on the computer screen. In PCL, it is the position where the next character or graphics dot will be printed. After printing a character, CAP moves to the right a distance equal to the width (escapement) of that character. After printing a raster row, CAP moves one dot row down.

Move CAP Commands

CAP may be moved anywhere within the logical page using Move CAP Horizontal (*Esc&a#C, Esc&a#H, Esc*p#X*) and Vertical (*Esc&a#R, Esc&a#V, Esc*p#Y*) commands and control codes (CR, SP, BS, HT, LF, FF, etc). Either absolute or relative movement can be specified. ***Absolute motion*** is referenced to the intersection of the top margin and the left boundary of the logical page (0,0). A Move CAP command with an unsigned value field is an absolute move. ***Relative motion*** is referenced to the current position of CAP. A Move CAP command with a signed value field is a relative move.

Relative Motion

Default CAP

CAP is ***floating*** until printable data or a command affecting CAP is received. The floating CAP is always at the (0,0) position on left margin and 3/4 of a line space below the top margin. A floating CAP tracks the (0,0) position with changes to orientation, top margin, left margin, and line spacing. Reception of any printable character or command affecting CAP will ***fix*** CAP. A fixed CAP is not affected by changes to the orientation, top margin, left margin, or line spacing.

The HP-GL/2 Cursor

The HP-GL/2 cursor is called the ***Pen Location***. It may be moved anywhere within the current HP-GL/2 addressable area

3.5 Units of Movement

A printer's resolution is not related to the size of the units used to position CAP. The units of movement on the X-axis of the PCL coordinate system may be PCL Units, decipoints, or columns. The units on the Y-axis may be PCL Units, decipoints, or rows.

PCL Unit

The size of a PCL Unit is user-definable by the Unit of Measure command (*Esc&u#D*). Devices using this command round movements using the Move CAP (PCL Unit) commands to the nearest PCL Unit, rather than to the nearest internal unit. The default PCL Unit is 1/300" unless changed by the control panel.

PCL Units were formerly referred to as "dots", but were renamed to prevent confusion with the printer's physically printed dots, which are determined by the printer's resolution (see the discussion on resolution on the next page.)

Decipoint

A decipoint is 1/720" or 1/10 of a PCL point. (A PCL point is *exactly* 1/72" as opposed to a typographic point which is *approximately* 1/72".)

Column

Column width depends on the current CMI (character motion index).

Rows

Row height depends on the current LMI (line motion index) or lines-per-inch (lpi) setting.

Printer Internal Units

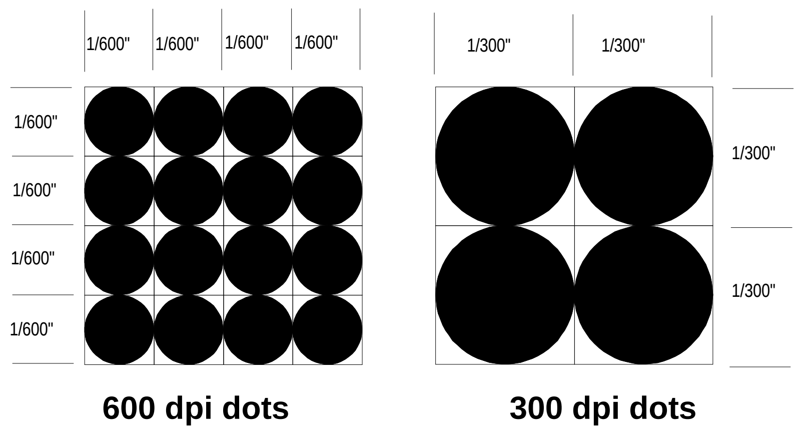
Internally, the printer maps PCL units, decipoints, columns, and rows to its own internal units. LaserJets generally use 1/7200" as an internal unit; DeskJets below 1200 use 1/3600". CAP positioning is always rounded to the nearest internal unit; however, the start position of printed text glyphs is rounded to the nearest PCL Unit if the Unit of Measure command (Esc&u#D) is supported.

HP-GL/2 Units

In HP-GL/2, the device moves in either *user units* or absolute *plotter units*. One plotter unit equals .025 mm (approximately .00098 in). User units can be set to any size by HP-GL/2 scaling commands. Both types of HP-GL/2 units are converted to the device resolution prior to printing.

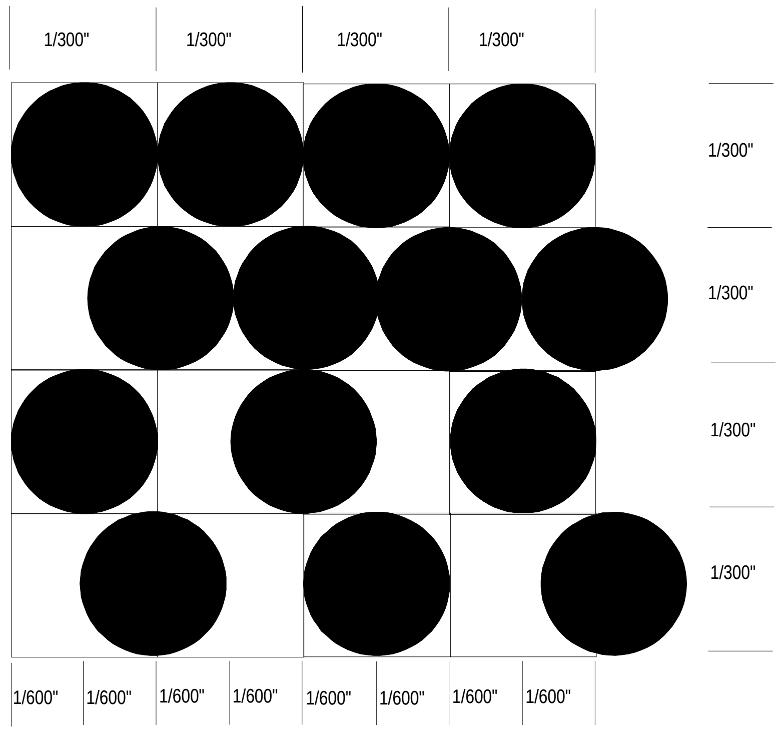
3.6 Resolution

A dot is the smallest mark a printer can make. A printer’s resolution in dots per inch (dpi) refers to the number of dots that can be placed horizontally and/or vertically in a square inch.



Addressability

Addressability refers to the ability of a device to position a dot. As shown below, a device with 300 dpi resolution may have 600 dpi addressability. Usually addressability is defined in the horizontal direction.



Raster Graphics Resolution

Some devices allow the selection of different raster resolutions, up to the maximum machine resolution. At lower raster resolutions, each pixel, which is the smallest picture element that can be defined, consists of more than one dot. However, dot-spacing remains the same, and the same number of dots-per-inch are printed.

When 150 dpi is requested in a 300 dpi machine, 300 dpi is still printed; but the pixel is now composed of four dots, and is twice as wide and twice as tall.

bit	300 dpi	150 dpi
1	•	• •
		• •

Requesting a lower resolution does not print less detail. The printer still prints at its highest resolution; it just prints the image larger. Using the same data, a 150 dpi image is printed twice as large as a 300 dpi image (with four times the number of dots). The larger image may be clipped if it exceeds the raster area. To keep the destination image the same size, it is necessary to send the printer one-fourth the information.

Chapter 4: Escape Command Syntax

Contents of this Chapter

This chapter contains information required to implement PCL command-level parsing. Text level processing, during which the device is processing printable characters, is described in Chapter 5.

4.1 Introduction to PCL Commands

There are three types of PCL commands:

- Control Codes
- Two-Character Escape Sequences
- Parameterized Escape Sequences

Control codes are discussed in Chapter 5. This chapter defines the syntax for two-character and parameterized escape sequences.

Escape Sequences begin with the ASCII escape character *Esc* (decimal 27), followed by at least one other character. A PCL device receiving *Esc* tries to interpret subsequent characters as a PCL command rather than as data to be printed. If the device does not recognize the command, it remains at the text processing level: syntactically correct but unrecognized escape sequences are ignored; syntactically incorrect sequences are ignored and printed.

4.2 Two-Character Sequences

Two-character escape sequences take the form *EscX*, where "X" is an ASCII character between 48-126 decimal (0 through ~) that defines the operation being performed. *EscE* (Reset Printer) and *Esc9* (Clear Margins) are examples of two-character escape sequences.

4.3 Parameterized Sequences

An escape sequence is parameterized if the character following *Esc* is between 33-47 decimal (! through /). Parameterized escape sequences have the form:

Esc X y # Z_n[Data]

or, when combining escape sequences with the same parameterized and group characters:

Esc X y # z_1 # z_2 # z_3 ... # Z_n[Data]

In parameterized escape sequences, **y**, **#**, $\mathbf{z_i}$ and **[Data]** are optional.

X	***Parameterized Character*** **-** An ASCII character between 33-47 decimal (! through /) identifying the sequence as parameterized.
y	***Group Character*** **-** An ASCII character between 96-126 decimal (‘ through ~) identifying the type of control being performed.
#	***Value Field*** **-** A group of ASCII characters specifying an unsigned numeric value between 0 and 2^{32}-1, or a signed numeric value between -2^{31} and 2^{31}-1. If no value field is present, zero is assumed.
$\mathbf{z_i}$	***Parameter Character*** **-** An ASCII character between 96-126 decimal (‘ through ~) identifying the function that interprets the previous value field. This lower-case character is used instead of Z_n (which is z_i + 32) when combining escape sequences.
$\mathbf{Z_n}$	***Terminating Character*** **-** An ASCII character between 64-94 decimal (@ through ^) identifying the function that interprets the previous value field. Z_n terminates an escape sequence or a string of combined escape sequences.
[Data]	***Binary Data*** **-** Eight-bit data (e.g., graphics data, downloaded fonts, etc.) that immediately follows the terminating character. The number of data bytes is specified by the value field.

The following are examples of valid PCL commands:

Command	**Description**
EscE	Printer Reset (two-character escape sequence).
Esc(U	Select USASCII as primary character set (no group character or value field).
Esc(8U	Select Roman-8 as primary character set (no group character).
Esc(sB	Select medium stroke weight as primary (no value field).
Esc(s3B	Select bold stroke weight as primary.
*Esc*b5W12345*	Send 5 bytes of binary raster data.

Parameterized Character (X)

This ASCII character between 33-47 decimal (! through /) identifies the escape sequence as parameterized. It should be chosen in accordance with ANSI X3.41. Although some exceptions exist, the following parameterized characters are recommended for feature categories.

%	Personality mode change.
&	Device feature control.
(	Designates a font as primary.
)	Designates a font as secondary.
*	Graphics control.

Group Character (y)

This optional ASCII character between 96-126 decimal (‘ through ~) identifies the family of functions being performed. Within HP a large number of usages have been documented, and these guidelines should be followed if possible.

Value Field (#)

This is a group of characters specifying a numeric value. A value is represented as an ASCII string of decimal digits (0 through 9) specifying a numeric value between -2^{31} and $2^{31} - 1$. Zero is assigned if a value field is expected but not present. Leading zeros and blanks are ignored.

A plus "+" or minus "-" sign may start a value field; and spaces between the sign and the numeric value are stripped. Following the sign, or in its absence, a value field can be started by a decimal point "." or the decimal digits "0" through "9". Once started, a value field is closed by any character in the ranges 32-45 (SPACE through -), 47 (/), 58-63 (: through ?), or the second occurrence of a decimal point (46). Examples of value commands with signed value fields are Simple Color (*Esc*r#U*), Mechanical Print Quality (*Esc*o#Q*), and the Move CAP commands.

Value fields are assumed to be integer unless otherwise specified. The first decimal point indicates the start of a fractional portion. The number of decimal places after the decimal point is device dependent. Fractional portions of the value field are ignored if not used.

If multiple value fields are received when only one is required, the first value field should be used. Presently, no commands use more than one value field.

Commands not using the plus "+" or minus "-" sign as part of their definition should parse and use the sign to compute magnitude. Binary transfers can ignore the sign.

DEVICE NOTE: LaserJets prior to LJIII use two decimal places for decipoint commands and four decimal places for CMI and LMI commands. These were the limits within which these products could maintain accuracy when converting to their internal units of measure.

DEVICE NOTE: DeskJets below DJ1200C use two decimal places for all fractional values.

The following are examples of value fields and the resulting data passed to the command. The brackets are used to distinguish leading or trailing spaces in the examples.

Value Field	**Resulting Data**
{9}	9 — no spaces, signs, or fractional value.
{ 009 }	9 — leading 0's and spaces and trailing spaces are stripped.
{+ 007}	Positive signed 7 — spaces after the sign are stripped.
{-7}	Negative signed 7 — no spaces or zeros after the negative sign.
{} or { }	0 — since the value field is blank or no value field is present.
{42187}	32767 — magnitude is limited to 32767 (with noted exceptions)
{4./25}	4 — The "/" terminates the value field.
{4.75}	4.75
{4.75}	4 if a decimal portion is not expected.

Parameter Character (z_i)

This ASCII character between 96-126 decimal (' through ~) is used to combine escape sequences having the same parameterized and group characters. After executing the function identified by z_i, the parser interprets succeeding data as a value field. Z_n, the upshifted (z_i - 32) equivalent, signifies escape sequence termination when used instead of z_i.

Terminating Character (Z_n)

This ASCII character between 64-94 decimal (@ through ^) signifies escape sequence termination when it is used instead of z_i, the downshifted (Z_n + 32) equivalent.

The terminating character Z_n appears in two forms:

1. If at least one parameter is specified, the last parameter character in an escape sequence or string of combined escape sequences is upshifted from its column 6 or 7 position in the ASCII table to the equivalent position in column 4 or 5. That is, z_i becomes the terminating character z_i - 32. For example, *Esc&l6d...* and *Esc&l6D* both set line spacing to 6 lines per inch.

2. If no parameters (and therefore no values) are specified, the group character is the terminator. That is, y becomes Z_n. For example, *Esc(@* is a legal escape sequence with no group character.

The parameter and terminating characters have the same function; but the lower-case parameter character (Z_n + 32) allows escape sequences from the same group to be combined. When the parser encounters z_i, it expects a value field. On the other hand, Z_n places a device back at the text level of processing.

Binary Data Transfers [Data]

Some parameterized escape sequences transfer binary data (graphics, transparent print data, font downloading, etc.). These sequences contain a data field immediately following the terminating or parameter character. The data field contains 8-bit binary data that is not interpreted by escape sequence parsing. The value field of the escape sequence must specify the correct number of data bytes. Extra bytes are ignored by the command and may be considered printable data. Insufficient bytes may cause the device to consider characters following the command as part of the data field.

To ensure that products not supporting one or more of these commands can interpret this data in a non-disruptive fashion, the following guidelines should be used for binary transfer escape sequences:

- The format is *EscXy#w/W [binary data]*. The binary data string contains "#" bytes.
- The value field must describe the number of bytes to be transferred.
- Binary transfer commands should use lowercase "w" as the parameter character or uppercase "W" as the terminating character.
- The "w" parameter character or "W" terminating character should be followed immediately by the binary data.
- Unsupported binary escape sequences denoted by the "w" or "W" terminating character should be handled as an unrecognized command and the binary data ignored.
- When illegal or unsupported escape sequences are discarded, the binary data associated with the command must also be discarded.

DEVICE NOTE: LaserJets do not currently support the lowercase "w" as indicating that binary data follows and will continue parsing the binary data, looking for value field or terminating character.

NOTE: Not all binary transfer escape sequences are defined by the "w" parameter character or "W" terminating character. However, this is the preferred parameter for standardization. The following are some sequences that do not end with a "w" or "W"; these commands should be parsed and any binary data discarded if the device does not support the command.

Esc & p # x/X [Transparent data]	Transparent print data
*Esc * b # v/V [Raster data]*	Raster graphics transfer by plane

NOTE: The following commands should not be interpreted as binary data commands:

Esc & k # W	Device Specific Control
Esc (W	Select Primary Font Character Set
Esc) W	Select Secondary Font Character Set
Esc & d W	Underline

4.4 Rules for PCL Commands

PCL Commands as Modes

Most PCL commands are defined as modes: once a feature has been selected, it remains in effect until a device reset. However, other commands may interact and change the feature.

Order of Execution

Functions should be executed as they are decoded. For example, changing line spacing from 6 to 8 lpi and then setting text length to 80 lines has a completely different meaning than setting text length to 80 lines and then changing line spacing from 6 to 8 lpi. Another example might involve page length and top margin: the top margin will move if set after page length.

Unrecognized Sequences

A syntactically correct escape sequence that is not recognized is ignored.

An escape sequence that cannot be fully implemented should not necessarily be ignored, since **some** action may be more appropriate than no action. For example, attempting to set the right margin past the physical limits of the printer should cause the right margin to be set at the printer's physical limit.

The Simplex/Duplex command (*Esc&l#S*) is an example of a command that is not completely ignored by some non-duplex printers; it causes a conditional page eject in LaserJets and DeskJets above 1000.

Illegal Syntax

If an illegal character is received within an escape sequence, processing of the sequence terminates. For data preservation, the illegal character should be interpreted at the text level independently of previous characters.

4.5 Combining Commands

Parameterized (**but not two-character**) escape sequences can be combined, eliminating intermediate escape characters. To combine escape sequences, use the parameter character zi, instead of its upper-case equivalent Z_n (z_i - 32), for every escape sequence except the last.

Use the following three rules to combine parameterized sequences into one string:

1. The first two characters after *Esc* (the parameterized and group character) must be the same for all the commands being combined. In the example below, these are "*&*" and "*l*".

2. All alphabetic characters within the combined command must be lower-case except the final letter, which is upper-case to notify the device when the command is complete. In the example below, the terminating character "*E*" in the first sequence is changed to "*e*" in the combined sequence.

3. Combined commands are executed as they are parsed (from left to right). That is, if several commands are concatenated and the last command has a syntax error, all the commands up to the last one are executed.

EXAMPLE: To set the top margin to line 10 and the text length to 70 lines, either of the following could be sent:

Esc&l10E and *Esc&l70F*
or
Esc&l10e70F

4.6 Backus-Naur Form (BNF)

PCL escape sequence syntax can be described by a rule system derived from Backus-Naur Form (BNF). The following symbols are used:

bold	indicates syntactic categories
::=	"to be rewritten as"
|	separates choices
| ... |	specifies the range of choices
XXH	character code in hex
$\{\}_1$	choose 1 of the enclosed items
$\{\}_{0+}$	the enclosed items occur 0 or more times
$\{\}_{1+}$	the enclosed items occur 1 or more times
Σ	Start symbol

Other items are terminal symbols of the language.

BNF Rewriting Rules (Productions)

Σ ::= **two character sequence** | **parameterized_form1** | **parameterized_form2**

esc ::= 1BH

space ::= 20H

right_brace ::= 7BH

w ::= 88H

data ::= 0 ... 255

two_character_sequence ::= **esc** 0 **|** 1 **|** 2 **| ... | right_brace |** ~

parameterized_character ::= ! **|** " **|** # **| ... |** - **|** . **|** /

X ::= **parameterized character**

group_character ::= ’**|** a **|** b **| ... | right_brace** | ~

y ::= **group_character**

parameter character ::= ’**|** a **|** b **| ...** | **right_brace |** ~

z_i ::= **parameter_character**

terminating_character ::= @ **|** A **|** B **| ... |**] **|** ^

Z_n ::= **terminating_character**

digit ::= 0 **|** 1 **|** 2 **|** 3 | 4 | 5 **|** 6 **|** 7 | 8 **|** 9

integer ::= **{ digit }**$_{1+}$

fraction ::= **. { digit }**$_{0+}$

float ::= **integer fraction**

number ::= **integer** | **fraction** | **float**

unsigned_value ::= **{ space }**$_{0+}$ **number**

signed_value ::= **{ space }**$_{0+}$ **{ + | - }**$_1$ **{ space }**$_{0+}$ **unsigned_value**

value_field ::= **{ unsigned_value | signed_value }**$_{0+}$

::= **value_field**

parameterized_form1 ::= **esc X y { # z_i }**$_{0+}$ **# Z_n**

parameterized_form2 ::= **esc X # Z_n**

parameterized_form3 ::= **esc X y { # w { data }**$_{0+}$ **}**$_{0+}$ **# W { data }**$_{0+}$

4.7 Converting a Floating Point Number

Some of the fields within the structure of a binary transfer may be in IEEE floating point. The following formula converts a floating point number into an IEEE 754-1985 4-byte encoded floating point number.

Variable	Meaning
X	Floating point number in base 10
Sign	X's sign (either 1 or -1)
S	Sign bit (1 if Sign is -1, 0 if Sign is 1)
E	X's biased base 2 exponent in base 10
F	X's base 2 fraction in base 10

Formula to convert IEEE 754-1985 number into native machine format

```
If E = 0 And F = 0 Then
    X = 0
Else
    X = Sign * ( 1 + ( F / 2^23 ) ) * 2^( E - 127 )
End If
```

Formula to convert native machine number into IEEE 754-1985 format

```
If X = 0 Then
    S = 0
    E = 0
    F = 0
Else
    Sign = X / ABS(X)
    S = ( 1 - Sign ) / 2
    E = INT( LOG( ABS(X) ) / LOG(2) ) + 127
    F = INT ( ( ( X / 2^(E-127) ) - 1 ) * 2^23 )
End If
```

Example:

```
X = 201.187
Sign = 201.187 / ABS(201.187)
Sign = 201.187 / 201.187
Sign = 1
S = ( 1 - Sign ) / 2
S = ( 1 - 1 ) / 2
S = 0
E = INT( LOG( ABS(201.187) ) / LOG(2) ) + 127
E = INT( LOG(201.187) / LOG(2) ) + 127
E = INT( 2.303599914701 / .301029995664 ) + 127
E = INT( 7.652393276024 ) + 127
E = 7 + 127
E = 134 in base 10 representation, which is 10000110 in base 2
F = INT ( ( ( 201.187 / 2^(134-127) ) - 1 ) * 2^23 )
F = INT ( ( ( 201.187 / 2^7 ) - 1 ) * 2^23 )
F = INT ( ( 1.5717734375 - 1 ) * 2^23 )
F = INT ( 0.5717734375 * 2^23 )
F = INT ( 0.5717734375 * 8388608 )
F = INT ( 4796383.232 )
F = 4796383 in base 10 representation, which is
               10010010010111111011111 in base 2
```

To check:

```
X = Sign * ( 1 + ( F / 2^23 ) ) * 2^( E - 127 )
X = 1 * ( 1 + ( 4796383 / 2^23 ) ) * 2^( 134 - 127 )
X = ( 1 + 0.5717734375 ) * 2^7
X = 1.5717734375 * 128
X = 201.18699646
```

Chapter 5: Text Processing

Contents of this Chapter

This chapter describes the following PCL commands:

5.1 Introduction

Whereas Chapter 4 described the processing of character codes within escape sequences, this chapter describes the processing of character codes other than escape sequences.

PCL devices normally print any character code that is not a valid escape sequences or control code. A device receiving a valid escape sequence or control code suspends character formatting, performs the specified action, and begins formatting again.

The following terms are used in this chapter:

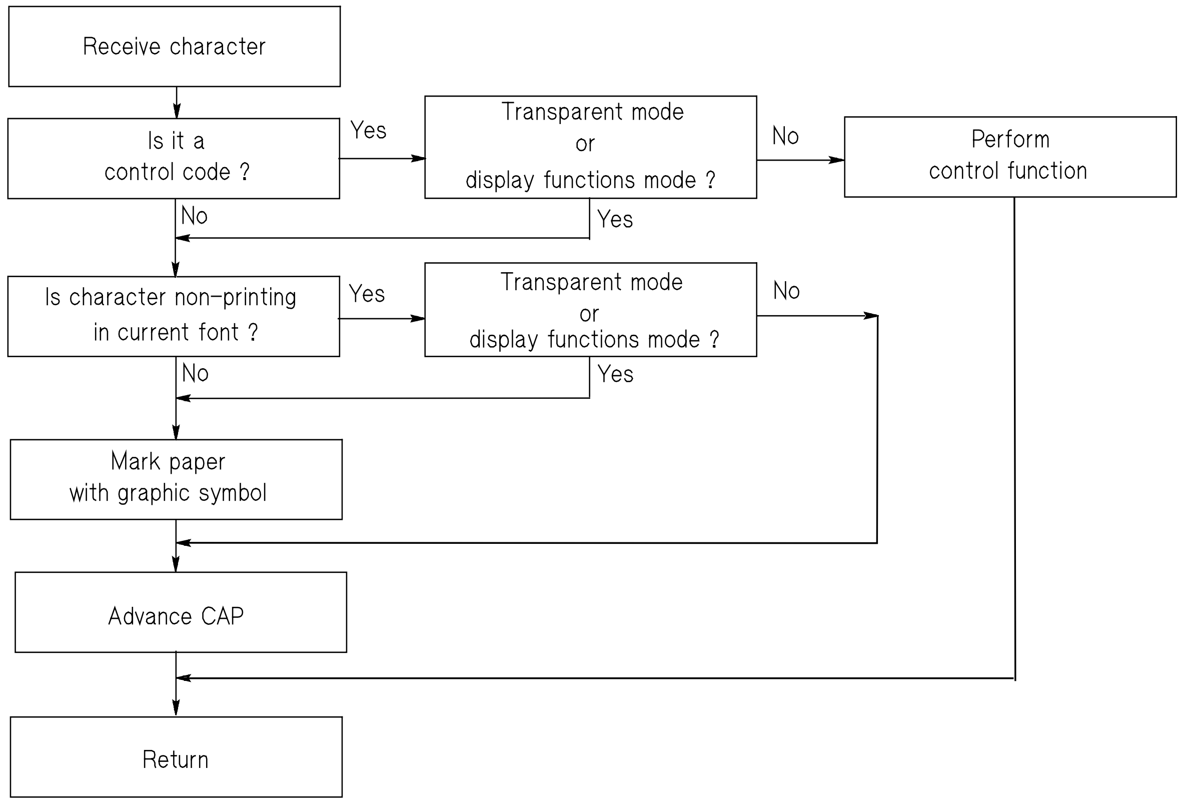
Character Code — n-byte binary code representing a character.

Control Code — A character code that initiates a device function (e.g., formfeed). Graphics symbols associated with control codes may be printed by using the Transparent Data Transfer command.

Symbol Set — A mapping of character codes to glyphs (graphic symbols).

Character Processing

The simplified flowchart below shows how an incoming character code is handled.


Receive character
Is it a control code ?
Yes
Transparent mode or display functions mode ?
No
Perform control function
No
Yes
Is character non-printing in current font ?
Yes
Transparent mode or display functions mode ?
No
No
Yes
Mark paper with graphic symbol
Advance CAP
Return

Generic Character Code Chart

Figure 5.2 shows a generic character code chart. The areas, special codes, and control codes used in PCL are marked.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL		SP													
1																
2																
3																
4																
5																
6																
7	BEL															
8	BS															
9	HT															
A	LF															
B	VT*	ESC														
C	FF															
D	CR															
E	SO															
F	SI															
	AREA0		AREA1						AREA2		AREA3					

* Vertical Tab (VT) is obsolete.

5.2 Control Codes and Special Codes

Control Codes are character codes that execute a unique device function (e.g., formfeed); however, some control codes may cause a non-operation. All the symbol set types defined for PCL devices contain the ten control codes and the SPACE character described below.

Control codes may be associated with graphic symbols that are printable in Transparent Data Transfer or Display Functions Mode. A graphic symbol downloaded to a control code is a *hidden* graphic and is only printable by Transparent Data Transfer (*Esc&p#X*) or Display Functions Mode (*EscY*).

Backspace <*BS*>

Moves CAP one character position backward on the current line.

In horizontal text path mode (*Esc&c#T*), no action occurs if CAP is already at the left margin. If CAP is to the left of the left margin (via a *Move CAP* command), BS functions as if the left margin is column 0, the logical page left boundary.

In vertical text path mode (*Esc&c#T*), no action occurs if CAP is already at the top margin. If CAP is above the top margin (via a Move CAP command), BS functions as if the top margin is row 0, the logical page top boundary.

In proportional spacing, a single BS centers the overstriking character with the character being overstruck. After printing the overstrike character, CAP is at the same position as before the BS. Multiple backspaces each move back the distance of the last printable character or space.

In HP-GL/2 mode, a backspace as the first character of a label is ignored.

Bell <BEL>

Causes a bell or audible alarm to sound if one is present; otherwise causes no action.

Carriage Return <*CR*>

In horizontal text path mode (*Esc&c#T*), CR moves CAP to the left margin on the current line. In vertical text path mode, CR moves CAP to the top margin on the current line. If CAP is left of the left margin, CR moves CAP to the right until it is coincident with the left margin.

In HP-GL/2 mode, the pen location is updated to the carriage return point, usually the pen location when the LB command was executed, adjusted by any line feeds.

Escape <ESC>

Provides supplementary control of printer functions. The escape character itself is a prefix for the string of one or more characters which follow. Once an escape character is received (unless it appears within binary data), normal text processing is suspended until the supplemental function has been activated or the escape sequence has been determined to be invalid.

Formfeed   <*FF*>

In horizontal text path mode (*Esc&c#T*), FF moves CAP to the same horizontal position at the top of form on the next page. ***Top of form*** = top margin + (3/4 × line spacing). In vertical text path mode (*Esc&c#T*), FF moves CAP to the same vertical position at [right margin - (3/4 × line spacing)].

Multiple formfeeds in sequence are not interpreted as a single formfeed.

An *implicit* formfeed is performed if printable data has been sent before *EscE*, a LF that moves CAP into the perforation skip area (skip is enabled) or onto the next page, by page size or orientation changes, or by any other command that ejects a "dirty" page.

Horizontal Tab   <*HT*>

HT moves CAP to the next tab stop on the current line (in vertical text path mode, the current line extends vertically from top to bottom). Tabs represent a logical position and thus refer to different physical positions for different settings of CMI.

CMI determines current column width. If CMI is changed, the physical location of each tab stop moves. HT has no affect if the CMI is 0.

In horizontal text path mode (*Esc&c#T*), the left margin is the first tab stop; additional tab stops are fixed at every 8 columns to the right margin. In vertical text path mode, the first tab stop is at the top margin; additional tab stops are fixed at increments of (8 * current CMI) to the bottom margin.

The following are some boundary cases for horizontal text path mode:

- If the requested tab stop is to the right of the right margin and CAP is at or to the left of the right margin, HT moves CAP to the right margin.
- If the requested tab stop is to the right of the printable area and CAP is to the right of the right margin, HT moves CAP to the right edge of the printable area.
- If CAP is to the left of the left margin, HT moves CAP to the left margin.

The following are some boundary cases for vertical text path mode:

- If the requested tab stop is outside the bottom margin and CAP is at or above the bottom margin, HT moves CAP to the bottom margin.
- If the requested tab stop is outside the printable area and CAP is below the bottom margin, HT moves CAP to the edge of the printable area.
- If CAP is above the top margin, HT moves CAP to the top margin.

Tabs do not cause lines to be wrapped if end-of-line wrap mode is enabled.

NOTE: The current unit of measure setting (*Esc&u#D*) affects HT on devices implementing the Unit of Measure command (*Esc&u#D*). For example, if the unit of measure is 300 (one PCL Unit = 1/300 inch), the default CMI is rounded to the nearest 1/300 inch. Rounding always occurs, whether CMI has been implicitly set by font selection or explicitly set by the CMI command (*Esc&k#H*).

DEVICE NOTE: Although DJ850C implements the Unit of Measure command, HT is not affected by the unit of measure setting.

Linefeed <*LF*>

In horizontal text path mode (*Esc&c#T*) , LF moves CAP to the same horizontal position one row down. If perforation skip mode is enabled, a LF that would go beyond the text length boundary moves CAP to the same horizontal position at the top of form on the next page. If perforation skip mode is disabled, text is printed to the end of the page and onto the top edge of the next page. Text in the unprintable region may be lost.

DEVICE NOTE: If perforation skip is on and CAP is within the top margin area, a LF moves CAP one line down on LJs and DJs above 1000, and to the top of form on DJs below 1200.

In vertical text path mode (*Esc&c#T*), LF moves CAP to the same vertical position one column to the left. If perforation skip mode is enabled, a LF that would cause CAP to go beyond the left margin moves CAP to the same vertical position at the right margin of the next page; that is, a linefeed causes a perforation skip. If perforation skip mode is disabled, text is printed to the left margin and onto the right edge of the next page. Text in the unprintable region may be lost.

DEVICE NOTE: If perforation skip is on and CAP is within the right margin area, LF moves CAP one column to the left on LJs and DJ6xx, and to the right margin on DJs below 1200.

In HP-GL/2 mode, the pen position is advanced one line. For HP-GL/2 labels, a line is defined to be the height of the character cell.

Null <NUL>

Causes no action in a PCL printer.

Shift Out <SO>

Invokes the currently designated secondary font as the active font. Subsequent characters are printed from the secondary font, which remains active until a Shift In or Reset (*EscE*). The default is **not** Shift Out. (See Chapter 9)

Shift In <SI>

Invokes the currently designated primary font as the active font. Subsequent characters are printed from the primary font, which remains active until a Shift Out. The default is Shift In. (Chapter 9)

Space <*SP*>

Moves CAP forward one character position (defined by the CMI) on the current line.

In horizontal text path mode (*Esc&c#T*), CAP does not move if it is already at the right margin and end-of-line wrap is not enabled. If end-of-line wrap is enabled, CAP moves to the left margin of the next line and then prints the space. Trailing spaces are also printed.

In vertical text path mode, CAP does not move if it is already at the bottom margin and end-of-line wrap is not enabled. If end-of-line wrap is enabled, CAP moves to the top margin of the next line and then prints the space. Trailing spaces are also printed.

Space may be a either printable character or a control code. If a glyph is defined for the space code, space is printable; otherwise, it is a control code. For proportionally-spaced fonts, a space control code updates CAP by the current CMI value; however, a printable space moves CAP the width of the character. For fixed-pitch fonts, a space, whether control code or printable, updates CAP according to the CMI value.

A printable space fixes CAP and makes the page "dirty", subject to conditional page ejects. An unprintable space does not make the page dirty.

DEVICE NOTE: On DJ6xx and 8xx all spaces, printable and unprintable, make the page dirty.

The space character may be redefined in a downloadable font. When space is redefined in this way, the font's default CMI changes to match the new width of space, not the current CMI.

DEVICE NOTE: DeskJets below 600 do not allow redefinition of SPACE.

In HP-GL/2 mode, the pen position is updated one column to the right of the current position. The space width may be modified by the ES command.

Line Termination *Esc & k # g/G*

Controls how CR, LF, and FF are interpreted.

Value(#)		=	0	CR = CR;	LF = LF;	FF = FF
=	1	CR = CR,LF;	LF = LF;	FF = FF		
=	2	CR = CR;	LF = CR,LF;	FF = CR,FF		
=	3	CR = CR,LF;	LF = CR,LF;	FF = CR,FF		
Default		=	0			
Range		=	0 to 3 (out-of-range values are ignored)			

For example, a value field of 1 causes the printer to insert a carriage return (CR) and linefeed (LF) control code for every CR. A linefeed or formfeed is sent as is.

In Display Functions Mode (*EscY*), LF and FF are not executed, but a glyph is printed if it exists for that character code, or a space if it does not. Display Functions handles a CR by printing the glyph or space and then executing a CR-LF.

5.3 Symbol Sets

A symbol set is a mapping from character codes to glyphs (graphic symbols). PCL defines four types of symbol sets: HP-7 and HP-8, modeled after the ISO-646 and ISO-8859 definitions for character coding, PC-8 for IBM PC compatibility, and *large* (2-byte) to provide more characters for Asian fonts.

In the diagrams below:

Hatch	Indicates control codes no longer implemented
Solid	Indicates control codes that are still implemented.

HP-7 Compatible (96 Glyph)

The HP-7 type is a unique subset of the HP-8 type, except that characters are defined only in **area0** and **area1** (0 through 127). This type is recommended for new designs.

AREA0
AREA1
AREA2
AREA3

NOTE: A ***printable*** graphics symbol is one that prints a mark on the paper, or advances CAP.

Area0 contains nonprinting/nonspacing characters that are ignored or processed as control codes. In Transparent Data Transfer a graphic symbol is printed and CAP is advanced. In Display Functions Mode a graphic symbol is printed, CAP is advanced, and CR is executed as CR-LF.

Area1 characters are printable graphic symbols that mark the page and advance CAP. **Area2** characters are nonprinting/nonspacing: no marks are printed and CAP is not advanced. **Area3** characters advance CAP but print no mark. In Display Functions Mode or Transparent Data Transfer, character codes in **area2** and **area3** advance CAP by the CMI distance.

Area0 and **area1 c**haracters may be downloaded with graphic symbols. Attempts to download an **area2** or **area3** character are ignored.

DEVICE NOTE: DeskJets below 600 except the 540 treat a character code in area2 as if it were in area0. A character code in area3 is treated as if like a graphic character in area1.

HP-8 Compatible (192 Glyph)

The HP-8 type is based on ISO-8859 coding for 8-bit symbol sets. As shown below, **area0** and **area2** are treated as nonprinting/nonspacing characters or control codes; and **area1** and **area3** are printing characters.

Area0 contains nonprinting/nonspacing characters that are ignored or processed as control codes. In Transparent Data Transfer a graphic symbol is printed if the glyph exists and a space if it does not; then CAP is advanced. In Display Functions Mode a graphic symbol is printed, CAP is advanced, and CR is executed as CR-LF.

Area1 and **area3** characters are printable graphic symbols that advance CAP.

Area2 characters are nonprinting/nonspacing. However, in Transparent Data Transfer a graphic symbol is printed and CAP is advanced. In Display Functions Mode a graphic symbol is printed, CAP is advanced, and CR is executed as CR-LF.

The character at position A0h is treated like any other graphic symbol. Many symbol sets define it as a blank with a fixed width set by the current font; it is then not processed like the space character.

All character codes may be downloaded with graphic symbols. Symbols downloaded to codes located in **area0** and **area2** are hidden graphics and are only printable in Transparent Data Transfer or Display Functions Mode.

PC-8 Compatible (256 Glyph)

The PC-8 type is used for compatibility with IBM PCs. As shown below, **area0**, **1**, **2**, and **3** have no meaning; only control codes 00, 07-0F, and 1B are nonprinting.

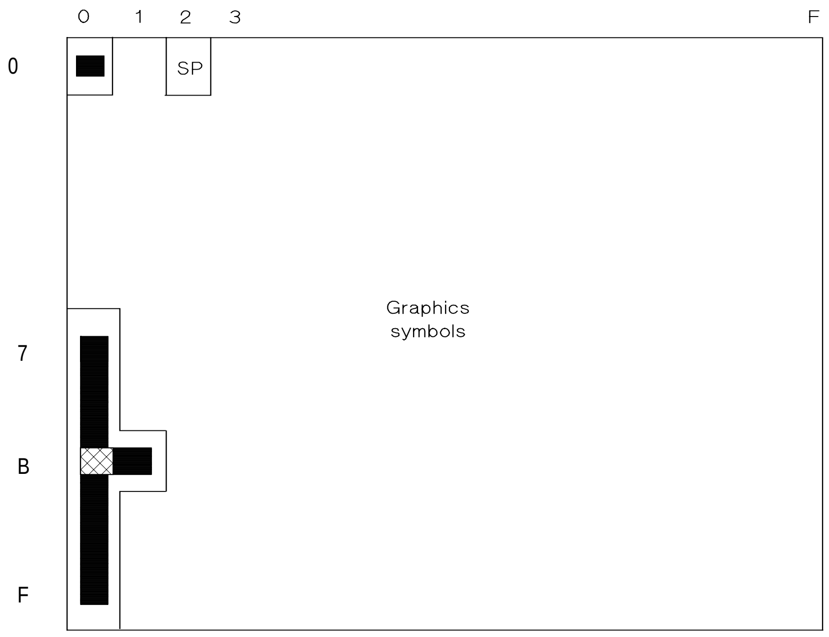


Figure 5-5 PC-8 Compatible Set

Characters defined as control codes are nonprinting/nonspacing and are processed as control codes. In Transparent Data Transfer a graphic symbol is printed and CAP is advanced. In Display Functions Mode a graphic symbol is printed, CAP is advanced, and CR is executed as CR-LF.

A character not defined as a control code is a printable glyph that advances CAP.

The character at position F/F is treated like any other graphic symbol. Many symbol sets define it to be a blank, but this blank is of a fixed width set by the current font and is not processed like the space character. This is allowed by handling this character as a graphic and not a space.

Symbol sets Offered by HP

Many of the symbol sets offered by HP are based on standard codings provided by organizations like ISO (International Standards Organization) and ANSI (American National Standards Institute). Nonstandard sets have been implemented in unique situations, or when an application has required special characters. See Chapter 9, Font Selection, for a complete list of the symbol sets and PCL identifiers that HP has implemented or reserved for future products.

5.4 Printing Hidden Characters

The following options allow the printing of character codes that normally have hidden graphic symbols. Transparent Data Transfer (*Esc&p#X*) disables control code functions and instead prints graphic symbols for a specified number of bytes. Display Functions Mode disables and prints all escape sequences and control codes (except CR and *EscZ*).

Transparent Data Transfer *Esc & p # x/X*

Prints the graphic symbols associated with hidden character control codes.

Value(#)	=	Number of binary bytes to be printed ("binary" means not parsed)
Default	=	NA
Range	=	0 to 2^{32}-1

All subsequent character codes for the specified number of bytes are printed with the current font attributes. The parser ignores all control codes, including the *Esc* character; instead, the code's graphic symbol in the current symbol set is printed. For example, in the PC-8 symbol set, *Esc* is printed as a left arrow.

NOTE: Text Parsing Method (*Esc&t#P*) determines how transparent data may be interpreted.

Display Functions Mode ON *Esc Y*

This command prints a character code in the currently active font. Turning this mode ON has the following effects:

- All control code and escape sequence functions except *CR* and *EscZ* are disabled. *CR* marks the paper and executes CR-LF. *EscZ* marks the paper and disables Display Functions Mode.
- All character codes either mark the paper or produce a blank space.

This mode is intended as a programmer's debugging aid, and is not to be used for document preparation.

NOTE: Text Parsing Method (*Esc&t#P*) determines how character codes may be interpreted.

Display Functions Mode OFF *Esc Z*

This command turns off display functions mode. If display functions mode is ON when *EscZ* is received, the characters for the sequence are printed, and the mode is disabled. If display functions mode is OFF when *EscZ* is received, no operation is performed. The default is OFF.

5.5 Text Enhancements

Enable Underline *Esc & d # D*

Enables the automatic text underline enhancement.

Value(#)	=	0	Default single underline
	=	1	Single underline, fixed location below the baseline
	=	2	Double underline, fixed location below the baseline
	=	3	Single underline, font dependent location (floating)
	=	4	Double underline, font dependent location (floating)
Default		=	0
Range		=	0 to 4 (the default is selected for out-of-range values)

NOTE: This command must use a capital "D" as a terminator.

When underline is enabled, any positive horizontal motion, including spaces and CAP moves, is underlined (except a CR when CAP is to the left of the left margin). Underline remains enabled until explicitly disabled. The default state is underline disabled.

A single underline is produced if double underline is invoked but unavailable.

The only mode that must be implemented is 0.

In *fixed-position* underlining, the underline is drawn a fixed distance below the baseline. The distance is device-dependent, but font independent.

In *floating-position* underlining, the greatest underline distance specified in the font descriptors of all the fonts printed on the current line determines the underline position.

This command is ignored when the Text Path Direction (*Esc&c#T*) is set to vertical printing (1).

DEVICE NOTE: The floating underline position on DeskJets below 1200 is font-dependent: each font uses a different underline, and the greatest underline distance is ignored (i.e., LaserJets). DeskJets below 600 except the 540 implement a value of 3 or 4 and ignore out-of-range values.

Disable Underline *Esc & d @*

Disables automatic text underlining.

End-of-Line Wrap *Esc & s # c/C*

Defines the action that occurs when a line of text reaches the right margin in horizontal text path mode (*Esc&c#T*), or the bottom margin in vertical text path mode.

Value(#)	=	0	Enables End-of-Line Wrap
	=	1	Disables End-of-Line Wrap
Default		=	1
Range		=	0,1

When end-of-line wrap is enabled in horizontal text path mode (*Esc&c#T*), a character or space that would move CAP to the right of the right margin causes a CR-LF to be executed (prior to the printing of the character or space). In vertical text path mode, a character or space moving CAP below the bottom margin causes a CR-LF to be executed (prior to the printing of the character or space).

When end-of-line wrap is disabled in horizontal text path mode, a character or space that would move CAP to the right of the right margin is clipped (i.e., the entire glyph is not printed) and CAP is set to the right margin. In the vertical text path mode, a character or space that would move CAP below the bottom margin is not printed and CAP is set to the bottom margin.

If margins are set so that a given character cannot fit on a line, the character is clipped even if end-of-line wrap is enabled.

The character's escapement (delta X) determines whether or not the glyph fits.

DEVICE NOTE: DJs above 1000 use black width, not delta X, to determine whether to line wrap. DJs below 1000 determine whether to wrap by a character's cell width in fixed and multi-pitched fonts, and by escapement (delta X) in proportionally-spaced fonts.

5.6 Escapement Encapsulated Text

The Escapement Encapsulated Text (EET) command corrects for the following two situations:

- The printer’s linear TrueType scaler produces different escapements than the non-linear TrueType scaler used by Windows drivers.
- Unlike printer-based escapements, Windows applications justify text by changing both inter-word and inter-character spacing, causing a different glyph placement.

Horizontal CAP moves (*Esc&a#C*, *Esc&a#H*, *Esc*p#X*) can correct escapement (absolute CAP moves are needed because the driver does not know where the printer had updated CAP); however, then the driver must convert the escapement to an ASCII string and the printer’s escape sequence parser must convert the string to a binary number. It is more efficient, using fewer bytes, to send the escapement as binary and not in the escape sequence as ASCII.

Low-end printer memory costs suggest intermixing of characters and escapements, rather than grouping escapements and characters separately. (Grouping was originally proposed because GDI sends the character to the driver before the escapement to accommodate devices that render the characters as soon as they are received.)

The signed nature of the escapement allows for a negative or positive escapement. Text Path Direction (*Esc&c#T*) determines which direction is positive or negative.

The EET command supports “printable” characters by treating all characters like characters in Transparent Data Mode (*Esc&p#X*) — no control code functions will be executed.

Escapement Encapsulated Text *Esc & p # w/W*

Transfers a text string with an encapsulated escapement value for each text character. The escapement contained in font data is overridden.

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

DEVICE NOTE: DJ850C supports a range of 4 to 32767.

The data field must contain byte-aligned binary data, not ASCII. Invalid configurations are ignored and the specified number of data bytes discarded. Value field signs are ignored. Extra bytes are discarded.

This command handles the character string in a single line. The End-of-Line Wrap (*Esc&s#C*) state has no affect on this command.

Character placement at the right margin follows existing text rules: the character is rendered if either of the following is true; otherwise the entire character is clipped and CAP is unchanged.

- Character origin and escapement are to the left of the right margin.
- Character origin is to the left of the right margin; escapement is to the left of the right edge of the logical page.

If subsequent escapements are negative and characters again follow either of the above two rules, then characters are rendered. If the string starts to the right of the right margin, then character placement at the right edge of the logical page follows existing text rules.

DEVICE NOTE: DJ850C ignores Right Margin (*Esc&a#M*). Character placement at the right edge of the logical page follows text rules.

Unit of Measure (*Esc&u#D*) sets escapement unit size. Escapement units that do not add up to an integral number of dots are converted to a higher internal resolution and the error is collected. When the fractional portion of a dot exceeds 0.5 dots, the next dot position is rounded up.

The current underline state of Enable Underline (*Esc&d#D*) has no effect on this command because encapsulated text is not underlined.

Data field format is as follows:

	15 (MSB) 8	**7 (LSB) 0**	
	Format # = 0 (UBYTE)	Text Character 1 (UBYTE)	
	Escapement Value for Text Character 1 (SINT16)		
	Text Character 2 (UBYTE)	Escapement Value for Text Character 2 MSB (SINT16)	
	Escapement Value for Text Character 2 LSB (SINT16)		
	…		
	Text Character *n*-1 (UBYTE)	Escapement Value for Text Character *n*-1 MSB (SINT16)	
	Escapement Value for Text Character n-1 LSB (SINT16)	Text Character *n* (UBYTE)	
	Escapement Value for Text Character *n* (SINT16)		

Byte 0: Format Number

Value = 0 CAP relative escapements

A value of 0 configures for encapsulated 8-bit text characters with escapements relative to the previous CAP. The command is ignored for other values and the specified number of data bytes is discarded.

Byte 1: Text Character 1

Value = 0 to 255

Eight-bit value of the text character 1.

Bytes 2&3: Escapement Value for Text Character 1

Value = -32767 to 32767

Determines the new CAP position relative to the CAP after placing Text Character 1. The distance is measured in PCL Units.

DEVICE NOTE: DJ850C supports a range of 0 to 32767.

Byte 4: Text Character 2

Value = 0 to 255

Eight-bit value of the text character 2.

Bytes 5&6: Escapement Value for Text Character 2

Value = -32767 to 32767

Determines the new CAP position relative to the CAP after placing Text Character 2. The distance is measured in PCL Units.

DEVICE NOTE: DJ850C supports a range of 0 to 32767.

Byte 3*n*-5: Text Character *n*-1

Value = 0 to 255

Eight-bit value of the text character *n*-1.

Bytes 3*n*-4 & 3n-3: Escapement Value for Text Character *n*-1

Value = -32767 to 32767

Determines the new CAP position relative to the CAP after placing Text Character n-1. The distance is measured in PCL Units.

DEVICE NOTE: DJ850C supports a range of 0 to 32767.

Byte 3*n*-2: Text Character *n*

Value = 0 to 255

Eight-bit value of the text character *n*.

Bytes 3*n*-1 & 3*n*: Escapement Value for Text Character *n*

Value = -32767 to 32767

Determines the new CAP position relative to the CAP after placing Text Character *n*. The distance is measured in PCL Units.

DEVICE NOTE: DJ850C supports a range of 0 to 32767.

5.7 Text Parsing Method

The PCL parser must know whether one-byte or n-byte character codes are being sent. Text Parsing Method (*Esc&t#P*) lets PCL parse multiple-byte characters regardless of the large symbol set (e.g., Unicode, JIS, Shift-JIS) in use.

Text Parsing Method *Esc & t # p/P*

Specifies how the PCL data stream is to be parsed.

Value(#)	=	0,1	Character codes are processed as 1-byte characters.
	=	2	Strict 2-byte processing. Characters and control codes are represented by two bytes. The Unicode standard is an example.
	=	21	Printable characters are 2 bytes; control codes are 1 byte. Control code range is 0 to 0x20. A first byte between 0x21 to 0xFF is considered to be the upper byte of a 2-byte character.
	=	31	Shift JIS parsing. 1-byte control code range is 0-0x20; 1-byte character range is 0x21 to 0x80, 0xA0 to 0xDF, and 0xFD to 0xFF. 2-byte characters have a first byte between 0x81 to 0x9F or 0xE0 to 0xFC.
	=	38	If the 8th bit is set, the byte is the first byte of a 2-byte character code. If the 8th bit is not set, the byte represents a 1-byte character code.
Default	=	31 if the default symbol set is WIN31J; 38 if the default symbol set is GB2312-80; otherwise it is 0.	
Range	=	0,1,2,21,31,38 (other values map to 0)	

The default depends on the default PCL symbol set.

In strict 2-byte parsing (value=2), the parser cannot distinguish between an <ESC> in the data stream and the upper byte of a 2-byte character. Therefore, in strict two-byte processing, the host must send a <NULL> before each escape sequence. To start an escape sequence, 0x001B is sent and then the rest of the escape sequence.

If the value is 21, character codes between 0x21 to 0xFF are processed as the first byte of a two-byte character, and the next byte is processed as the second byte. Character codes outside this range are processed as one-byte values. This method can be used for parsing characters in Asian seven-bit encoding specifications, i.e., JIS X0208 (Japan).

If the value is 31, character codes between 0x81 to 0x9F and 0xE0 to 0xFC are processed as the first byte of a two-byte character, and the next byte is processed as the second byte. All character codes outside this range are processed as one-byte values. This method can be used for parsing characters in the Shift-JIS encoding specification.

If the value field is 38, character codes between 0x80-0xFF are processed as the first byte of a two-byte character, and the next byte is processed as the second byte. All character codes outside this range are processed as one-byte values. This method can be used for parsing Asian eight-bit encoding specifications, such as the Big Five and TCA encoding specifications (Taiwan) and KS C 5601-1992 (Korea), and GB 2312-80 (China).

When printing characters, the font from which the character is printed may be incompatible with the text processing method. For example, the user may have selected strict two-byte parsing, but the current font is an 8-bit font. When this happens, the requested character code is compared against the range of the currently selected font. If the character code is within range, the character is printed; otherwise, the character is considered to be out of range and no action is performed.

The Transparent Data command (*Esc&p#X*) prints character codes that are normally not printed. Since the value field specifies the number of bytes to be interpreted, the Transparent Data command is extended to interpret *n*-byte character codes. The selected text parsing method determines how the transparent print data should be interpreted. If, when reading the last byte of transparent data, another byte of data is required to complete the character, but there is none, the last byte is discarded.

EscE defaults text parsing method, which is part of the modified print environment. Therefore, text parsing method is saved by a macro call or overlay, but not by a macro execution. Text parsing method is defaulted at the beginning of an overlay macro and restored at completion.

Chapter 6: Printer Control

Contents of this Chapter

This chapter contains information on PCL job control, printer diagnostics, device-specific printer control, and undocumented internal HP escape sequences.

This chapter describes the following commands:

6.1 PCL Job Control

Job control languages like PJL and PML can perform job setup. PCL also contains many job control commands for setting printer features to a known state at the beginning of a job.

Although there may be exceptions, there are two general rules governing when to use a job control language and when to use PCL for setting environment variables:

1. Job control language is used for language switching (e.g., PCL to PostScript).
2. Job control language is used by the system administrator for general network setup.
3. PCL job setup is used for local setup of a specific machine. It is analogous to control panel setup for a particular job. After that job, the control panel is reset to user defaults.

At the beginning of every print job, PCL job control commands are issued before sending print data. Job control commands are usually grouped together and sent at job boundaries. Job control commands control the output of the entire print job and are grouped together at the beginning of a print file.

Reset *Esc E*

Returns all device features to their user-defaults (control panel settings). Control-panel settings are used because they can override factory defaults. Specifically, *EscE* performs the following:

- Prints all data received before the reset.
- Moves CAP to the intersection of the top of form and default left margin on the current page if the page is clean (no printable data has been sent); otherwise the page is ejected and CAP goes to the intersection of the top of form and default left margin on the next page.
- Deletes downloaded temporary (not permanent), fonts, symbol sets, patterns, and macros.
- Returns control to PCL if executed in HP-GL/2 mode.
- Executes an HP-GL/2 *IN* command (but does not create a default HP-GL/2 palette).
- Disables the auto macro overlay.
- Clears the palette stack
- Defaults the following:
  - Color palette to a 2-pen non-programmable black and white palette
  - Color lookup tables
  - Color pipeline
  - Logical page size
  - Logical page orientation
  - Print direction
  - Picture frame
  - Render algorithm
  - Viewing illuminant
  - Gamma correction
  - Print model logical operators
  - Foreground color
  - Source and pattern transparency modes
  - Text parsing method
  - Text path direction

EscE should be the first PCL command at the beginning of a job to establish default conditions; and it should be the last command at the end of a job to leave the machine in the user default state and clear any partially composed pages. *EscE* has no effect on I/O and causes no disruption in host-to-peripheral communication. The printer remains on-line and no data is lost.

EscE is a valid HP-GL/2 terminator and is the same as *IN*, but in addition:

- Creates a PCL 2-pen non-programmable black and white palette
- Defaults CAP
- Defaults to the PCL logical page orientation
- Defaults the picture frame
- Defaults the anchor point
- Defaults the HP-GL/2 plot size

Within an HP-GL/2 label in normal mode (TD0), *EscE* is a special case: it causes a device to reset and transition to the PCL environment. In transparent data mode (TD1), *EscE* does not reset the device.

Job Separation *Esc & l # t/T*

Provides a means for distinguishing print job boundaries when multiple print jobs exist in the output tray.

Value(#)	=	0	(obsolete) Sets stacking position to a known state
	=	1	Performs a job offset
Default		=	NA
Range		=	1

This command should be sent at the beginning of a print job, immediately following *EscE*.

Job separation occurs before a page goes to the output bin if the command is received before closing the page (the back side of the page in duplexing mode). Multiple job separation commands on the same page cause only one offset.

DEVICE NOTE: LJ500 physically offsets jobs. *Esc&l0T* sets the stacker to the default position. *Esc&l1T* sets the stacker to the position opposite that of the previous position.

DEVICE NOTE: LJ2000 performs job separation by inserting an identifying sheet edged with a heavy black bar between jobs.

DEVICE NOTE: LJIIISi and LJ4Si physically offset the jobs in the output stacker. Job separation is not supported when the face-up output tray is selected.

DEVICE NOTE: The HP5000 fanfold prints 'tick' marks between jobs on tractor feed strips.

DEVICE NOTE: DJs below 1200 do not eject the current page.

Media Destination *Esc & l # g/G*

Selects the destination bin for which the print job should be delivered.

Value(#)	=	0	Automatic selection
	=	1	Selects destination tray #1
	=	2	Selects destination tray #2
	=	3	Selects destination tray #3
Default		=	0
Range		=	0 to 3

DEVICE NOTE: LJIIISi and 4Si support values 1 and 2 only; tray #1 is the stacker unit on top of the printer and tray #2 is the face-up tray on the rear of the printer.

DEVICE NOTE: Color LJ supports 0, 1, and 2. Tray 1 is defined to be the correct-order, face-down stacker on top of the printer; tray 2 refers to the pull-out tray (which also serves as the lid to the main cassette) for reverse-order, face-up outputs.

Simplex/Duplex *Esc & l # s/S*

Designates either single-sided or double-sided printing mode for duplex printers.

Value(#)	=	0	Simplex
	=	1	Duplex, vertical binding
	=	2	Duplex, horizontal binding
Default		=	0
Range		=	0 to 2

Horizontally bound duplexed pages are bound along the width (short side) of the physical page. Horizontal binding duplex mode rotates back side data 180 degrees from front side data (Figure 6-1).

Vertically bound duplexed pages are bound along the length (long side) of the physical page. Vertical binding duplex mode does not change orientation from front to back (Figure 6-2).

This command causes a conditional page eject (if printable data has been sent).

DEVICE NOTE: DeskJets below 1200 do not eject the page.

Duplex Page Side *Esc & a # g/G*

Designates which side of the sheet to print.

Value(#)	=	0	Next side
	=	1	Front side
	=	2	Back side
Default		=	0
Range		=	0 to 2

This command causes a conditional page eject (if printable data has been sent). It may be used to skip a page; for example, a chapter typically begins on the front (odd) side of a page.

DEVICE NOTE: DeskJets below 1200 do not eject the page.

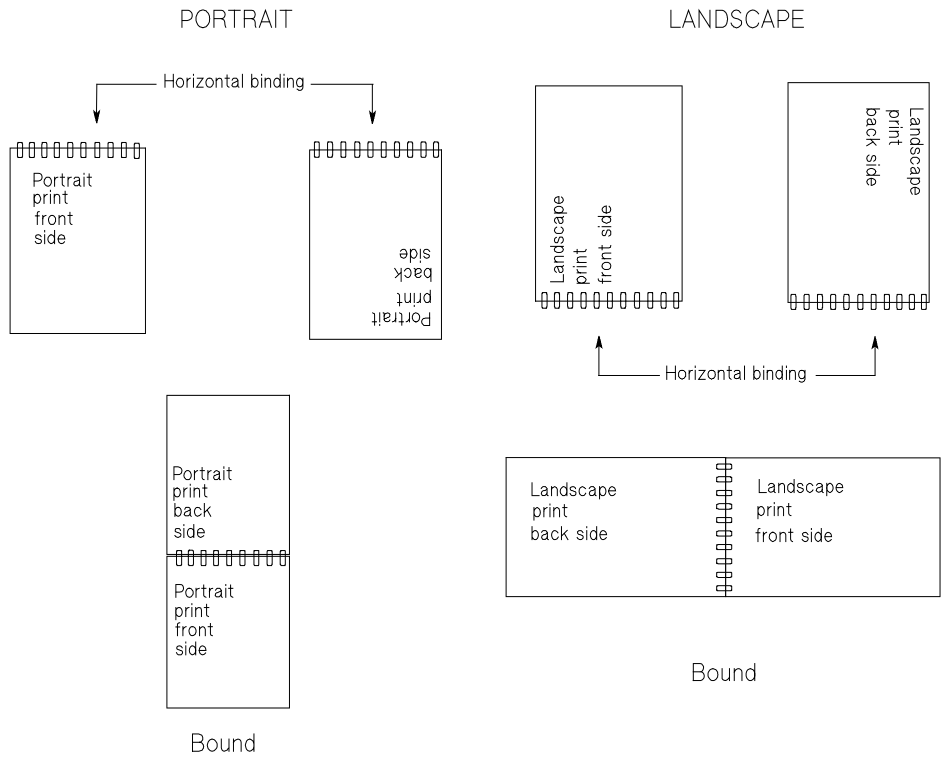

PORTRAIT
LANDSCAPE
Horizontal binding
Portrait print front side
Portrait print back side
Landscape print front side
Landscape print back side
Horizontal binding
Portrait print back side
Portrait print front side
Landscape print back side
Landscape print front side
Bound
Bound

Figure 6-1   Horizontal Binding Mode

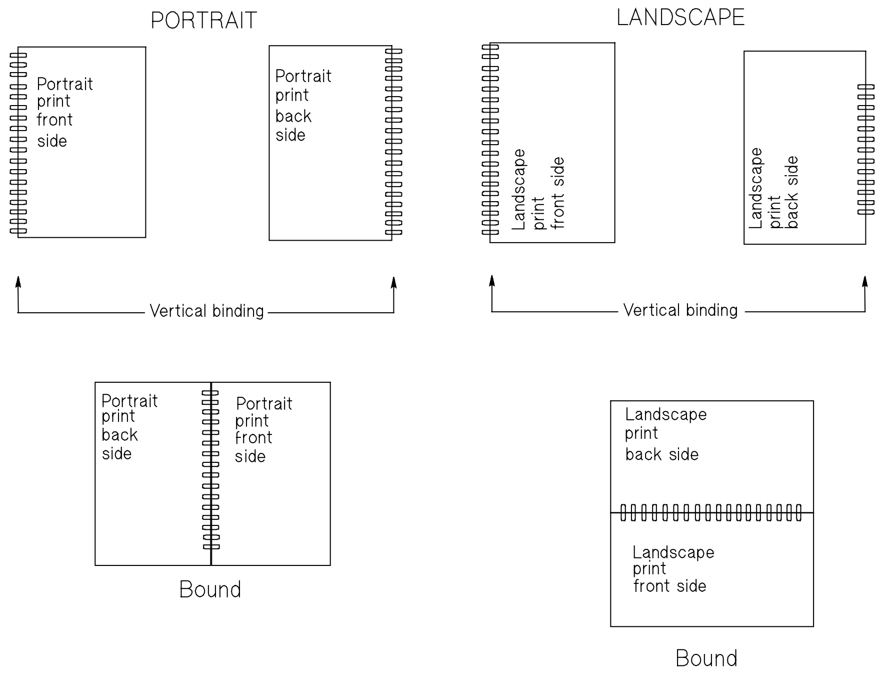

PORTRAIT
LANDSCAPE
Portrait print front side
Portrait print back side
Landscape print front side
Landscape print back side
Vertical binding
Vertical binding
Portrait print back side
Portrait print front side
Bound
Landscape print back side
Landscape print front side
Bound

Figure 6-2   Vertical Binding Mode

Left Registration *Esc & l # u/U*

Positions the logical page across the width (short side) of the physical sheet. Fractional units can be used to position to dot (device) accuracy.

Value(#)	=	Number of decipoints (1/720 inch)
Default	=	0 (the default position of the logical page)
Range	=	-2^{31} to $2^{31}-1$

Positive values move the logical page to the right along the width of the physical page, except on the back side of sheets printed in vertical binding duplex mode, where they move it to the left. Movement is independent of orientation. (As shown in figures 6-3 and 6-4, positive moves are in the direction away from the binding).

Negative values move the logical page to the left along the width of the physical page, except on the back side of sheets printed in vertical binding duplex mode, where they move it to the right. Movement is independent of orientation. (Figures 6-3 and 6-4).

NOTE: The +/- value is absolute with respect to the logical page default position, not relative to the present location.

NOTE: Registration may cause data loss by moving the logical page outside the printable area.

Top Registration *Esc & l # z/Z*

Determines the position of the logical page along the length (long side) of the physical sheet. Fractional units can be used to position to dot (device) accuracy.

Value(#)	=	Number of decipoints (1/720 inch)
Default	=	0 (the default position of the logical page)
Range	=	-2^{31} to $2^{31}-1$

Positive values move the logical page down the length of the physical page, except on the back of sheets printed in horizontal binding duplex mode, where they move it up. Movement is independent of orientation. (Figures 6-3 and 6-4).

Negative values move the logical page up the length the physical page, except on the back of sheets printed in horizontal binding duplex mode, where they move it down. Movement is independent of orientation. (Figures 6-3 and 6-4).

NOTE: The +/- value is absolute with respect to the logical page default position, not relative to the present location.

NOTE: Registration may cause data loss by moving the logical page outside the printable area.

Figure 6-3   Horizontal Binding Mode Offset

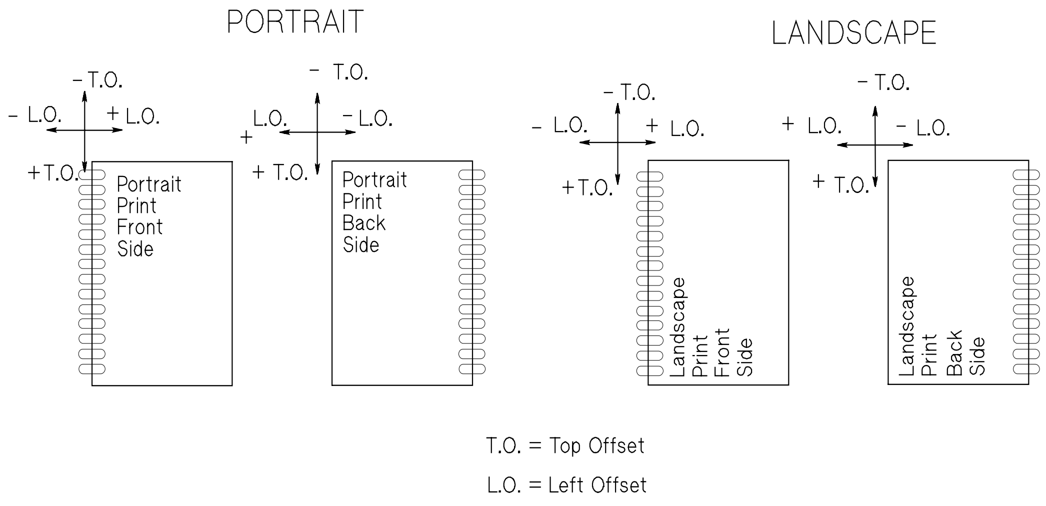


Figure 6-4   Vertical Binding Mode Offset

Copies *Esc & l # x/X*

Selects the number of printed copies of each page.

Value(#)	=	Number of copies
Default	=	1
Range	=	1 to 2^{32} -1 (negative values map to absolute values)

When this command is received anywhere on a page, it affects current and subsequent pages.

DEVICE NOTE: LJII series, LJIII, and LJIIID support 1-99. LJIIISi and LJ4 series support 1-32767.

DEVICE NOTE: Since the HP-GL/2 Replot command (RP) is inactive in dual context PCL5, *Esc&l#X* may be sent for multiple plots; it must be sent prior to closing the page on which the plot is defined.

Negative Motion *Esc & a # n/N*

Specifies whether negative motion will be used.

Value(#)	=	0	Picture contains negative motion (page formatting printers)
	=	1	Picture contains no negative motion (swath printers)
Default	=	0	
Range	=	0, 1	

Negative motion includes:

- Vertical motion toward the top of the page.
- HP-GL/2 operations.
- Print directions other than 0 degrees.
- Landscape text.
- When the top of the character cell on the next line is above the top of the character cell on the current line.

The default value of 0 delays printing until all the processing of input data for a page is complete.

A value of 1 allows data to be printed as received, rather than first stored in a buffer. Otherwise, printing will be delayed until all the processing of input data is complete.

NOTE: This command must be sent before any printable data is received.

DEVICE NOTE: Only PJ, PJXL, PJXL300, DJ1200C, and DJ1600C use this command.

Unit of Measure *Esc & u # d/D*

Establishes the size of a PCL Unit (formerly "dot").

Value (#)	=	Number of units per inch
Default	=	Device dependent
Range	=	96, 100, 120, 144, 150, 160, 180, 200, 225, 240, 288, 300, 360, 400, 450, 480, 600, 720, 800, 900, 1200, 1440, 1800, 2400, 3600, 7200

DEVICE NOTE: DJ850C implements only 300 and 600 and ignores all other values.

The value field establishes the number of units per inch used in the following commands. These commands formerly used a device's dot-per-inch resolution as the unit of measure. In fact, printer's that do not support this command still use dots as their unit of measure for these commands.

- Move CAP Vertical (PCL Units) — *Esc*p#Y*
- Move CAP Horizontal (PCL Units) — *Esc*p#X*
- Vertical Rectangle Size (PCL Units) — *Esc*c#B*
- Horizontal Rectangle Size (PCL Units) — *Esc*c#A*

In addition, since the current unit of measure setting affects how CAP movement values are rounded, it also affects the result of the following commands:

- Move CAP Horizontal (Columns) — *Esc&a#C*
- Horizontal Tab — <*HT*>
- Space — <*SP*>
- Backspace — <*BS*>

In addition, devices supporting this command — regardless of whether this command has been sent — round a selected font's CMI to the nearest PCL unit, rather than to the nearest internal unit. This may cause character spacing to be different on a 600 dpi printer vs a 300 dpi printer, or on a 300 dpi printer that supports this command vs a 300 dpi printer that does not. To get the same character placement on both printers, the unit of measure should be set to 300 and the CMI command (*Esc&k#H*) should be sent after font selection has occurred.

DEVICE NOTE: DJ850C does not round CMI to the nearest PCL Unit; so this command affects only the start of characters and PCL Unit commands.

This command does not affect binary raster data (bitmap fonts, raster graphics, patterns).

A control panel reset or *EscE* defaults the measuring unit. Since the unit of measure is part of the user environment, it is saved and restored whenever a macro is called or an overlay invoked, but is defaulted when the user environment is established for an overlay.

Bitmap and scalable fonts must retain their initial metric information. Conversion to the current selected unit must use the original units to avoid cumulative errors due to successive rounding.

Out-of-range values are mapped to the supported value with the minimum relative error. For example, the relative error of a unit selection of 4801 is closer to 7200 than 3600 — i.e., ABS (4801-7200) / 7200 < ABS (4801 - 3600) / 3600.

Peripheral Configuration *Esc & b # w/W*

Provides communication with the AppleTalk driver or the Peripheral Management Language (PML) parser.

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

DEVICE NOTE: DJ850C supports a value field range of 0 to 32767.

The data fields must contain byte-aligned binary data, not ASCII. Invalid configurations are ignored and the data discarded. Value field signs are ignored. Extra bytes are discarded.

Encapsulated PML

This command provides a way for PCL to send PML commands to devices that support "encapsulated" PML. PCL is used as a PML transport when a bidirectional link is unavailable or synchronization between the PCL data stream and the asynchronous control of PML is desired. Only the SET and DISABLE PML commands are accepted (GET and ENABLE are ignored). However, SET and DISABLE are not acknowledged. The format is:

PML<sp><PML command>

AppleTalk

This command can configure the device to receive PCL print jobs over an AppleTalk connection. AppleTalk may use this command to provide a job name, a document name, a page count, or to rename the printer for identification on the AppleTalk network.

In AppleTalk, the binary data field is in the format [key][space][value]. The *key* is a character string with a length of 1 to the device-supported length. An ASCII space character separates the key string and the value string: the first occurrence of a space signifies the end of the key. The *value* is a character string with a range of 1 to the device-supported length. Value strings are terminated by either a NULL character or expiration of binary data within the binary data field. While both the key and value strings can contain from 0 to 4,294,967,295 bytes each, their sum total must be less than the maximum allowed by the command. AppleTalk recognizes the following binary sequences:

JOB <job name>
RENAME <new name>
TYPE <type name>
ZONE <zone name>
DOCUMENT <document name>

JOB<sp>**jobname** renames the current job. All character names are valid. Only the first 127 characters (bytes) are used. There is no default job name.

RENAME<sp>**devicename** renames the name field of the printer's AppleTalk Network Identifier (Name Binding Protocol name field). The valid characters of the printer name include 0-255 except for the characters NULL <00H>, "@" <40H>, ":" <3AH>, "*" <2AH>, "=" <3DH>, "≈" <C5H>, and <FFH>. If the printer encounters a NULL <00H> in the device name, it uses the NULL character to terminate the device name. All the characters preceding the NULL will be used as the device name. The printer checks for invalid characters and ignores the escape sequence if an invalid character is encountered. A name must contain at least one character, and only the first 31 characters are used. The printer's name is not changed if the name is invalid.

TYPE<sp>**devicetype** renames the type field of the printer's AppleTalk Network Identifier (Name Binding Protocol type field). The valid characters for type include 0-255 except for the characters (from the Apple Character Set): NULL <00H>, "@" <40H>, ":" <3AH>, "*" <2AH>, "=" <3DH>, "≈" <C5H>, and <FFH>. If the printer encounters a NULL <00H> in the devicetype, it uses the NULL character to terminate the device type. All the characters preceding the NULL will be used as the device type. The printer checks for invalid characters and ignores the escape sequence if an invalid character is encountered. A type must contain at least one character, and only the first 31 characters are used. The printer's type is not changed if the type is invalid. Type changes occur only after the present job is completed.

ZONE<sp>**zonename** changes the zone field of the printer's AppleTalk Network Identifier (Name Binding Protocol zone field). The valid characters for the zone include 0-255 except for the characters NULL <00H>, "@" <40H>, ":" <3AH>, "*" <2AH>, "=" <3DH>, "≈" <C5H>, and <FFH>. If the printer encounters a NULL <00H> in the zone name, it uses the NULL character to terminate the zone name. All the characters preceding the NULL will be used as the zone name. The printer checks for invalid characters and ignores the escape sequence if an invalid character is encountered. The zone must contain at least one character, and only the first 31 characters are used. The printer's zone is not changed if the zone is invalid. Zone changes occur only after the present job is completed.

DOCUMENT<sp>**documentname** renames the current document. The valid characters for the document name include 0-255 except for the characters NULL <00H> and ":" <3AH>. If the printer encounters a NULL <00H> in the document name, it uses the NULL character to terminate the document name. All the characters preceding the NULL will be used as the document name. A document name must contain at least one character, and only the first 31 characters are used.

NOTE: AppleTalk commands are not used in DOS.

6.2 Printer Diagnostics

Self-test *Esc z*

A self-test performs the following actions:

- Processes all data preceding the self-test.
- Performs a full Reset (*EscE*). The page is ejected if printable data has been sent. Downloaded fonts are not deleted.
- Moves CAP to the top of form, if not already there.
- Performs the self-test.
- After self-test, moves CAP to the top of form, if not already there.
- Resumes execution without data loss (programmable features need not be saved).
- Prints results.

If no error is detected, the printer should remain on-line. If an error is detected, the printer should go to the off-line state.

DEVICE NOTE: HP5000 fanfold printers do not perform a full reset in a self-test.

DEVICE NOTE: LJ5P initiates either self-test or PCL fontlist, depending on a PJL environmental variable.

6.3 Device Specific Control

Gray Balance *Esc * b # b/B*

Determines whether black optimization is performed to make process black (composed of color inks) appear more black.

Value(#)	=	1	Enable gray balancing
	=	2	Disable gray balancing
Default		=	Device-dependent
Range		=	1,2

When black is constructed by setting all the bits in the CMY color planes, the resulting color may have a non-black hue.

Dry Timer *Esc & b # t/T*

Sets a minimum time between pages to ensure that a previous page dries before the next page is dropped on top of it.

Value(#)	=	Dry time in seconds	
Default		=	Device-dependent
Range		=	0 to 1200

The drying time for ink depends on media, print modes, and environment. For example, transparencies may need over 10 minutes in an unheated environment.

Color Raster Graphics Depletion *Esc * o # d/D*

Specifies the depletion method for raster image printing.

Value(#)	=	1	No depletion
	=	2	25% depletion of color data
	=	3	50% depletion of color data
	=	4	25% depletion of color data with color linearity compensation
	=	5	50% depletion of color data with color linearity compensation
Default		=	Device-dependent
Range		=	1 to 5

Graphics depletion selectively removes pixels from a graphics image to reduce the amount of ink placed on the page. Depletion will alter the color appearance.

Values 4 and 5 perform linear compensation by selectively depleting certain color intensities differently than other intensities for a more linear color intensity curve. In general, the gamma-corrected depletions result in better color output.

DEVICE NOTE: DJ550C does not deplete single-plane raster images or foreground color. Only the color planes (CMY) are depleted on 4-plane raster images (CMYK).

DEVICE NOTE: DJ850C maintains the proper hue and implements 4 identical to 2 and 5 identical to 3.

Mechanical Print Quality *Esc * o # q/Q*

Determines print quality for graphics on paper.

Value(#)	=	-1	Maps to 1
	=	0	No shingling
	=	1	50% shingling
	=	2	25& shingling
	=	3	25% shingling for 600 x 300 dpi data
Default	=	0	
Range	=	-1 to 3	

"Shingling" is an interlace technique with checkerboard masking to remove "banding" effects.

DEVICE NOTE: This command is interpreted this way for DJs below 1200; it is interpreted as shown below for PJXL300, DJ1200C, and DJ1600C.

Mechanical Print Quality *Esc * o # q/Q*

Determines print quality for graphics on paper.

Value(#)	=	-1	EconoFast
	=	0	Normal quality
	=	1	Presentation
Default	=	0	
Range	=	-1 to 2	

DEVICE NOTE: This command is interpreted this way for PJXL300, DJ1200C, and DJ1600C; it is interpreted as shown above for DJs below 1200.

Print Quality *Esc * o # m/M*

Specifies the desired print quality for a page.

Value(#)	= -1 Draft, Economy, or EconoFast mode
= 0	Normal quality
= 1	Presentation quality
Default	= 0
Range	= -1,0,1

The functional means of achieving the requested quality level are device specific.

If no printable data has been sent, CAP moves to the top of form at the left margin of the current page; if printable data has been sent, the current page is printed and CAP moves to the top of form at the left margin of the next physical page.

This command, in conjunction with the Media Type command (*Esc&l#M*), provides a high-level print mode specifier and replaces the need for the following:

Font Quality (Primary)	*Esc(s#Q*
Font Quality (Secondary)	*Esc)s#Q*
Raster Graphics Quality	*Esc*r#Q*
Mechanical Print Quality	*Esc*o#Q*

DEVICE NOTE: After this command, DJ660C and 850C ignore the Primary and Secondary Font Quality, Raster Graphics Quality, and Mechanical Print Quality commands until a reset.

Media Type *Esc & l # m/M*

Sets the print mode required for printing on various media types.

Value(#)	=	0	Plain paper
	=	1	Bond Paper
	=	2	Special paper
	=	3	Glossy film
	=	4	Transparency film
Default		=	0
Range		=	0 to 4

If no printable data has been sent, CAP moves to the top of form at the left margin of the current page. If printable data has been sent, the page is printed and CAP moves to the top of form at the left margin of the next physical page.

6.4 Internal HP Sequences

NOTE: Do not document the following escape sequences outside HP.

The following sequences should not appear in any user or ISV documentation.

Designate Primary Font *Esc (# @*

Designates the font or symbol set used for the primary font. (This command is used for testing. It should not appear in user or ISV documentation.)

Value(#)	=	0,1	Designates the symbol set of the default primary font.
	=	2	Designates the most recently requested symbol set for the primary font.
	=	3	Designates all default primary font attributes.
Default		=	NA
Range		=	0 to 3

A value field of 2 sets the symbol set of the primary font to the most recently requested symbol set (this may not be the symbol set of the current font). A value field of 3 sets all primary font attributes except orientation to those of the default font. Pitch is not changed if the font is proportional.

NOTE: This command is used for testing. It should not appear in user or ISV documentation.

Designate Secondary Font *Esc) # @*

Designates the font or symbol set to be used for the secondary font. (This command is used for testing. It should not appear in user or ISV documentation.)

Value(#)	=	0	Designates the symbol set of the default secondary font.
	=	1	Designates the symbol set of the default primary font.
	=	2	Designates the most recently requested symbol set for the secondary font.
	=	3	Designates default secondary font attributes except orientation.
Default		=	NA
Range		=	0 to 3

A value field of 2 sets the symbol set of the secondary font to the most recently requested symbol set (this may not be the symbol set of the current font). A value field of 3 sets all secondary font attributes except orientation to those of the default font. Pitch is not changed if the font is proportional.

NOTE: This command is used for testing. It should not appear in user or ISV documentation.

Driver Function Configuration *Esc * o # w/W [data]*

Allows drivers to configure parameters that might change after the printer is shipped.

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

To avoid proliferation of undocumented escape sequences, this sequence can perform multiple functions by encapsulating a block of parameters in a binary transfer. If the printer recognizes this sequence, the first byte of data is a format number. The printer ignores the sequence if it does not recognize the sequence or format number.

Byte 1:	Format number (key)
Bytes 2-n:	Function dependent

For format 1 to 3, the format of the function dependent bytes is:

Byte 1:	Format number
Byte 2:	Function index
Byte 3:	Sign of parameter: either "+" (2BH), "-" (2DH), or Null (00H)
Bytes 4-5:	Function dependent parameter (unsigned 16-bit integer: byte 4 is the most significant and byte 5 is the least significant)

DEVICE NOTE: Format 1 is understood by DJ500C; Format 2 by DJ500, 510, and 550C; Format 3 by DJ520 and 560C; Format 4 by DJ540 and 660C; Format 5 by OfficeJet; Format 6 by DJ310 and Color LJ; Format 7 by DJ850C. Commands of a lower format number may be understood by a printer from the same class that understands a higher format number, i.e., some of the Format 1 and 2 commands are understood by DJ520 and 560C.

DEVICE NOTE: As described below, Color LaserJet uses this sequence to specify lightness, saturation, under-color removal, scaling algorithm, color mode, and to download a color map specification.

Driver Configuration *Esc*o#W[device_id function_index arguments]*

Specifies lightness, saturation, under-color removal, scaling algorithm, color mode, and to download a color map specification.

device_id (key)	=	6	Specifies device as Color LaserJet		
function_index	=	0	Lightness - argument range [-100..100]		
		1	Saturation - argument range [-100..100]		
		2	Under-color removal - argument range [shadow, highlight, gamma] 0-255, unsigned bytes		
		3	Scaling algorithm - argument [	0	Pixel replication
				1	Bilinear interpolation
				2	Modified bilinear interpolation]
		4	Select color mode - argument [	0	No adjustment
				1	Business blue
				2	Transparency
				3	Vivid graphics
				4	Out of gamut (DIC)
				5	CIE Lab match (DIC)]
		5	Download color map - Color space [	1	CMY
				3	CIELab]
			Mapid [see valid Mapid list]		
			Data [14739 bytes]		

Lightness: Negative values darken (unlighten) color images, text or graphics, but have no effect on black or white data. Positive values lighten the image. Zero turns lightness adjustment off. This replaces *Esc*u#L*. This function index requires three bytes of data.

Saturation: Negative values desaturate (gray) color images, text or graphics, but have no effect on black or white data. Positive values increase saturation, making the image more vivid. Zero turns saturation adjustment off. This replaces *Esc*u#S*. This function index requires three bytes of data.

Under-Color Removal: Three values are specified: Shadow, Highlight, and Gamma. This function index requires five bytes of data:

- Shadow: unsigned byte, range [0..255]
- Highlight: unsigned byte, range [0..255]
- Gamma: ubytes in unsigned byte, range [0..255], internally scaled by 10 such that internal representation is [0..255]

Scaling: The backward compatible scaling algorithm is pixel replication. Bilinear Interpolation is a high quality scaling algorithm for smooth edge interpolated scaling. Modified Bilinear only interpolates when it is best to do so. This function index requires three bytes of data.

Select Color Mode: Specifies the color treatment mode for rendering the ensuing job.

- No adjustment:
  1. No color correction: the user sees pure engine behavior.

     + HP stays out of the way.
     - Non-linear response/output to linear color input. (color gradients/ramps show large hue shifts.)
     - Halftone dependent: hues vary according to the halftone chosen.

  2. Linearization only: the user sees the device as a linear device.

     + Non-halftone dependent: linearization for every supported halftone.
     + 3-space linear response to numerically linear input (i.e., smooth gradients that are independent of hue).
     - Output is not as vivid/saturated as users may prefer.
     - HP has calibrated.

  3. Same as Vivid Graphics: no adjustment #2, with the addition of user-preferred enhancements, like a boost in saturation/vividness.

- Business Blue: Same as Vivid Graphics (linearization plus user-preferred enhancements with the addition of mapping process blue (visually printed as purple) to a blue closer to a standard monitor.

- Transparency: Uses a map to render the best color output on transmissive media.

- Vivid Graphics: Produces a more saturated rendition of the input image.

- Out of Gamut: Prints colors that are out of gamut: all colors that are in gamut snap to white.

- CIE Lab Match: This map performs a true color match to requested CIE Lab input: there are no user-preferred enhancements.

NOTE: For screen matching, the long form of the CID command is used and the color maps are generated internally, dependent on the monitor calibration data: i.e., this command is not needed.

Download Color Map: The printer supports the downloading of color adjustment maps according to the halftone requested, the type of color treatment desired (including device-dependent or independent), and the type of media. This is to ensure the ability to provide the best output in the future as more is known about the stability/operation point of engines and customer preference.

Valid Mapid List

Description	
Cluster - No Adjust - LinearCMY	
Disperse - No Adjust - LinearCMY	
Scatter - No Adjust - LinearCMY	
ErrorDiffusion - No Adjust - LinearCMY	
Cluster - Business Blue - LinearCMY	
Disperse - Business Blue - LinearCMY	
Scatter - Business Blue - LinearCMY	
ErrorDiffusion - Business Blue - LinearCMY	
Cluster - Transparency - LinearCMY	
Disperse - Transparency - LinearCMY	
Scatter - Transparency - LinearCMY	
ErrorDiffusion - Transparency - LinearCMY	
Cluster - VividGraphics - LinearCMY	
Disperse - VividGraphics - LinearCMY	
Scatter - VividGraphics - LinearCMY	
ErrorDiffusion - VividGraphics - LinearCMY	
Cluster - OutOfGamut - LinearCMY	
Disperse - OutOfGamut - LinearCMY	
Scatter - OutOfGamut - LinearCMY	
ErrorDiffusion - OutOfGamut - LinearCMY	
Cluster - TrueMatch - LinearCMY	
Disperse - TrueMatch - LinearCMY	
Scatter - TrueMatch - LinearCMY	
ErrorDiffusion - TrueMatch - LinearCMY	
Cluster - No Adjust - LabToCMY	
Disperse - No Adjust - LabToCMY	
Scatter - No Adjust - LabToCMY	

ErrorDiffusion - No Adjust - LabToCMY	
Cluster - Business Blue - LabToCMY	
Disperse - Business Blue - LabToCMY	
Scatter - Business Blue - LabToCMY	
ErrorDiffusion - Business Blue - LabToCMY	
Cluster - Transparency - LabToCMY	
Disperse - Transparency - LabToCMY	
Scatter - Transparency - LabToCMY	
ErrorDiffusion - Transparency - LabToCMY	
Cluster - VividGraphics - LabToCMY	
Disperse - VividGraphics - LabToCMY	
Scatter - VividGraphics - LabToCMY	
ErrorDiffusion - VividGraphics - LabToCMY	
Cluster - OutOfGamut - LabToCMY	
Disperse - OutOfGamut - LabToCMY	
Scatter - OutOfGamut - LabToCMY	
ErrorDiffusion - OutOfGamut - LabToCMY	
Cluster - TrueMatch - LabToCMY	
Disperse - TrueMatch - LabToCMY	
Scatter - TrueMatch - LabToCMY	
ErrorDiffusion - TrueMatch - LabToCMY	

Underware Function Configuration *Esc & i # w/W [data]*

Allows access to underware functions for testing.

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

To avoid proliferation of undocumented escape sequences, this sequence can perform multiple functions by encapsulating a block of parameters in a binary transfer. If the printer recognizes this sequence, the first byte of data is a format number. The printer ignores the sequence if it does not recognize the sequence or format number. The format and usage of the binary data is device-dependent. One example might be the sending of a *key* (composed of several digits) that switches to a specialized testing language. The sequence can be used on multiple products just by changing the key, which is unlikely to be discovered by accident.

Byte 1:	Format number (key)
Bytes 2-n:	Function dependent

For format 1 to 3, the format of the function-dependent bytes is:

Byte 1:	Format number
Byte 2:	Function index
Byte 3:	Sign of parameter: either "+" (2BH), "-" (2DH), or Null (00H)
Bytes 4-5:	Function dependent parameter (unsigned integer: byte 4 is the most significant)

DEVICE NOTE: Format 0 is understood by PJXL300, DJ1200C, and DJ1600C; Format 1 by DJ500C; Format 2 by DJ510 and 550C; Format 3 by DJ520 and 560C; Format 4 by DJ540 and 660C; Format 5 by OfficeJet; Format 6 by Color LaserJet; Format 7 by DJ850C.

Chapter 7: Page Control

Contents of this Chapter

This chapter describes commands which affect a single page or a group of pages (as opposed to job control commands, which are issued at the beginning of the job and affect the entire job).

This chapter describes the following PCL commands:

7.1 Introduction

Job control commands are usually sent at print job boundaries, but page control commands are associated with a single page or group of pages. If consecutive pages have the same format, the appropriate page control commands need only be sent once for that group of pages. Page control commands determine such features as media source, size, orientation, margins, and text spacing.

7.2 Media Size

The default logical page length extends from the top edge of the physical page to the bottom edge. The width is device dependent and may not extend as far as the unprintable area on the sides.

Page Length (*Esc&l#P*) sets the logical page length for a given LMI. Page Size (*Esc&l#A*), which selects the physical page size from one of several pre-set sizes, indirectly sets logical page length (which is coincident with physical page length).

For comparison, some DeskJet 540 printable areas for standard media sizes in portrait orientation are shown below. The right edge of the printable area can be found if the left edge is known: the right edge equals the physical media width minus the left edge. On the DeskJet 540, logical page width is coincident with the sides of the printable area, but this is not the case on all devices.

Supported Media Size	**Printable Area**	**Unprintable Area**	
		LeftEdge	**TopEdge**
US Executive 7¼” × 10½” 184.1 mm × 266.7 mm	6.750” x 10.040” 171.5 mm x 255.0 mm	0.250” 6.3 mm	0” 0 mm
US Letter 8½” × 11” 215.9 mm × 279.4	8.00” x 10.540” 203.2 mm x 267.7	0.	0” 0

mm	mm	250” 6.3mm	mm
ISO & JIS A4 8.268” x 11.693” 210 mm x 297 mm	8.000” x 11.232” 203.2 mm x 285.3 mm	0.133” 3.4mm	0” 0mm
ISO & JIS A5 5.827” x 8.268” 148 mm x 210 mm	5.575” x 7.559” 141.6 mm x 192 mm	0.125” 3.2mm	0” 0mm
JIS B5 7.165” x 10.118” 182 mm x 257 mm	6.832” x 9.658” 173.6 mm x 245.3 mm	0.167” 4.2mm	0” 0mm
US Card Stock 4” x 6” 101.6 mm x 152.4 mm	3.750” x 5.291” 95.2 mm x 134.4 mm	0.125” 3	0” 0mm

		.2mm	
No.10 Envelopes 4 1/8” x 9 1/2” 104.8 mm x 241.3 mm	3.875” x 8.791” 98.4 mm x 223.3 mm	0.125” 3.2mm	0” 0mm

Page Length *Esc & l # p/P*

Designates the length of the logical page for a given LMI.

Value(#)	=	Number of lines at a given LMI
Default	=	Device dependent
Range	=	0 to the maximum supported paper size (other values ignored and current size retained)*****

* A value of 0 should set page length to the operator setting if possible; otherwise, to the physical page page length if it can be sensed; otherwise, to 11 inches.

This command performs the following actions:

- Prints any unprinted pages.
- Ejects the current page if printable data has been received (FF-CR).
- Sets text length, top margin, left margin, and right margin to user defaults.
- Moves CAP to the left edge of the logical page at the top margin and floats CAP.
- Disables the automatic macro overlay.
- Defaults the picture frame anchor point.
- Defaults the picture frame.
- Defaults the PCL print direction.
- Defaults the HP-GL/2 plot size.
- Defaults the soft clip window.
- Defaults the HP-GL/2 scaling points, P1 and P2.
- Updates the HP-GL/2 pen position to the lower-left corner of the picture frame.
- Clears the HP-GL/2 polygon buffer.
- Performs an HP-GL/2 *IN* command.

The printer may select a different page size for the same line count, since lines are defined by the current LMI. This command is ignored if LMI is 0.

The following table lists the page length line values associated with some standard paper sizes. To calculate the number of lines per page, multiply lines per inch (lpi) times the length of the physical page. For example, US Letter size paper is 11 inches; therefore: $6 \times 11 = 66$.

Portrait Orientation Page Length Settings

Paper Size	**6 lpi**	**8 lpi**
Letter	66	88
Legal	84	112
A4	70	93
Executive	63	84
Ledger	102	136
A3	99	132

DEVICE NOTE: The Page Length command must be used with pre-LJII's except LJ2000 to set page size. LJII and later and the LJ2000 may use either Page Length (*Esc&l#P*) or Page Size (*Esc&l#A*). Page Size is preferred.

DEVICE NOTE: Default length may be switch-selectable. Common lengths are 8.5, 11, 12.

Page Size   *Esc & l # a/A*

Designates the size of the media as one of those listed below.

Value(#)	=	Page size (listed below)
Default	=	Device dependent
Range	=	Listed below (unsupported values are ignored)

This command performs the following actions:

- Prints any unprinted pages.
- Ejects the current page if printable data has been received (FF-CR).
- Sets text length, top margin, left margin, and right margin to user defaults.
- Moves CAP to the left edge of the logical page at the top margin.
- Disables the automatic macro overlay.
- Defaults the picture frame anchor point.
- Defaults the picture frame.
- Defaults the PCL print direction.
- Defaults the HP-GL/2 plot size.
- Defaults the soft clip window.
- Defaults the HP-GL/2 scaling points, P1 and P2.
- Updates the HP-GL/2 pen position to the lower-left corner of the picture frame.
- Clears the HP-GL/2 polygon buffer.
- Performs an HP-GL/2 *IN* command.

NOTE: A minus sign distinguishes two paper mounting positions with the same paper size. Unsigned or positive selects lengthwise; negative selects widthwise. Devices supporting one mounting position ignore the sign.

This command also sets logical page length, since the logical page is defined to extend from the top and bottom edges of the physical page.

The following table lists the current paper sizes selectable by this command.

Value	Page Description	Page Size
1	US-Executive	7.25" x 10.5"
2	US-Letter	8.5" x 11"
3	US-Legal	8.5" x 14"
4	US-EDP	11" x 14"
5	European-EDP	12" x 14"
6	Ledger	11" x 17"
7	US Government Letter	8" x 10"
8	US Government Legal	8" x 13"
9	Folio	8.3" x 13"
10	Foolscap	8.5" x 13"
11	Ledger	11" x 17"
12	C Size	17" x 22"
13	D Size	22" x 34"
14	E Size	34" x 44"
15	Mini (US-Personal)	5.5" x 8.5"
20	ISO and JIS A10	26mm x 37mm
21	ISO and JIS A9	37mm x 52mm
22	ISO and JIS A8	52mm x 74mm

Value	Page Description	Page Size
23	ISO and JIS A7	74mm x 105mm
24	ISO and JIS A6	105mm x 148mm
25	ISO and JIS A5	148mm x 210mm
26	ISO and JIS A4	210mm x 297mm
27	ISO and JIS A3	297mm x 420mm
28	ISO and JIS A2	420mm x 594mm
29	ISO and JIS A1	594mm x 841mm
30	ISO and JIS A0	841mm x 1189mm
31	ISO and JIS 2A0	1189mm x 1682mm
32	ISO and JIS 4A0	1682mm x 2378mm
40	JIS B10	32mm x 45mm
41	JIS B9	45mm x 64mm
42	JIS B8	64mm x 91mm
43	JIS B7	91mm x 128mm
44	JIS B6	128mm x 182mm
45	JIS B5	182mm x 257mm
46	JIS B4	257mm x 364mm
47	JIS B3	364 x 515mm
48	JIS B2	515mm x 728mm
49	JIS B1	728mm x 1030mm
50	JIS B0	1030mm x 1456mm
60	ISO B10	31mm x 44mm
61	ISO B9	44mm x 62mm
62	ISO B8	62mm x 88mm
63	ISO B7	88mm x 125mm
64	ISO B6	125mm x 176mm
65	ISO B5	176mm x 250mm
66	ISO B4	250mm x 353mm
67	ISO B3	353mm x 500mm
68	ISO B2	500mm x 707mm
69	ISO B1	707mm x 1000mm
70	ISO B0	1000mm x 1414mm
71	Hagaki (Japanese single-size postcard)	100mm x148mm
72	Oufuku-hagaki (round-trip postcard)	148mm x 200mm
73	Same as 24 but card, not cutsheet	105mm x 148mm
74	Index card 4x6	4” x 6”
75	Index card 5x8	5” x 8”

Value	Envelope Description	Envelope Size
-90	International DL (landscape)	110mm x 220mm
-81	Number 10 (landscape)	4 1/8" x 9 1/2"
80	Monarch (Letter)	3 7/8" x 7 1/2"
81	Commercial-10 (Business)	4 1/8" x 9 1/2"
82	Catalog 1	6" x 9"
90	International DL	110mm x 220mm
91	International C5	162mm x 229mm
92	International C6	114mm x 162mm

93	International C4	229mm x 324mm
100	International B5	176mm x 250mm
101	Custom	Device specific
102	Commercial-9	98mm x 225mm
103	Commercial-11	114mm x 264mm
104	Letter Envelope	8.5” x 11”
105	US C5	6.5” x 9.5”
106	Postfix (Italian)	114mm x 229mm
107	Custom Envelope	Device specific
108	Custom Card	Device specific
109	Standard US Invitation (5.5 Baronial)	4 3/8" x 5 3/4"
110	Japanese Long Envelope #3	120mm x 235mm
111	Japanese Long Envelope #4	90mm x 205mm

DEVICE NOTE: Pre-LJIIs ignore this command and use Page Length (*Esc&l#P*). LaserJets set logical page width to (physical width - 1/2") for US portrait sizes, to (physical width - 12mm) for European portrait sizes, to (physical length - 2/5") for US landscape sizes, and to (physical length - 10 mm) for European landscape sizes.

DEVICE NOTE: Since DJ Classic only loaded envelopes widthwise (normally specified by -81), it ignored the sign. Later DJs maintained compatibility by switching to landscape for a -81 and not changing the current orientation for an 81. DJ850C saves the current orientation when it receives an 81, excutes the command, and then restores the saved orientation when it receives a Reset, Orientation, or Page Size command specifying anything but 81. DJs below 850 change but do not save orientation.

DEVICE NOTE: “Custom” (101) allows ranges to the largest supported paper size. Custom is treated like an envelope to prompt a manual feed. LJ4+ uses (3" x 5" to 8.5" x 14") and DJ540, 660, 850 use (5" x 8.5" to 5.83" x 14"). LJ4V uses 3.9" x 5.8" to 11.7"x 17.7".

DEVICE NOTE: Color LJ supports only the value of 6 to select ledger (11x17). LJ4 and Eclipse support both 6 and 11.

Media Eject Length   *Esc & f # f/F*

Sets the length of paper to be "ejected" when a page eject is requested on variable length media devices (e.g., fanfold printers and roll feed plotters).

Value(#)	=	Media eject length in decipoints (1/720 inch)
Default	=	Device dependent (should be taken from User Default Environment)
Range	=	0 to the maximum supported paper length (command is ignored for other values and the current paper length is retained)

This command performs the following actions:

- If printable data has been received, it executes a FF-CR using the old media eject length.
- Sets the new media eject length.
- Sets text length, top margin, left margin, and right margin to user defaults.
- Moves CAP to the top of form at the left edge of the logical page if not already there.
- Disables the automatic macro overlay.

This command is used when the page eject length does not equal the physical page size defined by the Page Size (*Esc&l#A*) and Page Length (*Esc&l#P*) commands because of media or equipment constraints.

The media eject length need not be the same as the physical page size defined by Page Size (*Esc&l#A*) or Page Length (*Esc&l#P*). If the media eject length is shorter than the physical page size, the physical page image will be clipped to fit the media eject size. If the media eject length is longer than the physical page size, the excess length will appear after the page image (to the bottom and right of the page image so the page image is placed at the upper left edge).

DEVICE NOTE: The default media eject length on HP5000 fanfold printers is the control panel setting for page length.

Page Width *Esc & f # g/G*

Designates the width of the logical page for a given CMI.

Value(#)	=	Logical page width in decipoints (1/720 inch)
Default	=	Device dependent
Range	=	0 to the maximum supported page width (command is ignored for other values and the current paper length is retained) *

* If a value of 0 is received, page width should be set to the operator setting if applicable; otherwise, the default page width should be based on the physical page width if it can be sensed; otherwise to 8.5 inches portrait, 11 inches landscape orientation.

This command performs the following actions:

- Executes FF-CR on the current page if printable data has been received.
- Sets text length, top margin, left margin, and right margin to user defaults.
- Moves CAP to the top of form at the left edge of the logical page if not already there.
- Disables the automatic macro overlay.

The page width remains in effect until a new Page Width command is received or a power cycle occurs.

NOTE: This command should be used in conjunction with the Media Eject Length command (*Esc&f#F*) to fully define a variable logical page size on hardcopy devices that support setting variable page sizes. Both the Media Eject Length and Page Width commands should be transmitted at the beginning of a page prior to any printable data; otherwise, when the command is sent the current page is closed and printed.

7.3 Media Source

Media Source *Esc & l # h/H*

Selects the media source.

Value(#)	=	-2	Preload (a sheet is loaded prior to printable data)
	=	-1	Tractor feed
	=	0	Print current page (source is unchanged)
	=	1	Tray 1 (usually the upper tray in a 2-tray system)
	=	2	Manual feed
	=	3	Manual envelope feed
	=	4	Tray 2 (usually the lower tray in a 2-tray system)
	=	5	Optional media source
	=	6	Optional envelope feeder *
	=	7	Autoselect
Default	=		Device dependent
Range	=		-2 to 7 (non-existent selections cause a default)

* Must be used in conjunction with an envelope selection using the *Page Size* command.

This command prints the current page and moves CAP to top of form at the left margin on the next physical page. The command remains effective until another tray is selected.

A value of 7 (autoselect) selects the current printer default source. The user, through the application, may select a particular tray for the first page or pages (e.g., a fancy cover page), then choose autoselect to pull paper from a default tray (e.g., containing standard paper). This is different than option 0, which continues printing from the currently selected source. The "default" source may be user-selected, or based upon the printer's own algorithm. The autoselect option allows the Windows driver to select the default media source at any time during printing by sending an *Esc&l7H*. The driver does not need to know which tray is loaded; it simply selects the default source, which is determined by a user setting or the printer's internal algorithm.

DEVICE NOTE: In a multibin LaserJet, *Page Size* has priority over *Media Source*: if paper is requested from a location containing an incorrect paper size, the printer selects the paper location with the currently selected size if one is available. The default paper source is determined by the default page size.

DEVICE NOTE: On the Color LaserJet, the following values mean:

1	=	Feed from main front cassette (Tray #1)
2	=	Feed from manual top rear input (Tray #2)
4	=	Feed from optional cassette or alternate feed unit cassett at the top rear
5	=	Feed from large media source (replaceable main cassette)

DEVICE NOTE: For HP5000 cutsheet printers, tray #1 is the tray with the largest paper capacity; and tray#2 is the tray with the second largest paper capacity.

7.4 Orientation

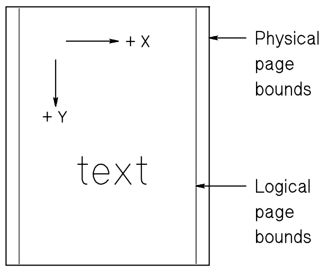
Orientation defines the position of the logical page on the physical page. *Portrait* orientation means (0,0) is toward the top left corner of the physical page: positive X direction is to the right and positive Y direction downward. In *reverse portrait* orientation, (0,0) is toward the lower right corner of the physical page: positive X direction is to the left and positive Y direction upward.

Landscape orientation means (0,0) is toward the lower left corner of the physical page with the X direction up and the Y direction to the right. In *reverse landscape* orientation, (0,0) is toward the upper right corner of the physical page with the X direction downward and the Y direction toward the left.

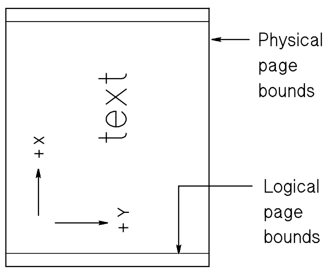
Logical page orientation changes text orientation, not raster graphics orientation; for example, raster graphics will continue to print in portrait if orientation is changed from portrait to landscape. Fonts automatically rotate to the current orientation on devices having auto-rotation.

Although orientation may be changed by either the Orientation command or the Print Direction command, these two commands have different side-effects:

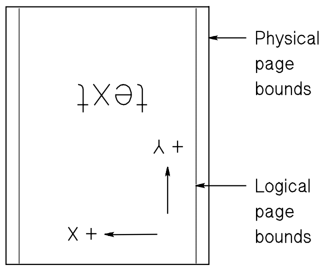
- Orientation ejects the page and Print Direction does not; therefore, Print Direction can be used to make orientation changes on the same page.
- Print Direction does not affect HP-GL/2 graphics unless HP-GL/2 is entered with *Esc%2B* or *Esc%3B.* HP-GL/2 graphics can always be rotated by the Orientation or "RO" commands.



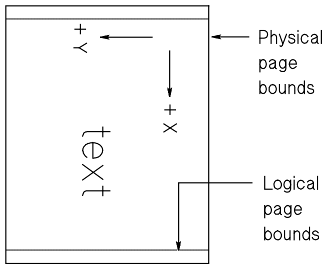
Portrait Orientation



Landscape Orientation



Reverse Portrait Orientation



Reverse Landscape Orientation

HP-GL/2 Orientation

Default HP-GL/2 orientation tracks the Orientation command (*Esc&l#O*). If HP-GL/2 was entered by *Esc%2B* or *Esc%3B*, HP-GL/2 orientation also tracks Print Direction (*Esc&a#P*).

In both coordinate systems, X-axes are parallel and increase in the same direction; Y-axes are parallel and increase in opposite directions. This remains true even if logical page orientation changes.

For more information, see Chapter 17.

How HP-GL/2 Orientation Tracks Logical Page Orientation

Orientation *Esc & l # o/O*

Defines the position of the logical page and the default direction of print with respect to the physical page.

Value(#)	=	0	Portrait
	=	1	Landscape
	=	2	Reverse portrait
	=	3	Reverse landscape
Default		=	0
Range		=	0 to 3

This command ejects the current page if it contains printable data and opens another page in the new orientation. CAP moves to the intersection of the left margin and top of form (top margin plus 3/4 line spacing). The command is ignored if the new orientation and the current orientation are the same.

This command has the following effects:

- Prints all data received before the command
- Executes a formfeed and carriage return.
- Sets the following to their user defaults:
  - Logical page
  - Print direction
  - Page length
  - Text length
  - Top margin, left margin, right margin
  - CMI and LMI
  - Picture frame
  - Picture frame anchor point
  - HP-GL/2 plot size
  - HP-GL/2 orientation
- Disables the auto macro overlay

This command should be sent at the beginning of a page because it ejects a page containing printable data. Since this command defaults the above features, it should be followed by commands that set any desired non-default values.

NOTE: Because it ejects the page, this command cannot be used to change text orientation within a page. Print Direction (*Esc&a#P*) can be used to print multiple directions per page.

This command affects text orientation, not raster graphics orientation. For example, if orientation is changed from portrait to landscape, raster graphics will continue to print in portrait. This can cause clipping if graphics margins are insufficient.

Fonts automatically rotate to the current orientation on devices having auto-rotation.

DEVICE NOTE: Pre-LJIIIs did not rotate fonts. Orientation was a selection criterion.

DEVICE NOTE: Pre-DJ550s except 540 lock out graphics data in landscape orientation.

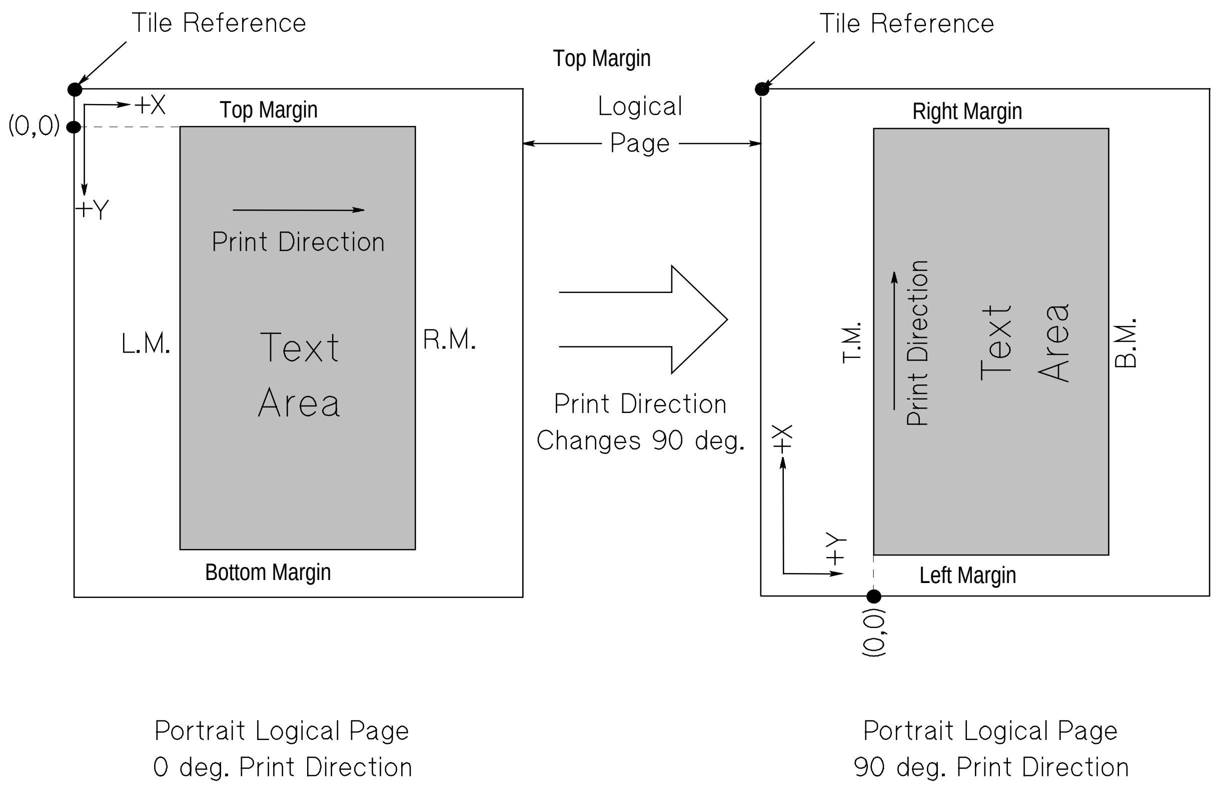
Print Direction *Esc & a # p/P*

Rotates the logical page coordinate system with respect to its current orientation without ejecting the page. This allows orientation changes on the same page.

Value(#)	=	Degrees of rotation (ccw 90 degree increments only)
Default	=	0
Range	=	0, 90, 180, 270

This command has the following effects:

1. The logical page coordinate system origin (0,0) rotates with the logical page. For example, rotating a default page (portrait orientation, 0° print direction) by 90° causes data to print in the landscape direction across the "portrait" page. That is, orientation is changed from portrait to landscape. Page width, length, top offset, and left offset are set appropriately.
2. If CAP is fixed (i.e., following printable data or commands affecting CAP), it remains at the same location on the physical page. If CAP is floating (i.e., before printable data or commands affecting CAP), it remains floating.
3. The margins are translated (as shown on the next page). For example, a print direction change from 0 to 90 degrees makes the left margin the new top margin, the top margin the new right margin, etc.
4. The positions stored in the CAP stack are translated to reference the same location on the physical page. For example, if print direction changes from 0 to 90 degrees, stored X coordinates become Y coordinates, and vice versa. This allows the Push/Pop CAP command to store exact physical page positions, regardless of print direction.
5. The picture frame anchor point coordinates are transformed so the anchor point and picture frame remain at the same location on the physical page.
6. Raster mode is terminated.
7. All subsequent print entities — raster, area fills, and characters — are rotated.
8. The HP-GL/2 coordinate system is also rotated if the Enter HP-GL/2 command (*Esc%#B*) has been sent with a value field of 2 or 3. Unless *Esc%#B* is sent with a value of 2 or 3, print direction has no effect on HP-GL/2 graphics, which can then only be rotated by RO or the Orientation command (*Esc&l#O*).
9. The pattern reference point (tiling) is not affected. However, the tiles may rotate to coincide with the current print direction depending on the argument passed with the Pattern Reference Point command (*Esc*p#R*).
10. The auto macro overlay environment is not disabled, since an overlay may invoke this command.
11. Print Direction does not default CMI.
12. The dither matrix (*Esc*t#J*, *Esc*m#W*) is fixed and does not move with print direction or orientation..



Changing print direction

7.5 CMI and LMI (formerly HMI and VMI)

In a fixed-space font, the Character Motion Index (CMI) defines the width of columns (horizontal text path) or the height of rows (vertical text path) used inter-character movement calculations. In a proportional font, CMI affects only the space character (unless space is downloaded as a glyph: then the printed space remains the glyph width). CMI is specified in 1/120" units.

The Line Motion Index (LMI) defines the distance between lines of print. It is the distance CAP moves for a linefeed operation. LMI is specified in 1/48 inch increments.

Character Motion Index (CMI) *Esc & k # h/H*

In horizontal text path mode (*Esc&c#T*), CMI designates the width of columns used for horizontal movement calculations; in vertical text path mode, CMI designates the height of rows used for inter-character movement calculations.

Value(#)	=	Number of 1/120 inch increments.
Default	=	Determined by the pitch value in the default font descriptor.
Range	=	0 to 2^{32} -1

CMI defaults to the width of the invoked font's space character when:

- Any of the font's characteristics (orientation, character set, pitch, etc.) are changed.
- Primary and secondary fonts are switched via SI and SO
- Text Path Direction (*Esc&c#T*) is changed.

For fixed pitch fonts, CMI affects all printable characters, including the space and backspace characters. For proportionally spaced fonts, CMI may affect only the space character: if the space character glyph exists, CAP moves the width of the space character; otherwise CAP moves according to CMI. For dual-pitch fonts, CMI directly affects the nominal (full-width) space of the font. Other spacings are linearly scaled according to the current CMI value: charter widths are multiplied by the ratio of the CMI to nominal width.

In vertical text path direction, CMI defaults to 112% of the font height. In fixed or dual pitch fonts, CMI assumes the 112% of the fixed-pitch font height. In proportionally-spaced fonts, CMI may affect only the space character. If the space character glyph exists, CAP moves 112% of the space character height; otherwise CAP moves according to CMI.

Devices which do not have an integral number of CMI units-to-dots should implement fractional CMI units for dot addressing.

Line Motion Index (LMI) *Esc & l # c/C*

Sets the vertical spacing between lines of print (the vertical distance CAP will move for a linefeed in horizontal text path mode, or the horizontal distance CAP will move for a linefeed in vertical text path mode).

Value(#)	=	Number of 1/48 inch increments between two consecutive lines of print.
Default	=	8
Range	=	0 to the current logical page length up to 2^{32} -1

This command performs the same functions as Line Spacing (*Esc&l#D*), except the measurement interval is in 1/48 inch increments instead of lines-per-inch (lpi). Both commands set linefeed and half linefeed spacing. To convert lpi to LMI:

$$\text{LMI} = 48.0 / \text{lpi}$$

If Page Length (*Esc&l#P*) follows an LMI change, physical page size is recalculated. Depending on the LMI modification, the printer may request a different page size.

A font change does not affect LMI.

Devices which do not have an integral number of LMI units-to-dots should implement fractional LMI units for dot addressing.

DEVICE NOTE: LJII and LJ2000 allow LMI to be set from the front panel.

Line Spacing *Esc & l # d/D*

Sets the number of lines printed per inch.

Value(#)	=	Number of lines per inch (lpi)
Default	=	6
Range	=	0 to the current logical page length up to a maximum of 32767

A value of 0 defaults line spacing to 6.

This command performs the same function as LMI (*Esc&l#C*), except that it sets LMI in lines per inch (lpi). Both commands set linefeed and half linefeed spacing. To convert LMI to lpi:

$$\text{lpi} = 48.0 / \text{LMI}$$

DEVICE NOTE: LJIIs and later support line spacings 1-4, 6 (default), 8, 12, 16, 24, 48 lpi.

7.6 Text Path

Asian printing requires the ability to print text vertically. Text Path Direction (*Esc&c#T*) provides one horizontal and two vertical printing modes for text.

Text Path Direction *Esc & c # t/T*

Determines the direction of CAP movement, whether vertical substitutes are applied, whether certain characters are rotated, and whether linefeeds cause vertical or horizontal movement.

Value(#)	=	-1	Rotated-character printing. Horizontal printing with full-width characters rotated 90° counter-clockwise. CAP advances left to right; linefeed advances top to bottom. Vertical substitutes are applied.
	=	0	Horizontal printing. CAP advances left to right; linefeed advances top to bottom. No vertical substitutes are applied.
	=	1	Vertical printing. One-byte characters are rotated 90°. CAP advances top to bottom; linefeed advances right to left. Vertical substutes are applied.
Default	=	0	
Range	=	-1, 0, 1	

A *full-width* character is a character whose escapement equals the font height. For TrueType, a full-width character's escapement in design units equals the font's scale factor. A *partial-width* character has an escapement less than the font height. *One-byte* characters are those that may be described by a symbol set size of less than 256; *two-byte* characters require a symbol set size of 256 or greater.

Rotated-character printing mode (-1), which is based on escapement, has the following effects:

- Advances CAP and linefeed in the same directions as horizontal (0) printing mode.
- Rotates full-width characters in large (multiple-byte) fonts 90° counterclockwise.
- Makes substitutions for characters that change their appearance, orientation, or positioning when written vertically.
- Affects only full-width characters, not partial-width characters.

Vertical printing mode (1), which is based on the number of bytes, has the following effects:

- Unlike 0 and -1, vertical mode advances CAP from top to bottom and linefeed from right to left.
- Rotates only one-byte characters 90°clockwise.
- Makes substitutions for characters that change their appearance, orientation, or positioning when written vertically.

To summarize the differences between the -1, 0, and 1 modes: "0" prints rows in portrait; "1" prints columns in portrait; and "-1" prints rows as columns in landscape. The -1 parameter, based on escapement, prints partial-width characters upright and rotates full-width characters 90° counterclockwise. The 1 parameter, based on the number of bytes, prints two-byte characters upright and rotates one-byte characters 90° clockwise. Therefore, characters printed by the -1 and 1 parameters will be rotated 90° from each other.

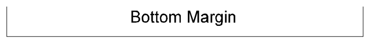
The rotated-character mode (-1) can be used to transform a portrait page with horizontal text into a landscape page with vertical text. (The Print Direction (*Esc&a#P*) command can be used to achieve other text orientations.)

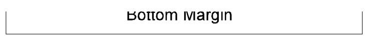
In the vertical printing mode (1), the character baseline for fixed and dual-fixed spacing fonts is vertical and through the center of the character. The starting CAP for a character is the top-center of the character. After printing the first character, CAP moves down by the CMI inter-character index (the default CMI is 112% of font height). After printing the first line, a linefeed moves CAP to the left of the current line by the linefeed distance specified by LMI.

Text Path Direction is secondary to Orientation (*Esc&l#L*) and Print Direction (*Esc&a#P*). Orientation determines the position of the logical page on the physical page (portrait, landscape, reverse portrait, reverse landscape). Print direction determines the position of the logical page coordinate system with respect to the current orientation. Then, using orientation and print direction as a reference, Text Path Direction applies the character-advance and linefeed directions. Raster graphics is unaffected and always prints along the increasing X axis of the logical page, whether in the portrait direction or as set by the Print Direction command.

For both the 1 and -1 modes, substitutions are made for characters that change their appearance, orientation, or positioning when written vertically. Vertical substitution characters are accessed through the Vertical Substitutes Character Segment downloaded with a Format 16 font (see Chapter 10).

EscE defaults text path direction, which is part of the modified print environment.





Vertical Printing Mode (1) **Rotated-character Mode (-1)**

NOTE: In the diagrams above, the printed characters denoted by "ABCDEF" are two-byte characters in vertical printing mode (1) and full-width characters in rotated-character mode (-1). In vertical printing mode (1), two-byte characters are not rotated; in rotated-character mode (-1), partial-width characters are not rotated.

7.7 Margins and Text Area

Margins are related to the logical page, not the physical page. Since the printer can only address the area within the logical page, the actual distance from the text area to the edge of the physical page is the margin plus the distance between the edge of the physical page and the edge of the logical page.

Margins represent a physical position and do not change with subsequent CMI changes. However, margins do correspond to different character positions for different horizontal print pitches.

NOTE: Text margins and graphics margins may differ. See Chapter 13 for information on raster graphics margins.

The only way to move CAP outside the margins is with the CAP Move commands (e.g., *Esc&a#C*, *Esc&a#H*). BS is ignored when CAP is at the left margin (in horizontal text path mode).

Once outside the right or left margins, all features except CR function normally to the edge of the logical page. For example, if CAP is to the right of the right margin, data is printed and CAP updated accordingly.

Text Area: Area defined by the left margin, right margin, top margin, and text length. The text area is entirely contained by the logical page: it may be the same size, or be restricted by margins within the logical page. Characters can be printed outside the text area under the following conditions:

- CAP Move commands can position CAP outside the text area for printing anywhere within the printable area.
- If perforation skip mode is disabled, characters are printed in the perforation region, between the bottom of the text area and the top of the text area on the next page.
- If perforation skip mode is enabled, characters may still be printed in the perforation region outside the text area (using a negative relative vertical CAP move to position CAP within the top margin, or a positive relative CAP move to position CAP within the bottom margin).

DEVICE NOTE: On DJs below 1200, the perforation area includes the bottom margin of the current page and the top margin of the next page; therefore, when CAP is within the top margin area a linefeed moves CAP to the top margin if perforation skip is enabled. On LJs and the DJ1200C and above, the perforation area includes only the bottom margin of the current page; therefore, when CAP is within the top margin area a linefeed moves CAP one line down if perforation skip is enabled.

Left Margin: Distance between the left edge of the logical page and the left edge of the text area. The term "left margin" may also refer to left edge of the text area.

Right Margin: Distance between the left edge of the logical page and the right edge of the text area. The term "right margin" may also refer to right edge of the text area.

Top Margin: Distance between the top of the logical page and the top of the text area. The term "top margin" may also refer to the top edge of the text area.

Bottom Margin: Distance from the bottom of the text area to the bottom of the logical page. There is no bottom margin command. The bottom margin can only be set indirectly by setting text length when perforation skip mode is enabled. The bottom margin may be calculated by subtracting the top margin and text length (at the current LMI) from the (logical page length plus top margin).

Text Length: Distance from the top margin to the bottom of the text area. Text length has meaning only if perforation skip mode is enabled. When perforation skip is enabled, text length defines the bottom margin.

Perforation Region: Distance from the bottom of the text area on the physical page to the top of the text area on the next physical page. Enabling perforation skip (the default) causes text to be print to the end of the text area of the current page and start again at the top of form on the next page. Disabling perforation skip mode causes text to be printed to the end of the page, into the unprintable region, and onto the the top edge of the next page; text length is ignored and the top margin is ignored except when executing <FF>, which moves to the top of form. The remainder of a partial line is not carried over to the next page. Historically, this term originated with continuous feed devices.

DEVICE NOTE: On DJs below 1200, the perforation area includes the bottom margin of the current page and the top margin of the next page; therefore, when CAP is within the top margin area a linefeed moves CAP to the top margin if perforation skip is enabled. On LJs and the DJ1200C and above, the perforation area includes only the bottom margin of the current page; therefore, when CAP is within the top margin area a linefeed moves CAP one line down if perforation skip is enabled.

Top of Form: After power on or reset, CAP moves to the baseline of the first row of characters. This position, called *top of form*, is 3/4 of a line below the top margin:

top of form = top margin + (3/4 * line spacing)

The phrase "move to top of form" means if the CAP is not at top of form, move it to the top of form of the next logical page.

In printers that have no means of sensing the edges of a sheet there is no guarantee that the top of form will be a specified distance from the physical edge of the sheet.

Printable Area: This is the area of the physical page in which the printer is able to place a dot; this is usually determined by the technology of the printing device. Although the text area may be larger than the printable area, text outside the printable area is lost.

Clear Horizontal Margins *Esc 9*

Resets the left and right margins to their default positions. CAP is unchanged.

Left Margin	=	left logical page boundary (column 0)
Right Margin	=	right logical page boundary

Left Margin *Esc & a # l/L*

Sets the left margin to the left edge of the specified column.

Value(#)	=	Column number
Default	=	Column 0 (left logical page boundary)
Range	=	0 to the right margin

If CAP is to the left of the new left margin, it moves to the new left margin; otherwise, this command does not affect CAP. Attempts to set the left margin to the right of the right margin are ignored; however, left and right margins can be set at the same location.

NOTE: Column 0 is the first column on the left edge of the logical page, not the physical page.

Right Margin *Esc & a # m/M*

Sets the right margin to the right edge of the specified column.

Value(#)	=	Column number
Default	=	Logical page right boundary
Range	=	Current left margin minus the logical page right boundary

This command is ignored for columns preceding the left margin. Specifying a column beyond the right logical page limit sets the right margin to the right logical page limit. If CAP is to the right of the new right margin, CAP is moved to the new right margin; otherwise this command does not affect CAP.

A characters whose escapement (delta X) would carry CAP across the right margin is not printed (and CAP is unchanged) unless end-of-line wrap is enabled: then the character prints at the left margin on the next line. Attempts to set the right margin to the left of the left margin are ignored; however, left and right margins can be set to the same location.

DEVICE NOTE: LJ2000 and the LJIII and 4 series print cells that overlap the right margin. Earlier LJ's clip overlapping characters.

Top Margin *Esc & l # e/E*

Specifies the distance between the top of the logical page and the top of the text area.

Value(#)	=	Number of lines from the top of the logical page
Default	=	½"
Range	=	0 to the length of logical page

DEVICE NOTE: DJs below 1200 default the top margin to ½" with perforation skip enabled, and 0" with perforation skip disabled.

The top margin is specified in lines whose spacing is determined by the current line spacing. This command is ignored if the current line spacing is 0, or if a value beyond the current logical page length is received.

This command does not affect a fixed CAP until the next page. A floating CAP is placed at the baseline of the first row of characters. This position, called *top of form*, which is the vertical position of the first line of print on the logical page, is calculated by:

Top of Form = top margin in inches + (.75 x LMI in inches)

Raster also starts at the top of form.

The top margin represents a physical position: once the top margin is set, it is unaffected by subsequent changes in line spacing.

NOTE: Since this command defaults text length, it should precede Text Length (*Esc&l#F*).

Text Length *Esc & l # f/F*

Sets length of the text area in lines from the top margin.

Value(#)	=	Number of lines
Default	=	Logical page length minus top margin minus 1/2 inch *
Range	=	0 to the logical page length minus top margin

* A negative result defaults text length to (logical page length - top margin).

This command is ignored if current LMI is 0 or a text length greater than (logical page length - top margin) is requested. A value field of 0 defaults text length.

Text length determines the bottom margin when perforation skip (*Esc&l#L*) is enabled. When perforation skip is disabled, text length is not used, but is saved

Perforation Skip Mode *Esc & l # l/L*

Controls perforation skip mode.

Value(#)	=	0	Disables perforation skip mode
	=	1	Enables perforation skip mode
Default		=	1 *
Range		=	0,1

* Historically, the default has been device dependent; "1" is recommended.

The perforation region extends from the bottom of the text area to the top margin of the next page (see the device note below for LaserJets and DeskJets 1200 and above. Text Length (*Esc&l#F*) determines the the perforation size (and the bottom margin).

DEVICE NOTE: On DJs below 1200, the perforation area includes the bottom margin of the current page and the top margin of the next page; therefore, when CAP is within the top margin area a linefeed moves CAP to the top margin if perforation skip is enabled. On LJs and the DJ1200C and above, the perforation area includes only the bottom margin of the current page; therefore, when CAP is within the top margin area a linefeed moves CAP one line down if perforation skip is enabled.

If perforation skip is enabled:

- In horizontal text path mode, printing ends at the specified text length and then starts at the top margin of the next page.
- In vertical text path mode, printing ends at the left margin and then starts at the right margin of the next page, with the vertical position maintained.

If perforation skip is disabled:

- In horizontal text path mode, text is printed to the end of one page onto the top of the next page; bottom and top margins are ignored.
- In vertical text path mode, text is printed to the left edge of the logical page and onto the right edge of the next page. Both the right and left margins is ignored.

Except for formfeed, perforation skip has no effect on top of form.

NOTE: Disabling perforation skip may cause text outside of the printable area to be lost. Enabling perforation skip mode enables top margin and text length.

DEVICE NOTE: DJs below 1200 default the top margin to ½" with perforation skip enabled, and 0" with perforation skip disabled.

Chapter 8: CAP

Contents of this Chapter

This chapter describes the following PCL commands:

8.1 Introduction

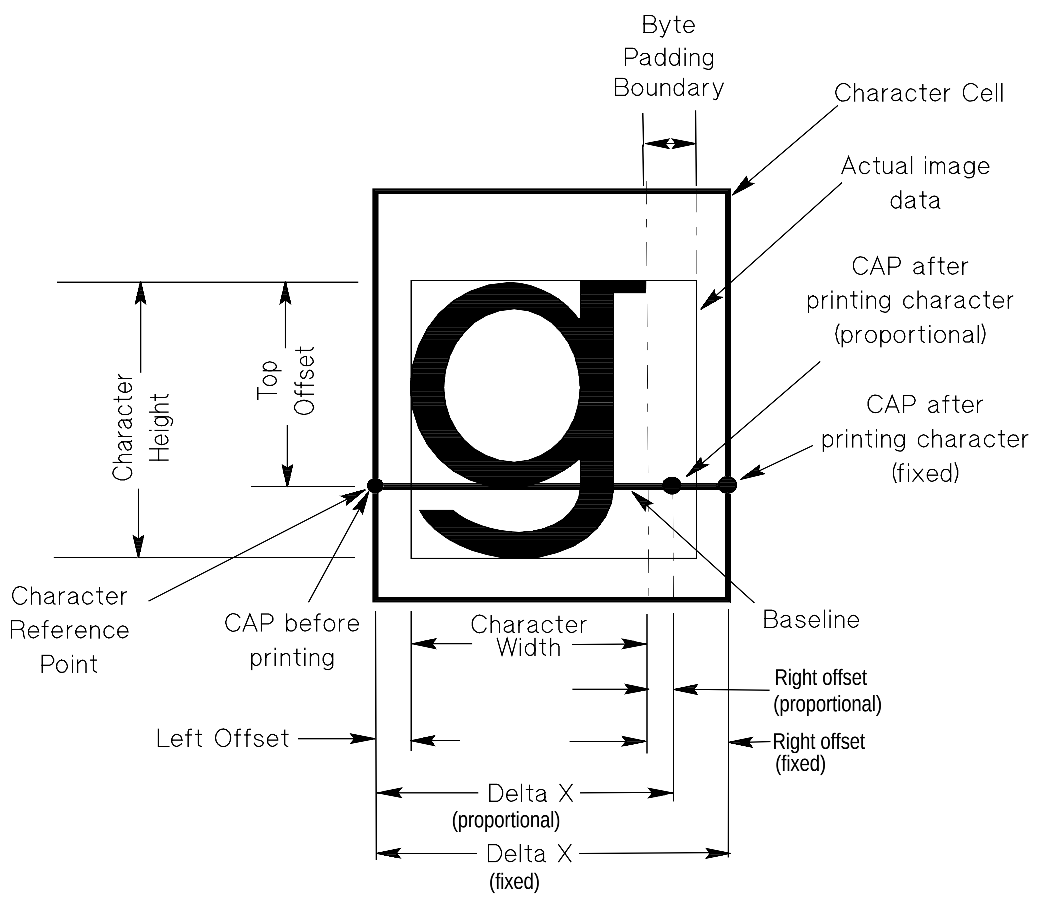
CAP (Current Active Position) is where the next character or graphics dot will be printed. The position of CAP after an image is printed depends on the type of image: text, raster graphics, vector graphics, or rectangular area fills.

CAP can be moved anywhere within the logical page using horizontal and vertical move commands.

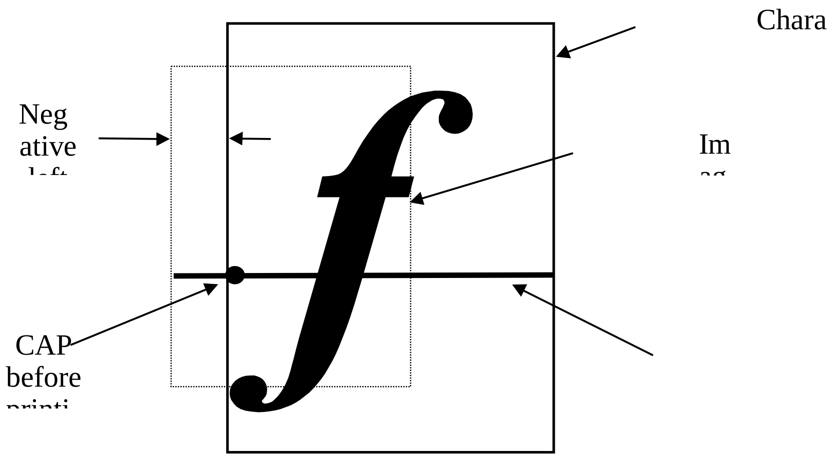
CAP for Text

Before a character is printed, CAP is the *character reference point*, on the baseline at the left of the character cell. After printing the character, CAP moves to the right by the *escapement* (delta X) of the character. The diagrams on the next page show characters having both positive and negative offsets (side bearings): i.e., the image (black) data may start before or after CAP.

For fixed-spaced fonts, CAP moves the CMI distance for each character printed (e.g., if CMI is 12/120", CAP moves 1/10" along the baseline for each character printed. For proportional-spaced fonts, the amount of space (escapement) moved for each character is obtained from font character data.



Character Cell in Portrait Orientation



Negative Left Offset (Side Bearing)

CAP for Raster Graphics

At the start of raster graphics, CAP is at the current Y position on the left graphics margin. After printing the raster image, CAP moves to the left graphics margin at the dot row following the last row of raster data. If a Raster Height command was used, CAP is finally located on the left graphics margin one dot row below the lowest part of the picture frame defined by the Raster Height command.

CAP for Vector Graphics

In HP-GL/2 mode, CAP is referred to as the *pen location*. Whenever a plotting instruction is completed, the pen location is updated so that the next instruction begins at the current point. However, there are some instructions, such as CI (circle), that do not change the current pen location. The definition of each HP-GL/2 command tells whether CAP is changed.

When entering HP-GL/2 from PCL mode, pen location can be specified as either the current CAP or the pen location prior to entering PCL mode. When entering PCL mode from HP-GL/2 mode, CAP can be specified as either the last CAP prior to entering HP-GL/2 mode, or as the last pen location prior to entering PCL mode.

NOTE: The HP-GL/2 coordinate system is different from the PCL coordinate system. See the *Vector Graphics* chapter for more information.

CAP for Rectangular Area Fills (Rules)

CAP starts printing rules at the upper left corner. CAP does not change: after printing it is in the same position as it was before printing.

Print-and-Space CAP Positioning

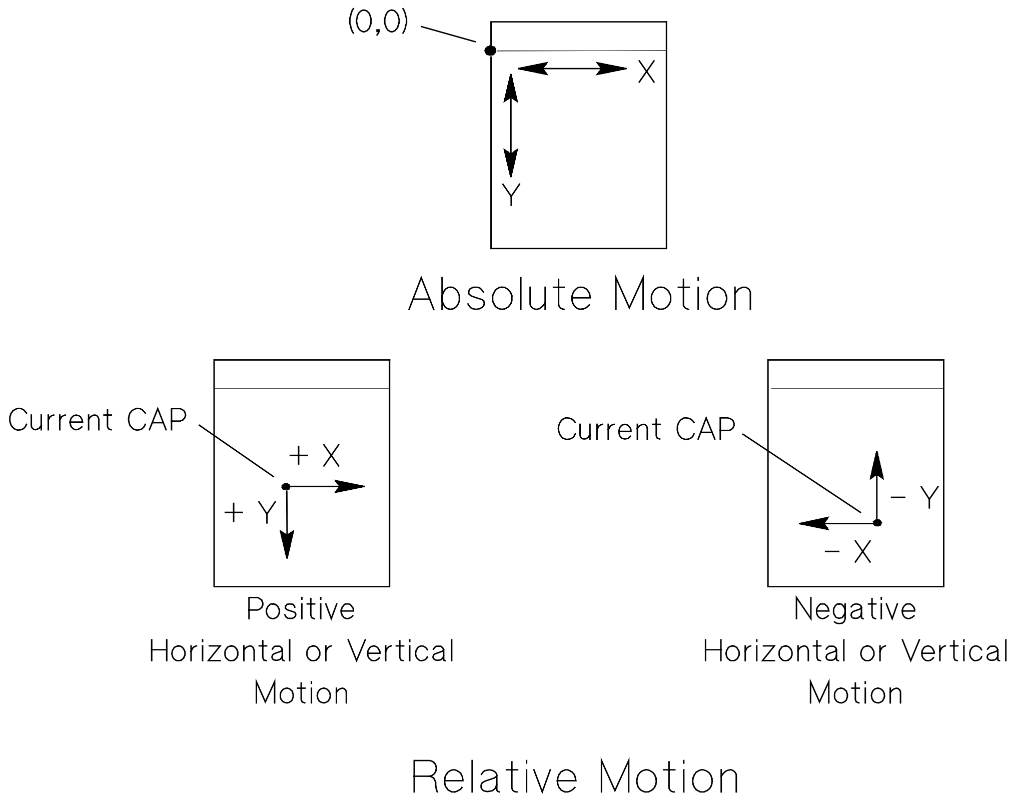
Print-and-space applications move CAP using the space, backspace, linefeed, and carriage return control codes instead of CAP Move commands.

Absolute and Relative Movement

Relative motion is referenced to the current active postion (CAP). *Absolute* motion is referenced to the intersection of the top margin and left edge of the logical page (0,0). A signed value field in a CAP movement command indicates relative movement.

Relative movement is used to establish a distance between objects. For example, many word processing applications use relative horizontal movement to justify a line of text. After determining the number of words on a line, the application calculates the amount of space between each word. Then relative moves are used after each word instead of the space character.

Absolute movement is used to guarantee an absolute distance from the left edge of the logical page. Placing a raster graphic on a certain spot is usually easier if the position is relative to a fixed position and not dependent on the distance from the previous image.


(0,0)
X
Y
Absolute Motion
Current CAP
+ X
+ Y
Positive Horizontal or Vertical Motion
Current CAP
- Y
- X
Negative Horizontal or Vertical Motion
Relative Motion

Absolute/Relative CAP Movement

Units of Movement

CAP positioning may be specified in **PCL Units**, **decipoints**, or **columns** on the X axis, and in **PCL Units**, **decipoints**, or **rows** on the Y axis.

Printers may map PCL Units, decipoints, columns, and rows to an internal measuring unit. All positioning is kept in internal units and rounded to dot positions when data is printed. This eliminates errors caused by successive rounding or truncation.

DEVICE NOTE: The LJIII and 4 series use an internal measuring unit of 1/7200 inch in the PCL language and 1/9600 inch in HP-GL/2. LJII uses 1/3600 inch and DJs below 1200 use 1/3600.

Decipoints

A decipoint is 1/720 of an inch, or 1/10 of a PCL point. (A PCL point is *exactly* 1/72 of an inch; a typeographic point is *approximately* 1/72 of an inch.)

Columns

The size of a column depends on the current CMI (horizontal motion index), which is initially the width of the font's space character, and changes with the selected font. For example, at 12 columns per inch (cpi) one column is 1/12th inch, and at 10 cpi one column is 1/10th inch.

PCL Units

The Unit of Measure command (*Esc&u#D*) specifies PCL Unit size in units-per-inch. The default unit for PCL movement is 1/300 of an inch. This is true even when a different printing resolution (such as 600 dpi) is selected.

PCL Units are used in the following commands:

- Move CAP Vertical (PCL Units) — *Esc*p#Y*
- Move CAP Horizontal (PCL Units) — *Esc*p#X*
- Vertical Rectangle Size (PCL Units) — *Esc*c#B*
- Horizontal Rectangle Size (PCL Units) — *Esc*c#A*

In addition, since the current units of measure setting affects how CAP movement values are rounded, it also affects the result of the following commands:

- Move CAP Horizontal (Columns) — *Esc*&a#C*
- Horizontal Tab — <*HT*>

HISTORICAL NOTE: PCL Units are independent of device resolution. However, in the past some machines have used "dot" movement, which varied with device resolution. For example, one dot equals 1/180" on PJ and PJXL, 1/300" on DJ and LJ, and 1/1200" on some typesetters.

Rows

Row positioning is dependent on the current LMI (vertical motion index). LMI is the distance between consecutive lines of text. This distance varies with line spacing. For example, at six lines per inch (lpi), one line is 1/6 of an inch; and at eight lpi, one line is 1/8 of an inch.

HP-GL/2 Units

In HP-GL/2 mode, the device moves in either *plotter units* or *user units*. One plotter unit equals .025 mm (approximately .00098 in). User units can be set to any size by HP-GL/2 scaling commands. Both types of HP-GL/2 units are converted to the device resolution prior to printing.

Default CAP

When the printer is powered on or reset, CAP is determined by the current top margin, left margin, orientation, and LMI settings when the first printable data or command affecting the CAP is received. Printable data (including printable SPACE) is defined as a mark on the page or graphics character. Commands affecting CAP include ASCII data, LF, space, explicit CAP move, etc.

Floating CAP

CAP is *floating* after power-on or reset until printable data or a command affecting CAP is received. A floating CAP moves in accordance with orientation, top margin, left margin, and line spacing commands.

Fixed CAP

CAP is *fixed* after a printable character or a command affecting CAP is received, A fixed CAP is not affected by changes to the orientation, top margin, left margin, or line spacing.

Once CAP is fixed, page length or orientation changes cause a page eject (FF-CR). This is called a *conditional page eject* because it occurs only if the page is "dirty". Some PCL commands (e.g., Duplex Page Side) cause an unconditional page eject, which occurs regardless of whether CAP is fixed or floating.

8.2 Horizontal Positioning Commands

Absolute horizontal positioning is referenced to the left logical page boundary. Relative horizontal positioning is referenced to CAP.

NOTE: The control codes <BS>, <CR>, <HT>, and <SP> also move CAP horizontally. See Chapter 5 for a description of these characters.

Move CAP Horizontal (Decipoints) *Esc & a # h/H*

Moves CAP horizontally by the specified number of decipoints (1/720 inch).

Value(#)	=	Number of decipoints
Default	=	NA
Range	=	-2^{31} to 2^{31} -1 (up to the logical page right and left limits)

A signed value field indicates *relative* movement: plus (+) or minus (-) signs move CAP right or left relative to CAP, respectively. The absence of a sign indicates *absolute* movement: CAP moves an absolute distance from the logical page left edge.

This command ignores margins and can move CAP anywhere horizontally within the logical page. Attempts to go outside the logical page will move CAP to the appropriate logical page limit.

Devices not having an integral number of decipoints-to-dots should implement fractional decipoints for dot addressing.

Move CAP Horizontal (Columns) *Esc & a # c/C*

Moves CAP horizontally by the specified number of columns.

Value(#)	=	Number of columns
Default	=	NA
Range	=	-2^{31} to 2^{31} -1(or to the logical page right and left limits)

A signed value field indicates *relative* movement: plus (+) or minus (-) signs move CAP right or left relative to CAP, respectively. The absence of a sign indicates *absolute* movement: CAP moves an absolute distance from the logical page left edge (column 0).

This command ignores margins and can move CAP horizontally anywhere within the logical page. Attempts to go outside the logical page will move CAP to the appropriate logical page limit.

The CMI command (*Esc&k#H*), which defines the size of a column, rounds CMI to the nearest internal unit. Devices supporting Unit of Measure (*Esc&u#D*) always round a selected font's CMI to the nearest PCL Unit, regardless of whether Unit of Measure has been specified ; however a subsequent CMI command will again round CMI to the nearest internal unit.

DEVICE NOTE: DJ850C does not round CMI to the nearest PCL Unit. Unit of Measure affects only the start of characters and PCL Unit commands.

Move CAP Horizontal (PCL Units) *Esc * p # x/X*

Moves CAP horizontally by the specified number of PCL Units.

Value(#)	=	Number of PCL Units
Default	=	NA
Range	=	-2^{31} to 2^{31} -1(up to the logical page right and left limits)

A signed value field indicates *relative* movement: plus (+) or minus (-) signs move CAP right or left relative to CAP, respectively. The absence of a sign indicates *absolute* movement: CAP moves an absolute distance from the logical page left edge.

Unit of Measure (*Esc&u#D*) specifies the current size of a PCL Unit. If Unit of Measure is not supported or is not specified, the unit-per-inch for PCL movement defaults to 1/300 inch, or to the control panel setting, which has precedence.

This command ignores margins and can move CAP horizontally anywhere within the logical page. Attempts to go outside the logical page will move CAP to the appropriate logical page limit.

DEVICE NOTE: PJ, PJXL, and Rugged Writer use 1/180" units. DJs below 1600, PJXL300, and LJs below LJ4 use a 1/300" unit or "dot" size.

8.3 Vertical Positioning Commands

Vertical positioning may be either absolute or relative. A signed value field indicates *relative* positioning; the absence of a sign indicates *absolute* positioning. Relative positioning is referenced to CAP. Absolute vertical positioning by row is referenced to top of form; absolute vertical positioning by decipoints or PCL Units is referenced to the top margin.

Vertical CAP positioning ignores perforation skip mode, allowing the user to move into the perforation region.

NOTE: The control codes <LF> and <FF> also move CAP vertically. See Chapter 5 for a description.

Move CAP Vertical (Rows) *Esc & a # r/R*

Moves CAP to the same column position on a new line based on the active line spacing.

Value(#)	=	Number of rows
Default	=	NA
Range	=	-2^{31} to 2^{31} -1(fractional values are allowed; valid to four decimal places)

A signed value field indicates ***relative*** movement: a minus (-) sign moves CAP upward relative to CAP; a plus (+) sign moves CAP downward relative to CAP.

Negative relative movement can extend into the top margin, where it is clamped to the logical page boundary. For this command only (not the other vertical move commands), positive relative movement can extend to the next logical page's lower boundary, where it is clamped. However, movement beyond the current logical page causes the current page to be printed.

The absence of a sign indicates ***absolute*** movement: CAP moves from the top of form an absolute distance computed by:

$$\text{Distance from top of logical page} = \text{top margin} + (3/4 \times \text{LMI}/48.0) + (\text{value} \times \text{LMI})/48.0$$

After power on or *EscE*, CAP moves to the top of form, if not already there.

Move CAP Vertical (Decipoints) *Esc & a # v/V*

Moves CAP to a new position along the vertical axis. If no such position exists, moves CAP to the logical page limit.

Value(#)	=	Number of decipoints
Default	=	NA
Range	=	-2^{31} to 2^{31} -1(rounded to the first decimal place)

A signed value field indicates ***relative*** movement: a minus (-) sign moves CAP upward relative to CAP; a plus (+) sign moves CAP downward relative to CAP.

Negative relative movement can extend into the top margin, where it is clamped to the logical page boundary. Positive relative movement is clamped to the bottom of the current logical page.

The absence of a sign indicates ***absolute*** movement: CAP moves an absolute distance from the top margin.

After power on or reset, CAP moves to the top of form, if not already there.

Devices not having an integral number of decipoints-to-dots should implement fractional decipoints for dot addressing.

Move CAP Vertical (PCL Units) *Esc * p # y/Y*

Moves CAP to a new position along the Y axis. If no such position exists, the printer moves to the logical page limit.

Value(#)	=	Number of PCL Units
Default	=	NA
Range	=	-2^{31} to 2^{31} -1

A signed value field indicates ***relative*** movement: a minus (-) sign moves CAP upward relative to CAP; a plus (+) sign moves CAP downward relative to CAP.

Negative relative movement can extend into the top margin, where it is clamped to the logical page boundary. Positive relative movement is clamped to the bottom of the current logical page.

The absence of a sign indicates ***absolute*** movement: CAP moves an absolute distance from the top margin.

Unit of Measure (*Esc&u#D*) specifies the current size of a PCL Unit. If Unit of Measure is not supported or is not specified, the unit-per-inch for PCL movement defaults to 1/300 inch, or to the control panel setting, which has precedence.

After power on or *EscE*, CAP moves to the top of form, if not already there.

DEVICE NOTE: PJ, PJXL, and Rugged Writer use 1/180" units. DJs below 1600, PJXL300, and LJs below LJ4 use a 1/300" unit or "dot" size.

8.4 Saving CAP

Some situations require the device to be able to save previous CAP positions. A common application is saving CAP prior to executing, calling or overlaying a macro, and then recalling the position after the macro command has been carried out.

Push/Pop CAP *Esc & f # s/S*

Allows CAP to be stored and recalled.

Value(#)	=	0	Push CAP
	=	1	Pop CAP
Default	=	0	
Range	=	0,1	

This command uses a stack to store or recall CAP: a value of 0 *pushes* CAP, leaving the present CAP unaffected. A value of 1 *pops* CAP, restoring it as the new CAP.

As with any stack, the last item pushed is the first item popped.

If a popped CAP is outside the logical page, it is moved to the appropriate logical page boundary.

The Print Direction command (*Esc&a#P*) causes the values stored on the stack to be transformed so as to maintain their physical locations on the page.

Attempts to save more positions than the device is capable of are ignored. Attempts to recall more positions than were stored are ignored. Reset (*EscE*) discards all stored values.

DEVICE NOTE: LJ's and DJs above 1000 have a maximum CAP stack size of 20.

Chapter 9: Font Selection

Contents of this Chapter

This chapter discusses the following PCL commands:

9.1 Introduction to Fonts

A font is a group of symbols that have similar characteristics. A font is described by its **symbol set**, **spacing**, **height**, **pitch**, **style**, **stroke weight**, **typeface**, and **orientation**. The user selects one font for printing at any one time. Fonts may be selected by ID or by their characteristics. This chapter describes the PCL commands that select a font.

Font Sources

The fonts used by a printer may be supplied with the printer (internal), may be added as ROM memory (cartridge and/or SIMM), or may be downloaded from the host (soft or RAM fonts). (Chapters 10 and 11 describe font downloading.)

Bitmap Fonts

Bitmap fonts have a fixed bit-pattern for each character. The initial size of each character is fixed, since the bit-pattern is fixed. Algorithmic enhancements to bitmpap fonts can create effects such as half-height, half-width, double-width, double-height, slant, and bold.

Scalable Fonts

Scalable fonts are stored as outlines that are not limited to a fixed size. An algorithm scales the outline to create fonts of different sizes.

Unbound Fonts

Unbound fonts are not limited to a single symbol set. The other font characteristics — spacing, style, stroke weight, and typeface — are all fixed. (Unbound fonts are discussed in Chapter 12.)

How Fonts are Selected

A font is a set of printable images specified by a collection of attributes. A change in just one attribute may indicate a different font. For example, 10-point Times Roman bold is a different font than 10-point Times Roman italics, or 12-point Times Roman bold.

The user "designates" a font by specifying its attributes. The printer "selects" a font by comparing requested attributes with those in its available fonts. Each font attribute has a different priority (e.g., symbol set has a higher priority than height, which has a higher priority than typeface).

The Font Select Table

The printer maintains a table containing all the attributes of the currently requested font. Font selection is made by specifying all the desired font’s characteristics using the font selection escape sequences in this chapter. For example, after specifying a Roman-8, fixed-space, 10 pitch, 12 point, upright, bold, Courier font, the font select table will appear as follows:

Symbol Set	Roman-8
Spacing	Fixed
Pitch	10 cpi
Height	12 point
Style	Upright
Stroke Weight	Bold
Typeface	Courier

The table is updated whenever a new attribute is specified. For example, to select a similar font with a medium stroke weight, only that single attribute must be specified. (Note, however, that HP recommends that driver writers specify all attributes of a font for each font requested.) The table then looks like this:

Symbol Set	Roman-8
Spacing	Fixed
Pitch	10 cpi
Height	12 point
Style	Upright
Stroke Weight	**Medium**
Typeface	Courier

Before printing, the printer performs the **Font Selection Algorithm** described below to find the available font most closely matching the attributes in the font select table.

Many- HP products provide font selection from the control panel. This may establish different default values for some attributes.

NOTE: Downloadable fonts may be designated by font ID as well as attribute (see Chapter 10). When a font is selected using the ID number, the characteristics in the font select table are updated to reflect that font’s characteristics.

Attribute Priority

The printer selects a font based on its characteristics (attributes), its physical location in the printer, and its orientation. Font selection priority is then:

1. Orientation [5]
2. Symbol set
3. Spacing
4. Pitch [1] [2]
5. Height [1] [2]
6. Style [6]
7. Stroke weight [2] [6]
8. Typeface family
9. Resolution [3]
10. Quality [2]
11. Location [4]

[1] For scalable fonts, pitch and height are operators, not attributes. Any legal pitch (fixed fonts) or height (proportional fonts) can be satisfied by a scalable font.

[2] DEVICE NOTE: These are operators on DeskJets below 660. Quality is not used on LaserJets.

[3] Bitmap fonts designed at 600 dpi are not available for selection at 300 dpi. In 600 dpi mode, font priority is: 600 dpi bitmap, scalable, 300 dpi bitmap. In 300 dpi mode, font priority is: 300 dpi bitmap, scalable

[4] Although location is not a font characteristic, it is a font selection consideration.

[5] DEVICE NOTE: Orientation is an operator on LJIII and later, and DeskJet 850C and above.

[6] Native (actual) weight and style take precedence over algorithmically-obtained weight and style.

NOTE: Character enhancements have a lower priority than native(actual) fonts.

EXAMPLE

Assume the user designates a font with a symbol set of Roman-8 and a height of 16 points. If the printer contains 10-point Roman-8 and 16-point ISO 8859/1 Latin 1, the printer will select 10-point Roman-8 because symbol set has the highest priority.

The process the printer uses for selecting a font is called the "Font Selection Algorithm", which is described in detail below.

The Font Selection Algorithm

Selection by attribute is an elimination process. The device scans the available font libraries for the font whose attributes most closely match the user's font request. The requested attribute with the highest priority is compared with the corresponding attribute in the available fonts. If only one font contains the matching attribute, that font is selected. If more than one font contains the matching attribute, the printer compares the next highest priority attribute, and so on, until only one font remains. The following summarizes the procedure used to select a font from the available fonts. The steps are performed in the given order.

1. **Orientation:** For bitmap fonts similar in all the above characteristics, the font with the orientation that matches the page orientation is selected. The Logical Page Orientation command (Esc&l#O) may be necessary for some printers in order to match font orientation to logical page orientation. Devices with "auto-rotation" automatically rotate the font to match the logical page orientation; all fonts are then available in all page orientations.

2. **Symbol set:** A symbol set is a unique ordering of the characters in a font. If the requested symbol set is unavailable, the device-dependent default is selected.

DEVICE NOTE: LJs default symbol set to Roman-8 if a user-selected symbol set does not exist.

3. **Spacing:** The inter-character spacing of fonts is either "fixed", or "proportional" to the shape of the character. If proportional spacing is requested but unavailable, fixed spacing is selected in the current pitch as described in the next paragraph.

4. **Pitch:** Pitch is the number of characters printed per inch (cpi). Pitch applies only to fixed-space fonts. If fixed spacing is available, the requested pitch is selected. For bitmap fonts, if the requested pitch is unavailable, the closest greater pitch is selected; if a greater pitch is unavailable, the closest lesser pitch is selected. For scalable fonts, the closest achievable pitch (greater or lesser) is rendered. For dual-pitch fonts, the full (nominal)width is used.

5. **Height:** Font height is an approximate measure of the body of the type in points (e.g., 24 points). A point is about 1/72 inch. Since point size may vary among vendors, height is a relative measurement for selection purposes, not a metric indicating absolute character height.

The device selects an available height closest to the requested height. "Closest" means the smallest absolute difference. For example, if 10 is requested when 6, 8, and 12 point fonts are available, both 8 and 12 point fonts are picked for the next selection criteria.

No font having a height within 0.25 points of the closest absolute difference is eliminated. For example, if a 12 point font is requested, all fonts between 11.75 and 12.25 are considered. Or, if 6, 8, and 11.75 point fonts are available and 10 is requested, both 8 and 11.75 are considered (i.e., the *range*, not the endpoints, is considered).

NOTE: Any legal height is available for proportionally spaced scalable fonts.

6. **Style:** Describes the posture, width, and structure of the font symbols. (e.g., upright, italic). Style is ignored if the requested style is unavailable in the remaining fonts.

7. **Stroke Weight:** Stroke weight is the thickness of the strokes that compose characters.

If the requested weight is unavailable and 0 or greater, the closest thicker weight is selected. If a thicker weight is unavailable, the closest thinner weight is selected.

If the requested weight is unavailable and less than 0, the closest thinner weight is selected. If a thinner weight is unavailable, the closest thicker weight is selected.

8. **Typeface Family:** Typeface describes the unique distinguishing features of a font's graphics symbols (e.g. Times Roman, Helvetica). Typeface is ignored if the requested typeface is unavailable in the remaining fonts.

9. **Resolution:** All scalable fonts will automatically print at any resolution. A 600 dpi font has priority over a 300 dpi font with the same characteristics and location. A bitmap font designed at 300 dpi can be printed on a 600 dpi printer; but a bitmap font designed at 600 dpi cannot be printed at 300 dpi.

10. **Quality:** Quality (also called "density") describes the definition of the printed symbols. The nearest fit to the requested quality is selected.

11. **Location:** PCL and HP-GL/2 font location ordering, from highest to lowest priority, is:

Soft font (lowest ID) — bitmap
Soft font (lowest ID) — scalable
Read/Write removeable disk (lowest ID) — bitmap
Read/Write removeable disk (lowest ID) — scalable
Read/Write removeable flash (lowest ID) — bitmap
Read/Write removeable flash (lowest ID) — scalable
Read/Write permanent disk (lowest ID) — bitmap
Read/Write permanent disk (lowest ID) — scalable
Read/Write permanent flash (lowest ID) — bitmap
Read/Write permanent flash (lowest ID) — scalable
Cartridge font[1] — bitmap
Cartridge font[1] — scalable
SIMM font[2] — bitmap
SIMM font[2] — scalable
Internal font — bitmap
Internal font — scalable

[1] DEVICE NOTE: On LJIII the left cartridge has priority over the right cartridge. On DJ the back cartridge has priority over the front.

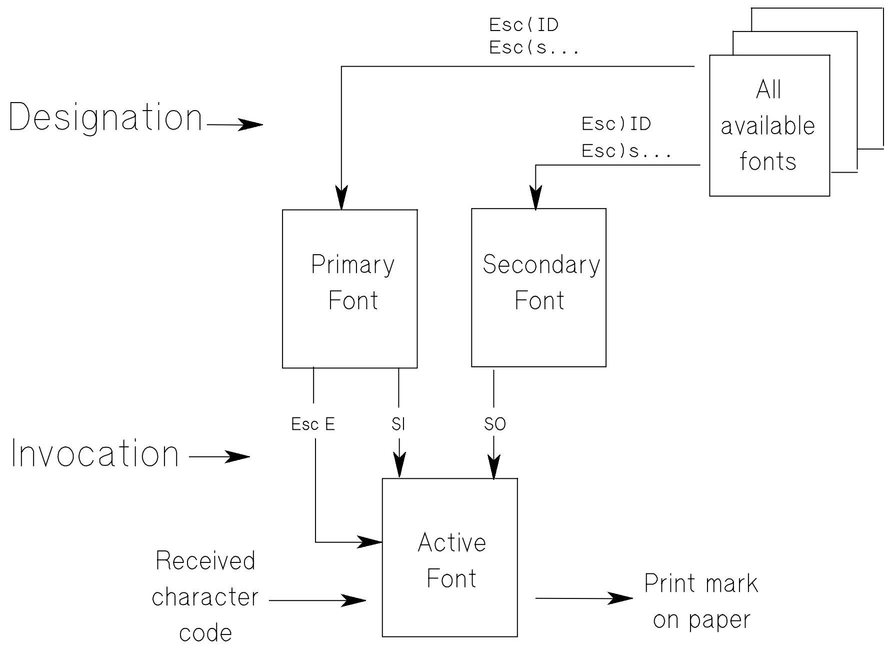
[2] SIMM1 has highest priority, then SIMM2, SIMM3, etc.

9.2 Primary and Secondary Fonts

PCL devices may maintain two independent font select tables. This provides access to a *primary* font and a *secondary* font, only one of which is selected at a time.

Power-on, reset, or the *SI* (15) control code invoke the primary font. *SO* (14) invokes the secondary font, which remains invoked until receipt of an *SI* or reset. The default fonts for the primary and secondary fonts in a PCL device are user configurable.

Font attribute selection is independent of primary/secondary font selection. Font attributes, including symbol set, need not be the same for primary and secondary fonts, since these are independent fonts.



Primary Font <*SI*>
Secondary Font <*SO*>

Activates the currently designated primary or secondary font. The primary font remains invoked until receipt of *SO*, *EscE*, or device reset. The default state is *SI*. Switching between the primary and secondary fonts defaults CMI.

9.3 Selection by Attribute

The user requests a font by designating its attributes with the following commands.

Font Symbol Set (Primary) *Esc (ID*
Font Symbol Set (Secondary) *Esc) ID*

Identifies the set of symbols in a font.

ID	=	Identification value from the table: consists of a decimal value and a letter value
Default	=	Device dependent
Range	=	NA

The ID (identification number) consists of a number portion and a letter portion. For example, to specify ASCII (0U) as the symbol set for the primary font, send *Esc(0U*. Some of the possible values for the identification number (ID) are listed on the symbol set tables on the following pages.

The legal range of the number portion of ID is 0 to 2047. The legal range of the letter portion of ID is the upper-case ASCII characters "A" through "Z", except for "X". Symbol sets with ID's of "X" can only be selected with the Font ID. "Q" is used with HP "Specials" symbol sets and not recommended for general use. "Y" is used for barcode symbol sets. "Z" is reserved.

The device-dependent default is selected if the specified symbol set does not exist or cannot be satisfied by any font.

If an unbound scalable font is selected by ID, this command should be used to specify the symbol set. Otherwise, the symbol set of the former font (as listed in the font select table) will be used. (See "Font Selection by ID" later in this chapter.)

To specify a user-defined symbol set, use the symbol set value defined by the Symbol Set ID command (*Esc*c#R*). (See Chapter 12.)

The following table of HP-supported symbol sets and IDs is current as of 1/01/95, but users should reference *The Book of Characters* (HP BPR division).

For convenience, the table is repeated with the symbol set names sorted alphanumerically.

	Symbol Set Name		Symbol Set Name
	Math-7 (same as 0M)		Engravers Subset
	Line Draw-7 (same as 0L)		Engravers
	HP Large Characters		Microsoft Publishing
	Adobe Cyrillic		DeskTop
	GB 2312 (1980 mainland China)		Document
	Chemical Pi Set #0		PC-1004
	Chemical Pi Set #1		PS Text
	Chemical Pi Set #2		PS ISO Latin 1
	Chemical Pi Set #3		MC Text
	Chemical Pi Set #4		Ventura International
	Chemical Pi Set #5		Ventura US
	Chemical Pi Set #6		Lotus LICS set
	Chemical Pi Set #7		Swash Characters
	ISO 60: Danish/Norwegian		Small Caps and Old Style Figures
	ISO 61: Norwegian Version 2		Old Style Figures
	Devanagari		Fractions
	Roman Extension		Lining Figures
	ISO 4: United Kingdom		Small Caps and Lining Figures

	Windows 3.1 Latin 2		Alternate Caps
	Adobe East European		Lotus LMBCS Group 0
	ISO 25: French (obsolete)		Lotus LMBCS Set
	ISO 69: French		Kanji Ward #0 - Bitstream
	reserved 33-255 for future Asian		Kanji Ward #99 - Bitstream
	reserved 33-255 for future Asian		ISO 14: JIS ASCII
	HP German		ISO 13: Katakana
	ISO 21: German		ISO 57: Chinese
	Greek-8		Kana-8 (JIS 210)
	Windows 3.1 Latin/Greek		Korean-8
	PC-851 Latin/Greek		JIS 208 (1990)
	PC-869 Latin/Greek		JIS flavored Unicode
	PC-8 Latin/Greek		Windows 3.1J DBCS
	Varityper/Ancient Greek		JIS C 6226-1978
	Epson Latin/Greek based on 437		JIS X 0208-1983
	Adobe Greek		UNIX Cyrillic set
	reserved 33-255 for future Asian		Illuminae Kyrillikae
	reserved 33-255 for future		JIS Kanji 6226-1983 Level 1

	Asian		
	Hebrew-7		JIS Kanji 6226-1983 Level 2
	ISO 8859/8 Latin/Hebrew		Line Draw-7
	Hebrew-8		HP Block Characters
	Windows 3.1 Latin/Hebrew		Tax Line Draw
	Varityper/Kivun Hebrew		Line Draw-8
	PC-862 Latin/Hebrew		Ventura ITC Zapf Dingbats
	KS C 5601 (1993 Korean)		PS ITC Zapf Dingbats
	KS C 5601 (1987 Korean)		ITC Zapf Dingbats Series 100
	1st row Korean KS C 5601 (1987		ITC Zapf Dingbats Series 200
	94th row Korean KS C 5601 (1987		ITC Zapf Dingbats Series 300
	-1st set KS C 5601 1993		Windows 3.1 Baltic
	-114th set KS C 5601 1993		Carta
	ISO 15: Italian		Ornaments
	reserved 33-255 for future Asian		Universal News & Commercial Pi
	reserved 33-255 for future Asian		Chess
	Studio Script		Astrology 1

	Symbol Set Name		Symbol Set Name
	Astrology 2		Logos: Company 2
	Pi Set #0		Logos: Company 3
	Pi Set #1		Logos: Company 4
	Pi Set #2		Logos: Company 5
	Pi Set #3		Logos: Company 6
	Pi Set #4		Logos: Company 7
	Pi Set #5		Logos: Company 8
	Pi Set #6		Logos: Company 9
	Pi Set #7		Logos: Company 10
	Pi Set #8		Logos: Company 11
	Pi Set #9		Logos: Company 12
	Animals		Logos: Company 13

	Astrology 1		Logos: Company 14
	Astrology 2		Logos: Company 15
	Astrology 3		Logos: Company 16
	Borders & Ornaments 1		Logos: Company 17
	Borders & Ornaments 2		Logos: Company 18
	Borders & Ornaments 3		Logos: Company 19
	Borders & Ornaments 4		Logos: Company 20
	Borders & Ornaments 5		Logos: Company 21
	Borders & Ornaments 6		Logos: Company 22
	Business & Services 1		Logos: Company 23
	Business & Services 2		Logos: Company 24
	Business & Services 3		Logos: Company 25
	Commercial 1		Logos: Company 26

	Commercial 2		Logos: Company 27
	Communication 1		Logos: Company 28
	Communication 2		Logos: Company 29
	Communication 3		Logo: Services 1
	Communication 4		Logo: Services 2
	Communication 5		Logo: Services 3
	Communication 6		Math & Technical 1
	Communication 7		Math & Technical 2
	Communication 8		Math & Technical 3
	Credit Cards		Math & Technical 4
	Ecology		Math & Technical 5
	Games & Sports 1		Math & Technical 6
	Games & Sports 2		Math & Technical 7
	Games & Sports 3		Math & Technical 8

	Games & Sports 4		Math & Technical 9
	General Symbols 1		Math & Technical 10
	General Symbols 2		Math & Technical 11
	General Symbols 3		Math & Technical 12
	General Symbols 4		Math & Technical 13
	General Symbols 5		Math & Technical 14
	Holidays 1		Math & Technical 15
	Industry & Engineering 1		Math & Technical 16
	Industry & Engineering 2		Math & Technical 17
	International Symbols 1		Medical & Pharmaceutical 1
	International Symbols 2		Medical & Pharmaceutical 2
	Legal Trademarks		Military 1
	Logos: Company 1		Military 2

	Symbol Set Name		**Symbol Set Name**
	Musical		Lucida Stars
	Numerics 1		Wingdings
	Numerics 2		Wingdings 2
	Numerics 3		Wingdings 3
	Numerics 4		Math-7
	Numerics 5		Tech-7
	Numerics 6		PS Math
	Numerics 7		Ventura Math
	Numerics 8		Math-8
	Numerics 9		Universal Greek & Math Pi
	Numerics 10		TEX Math Extension
	Numerics 11		TEX Math Symbol

	Phonetics 1		$\TeX$ Math Italic
	Phonetics 2		Lucida Bright Math Extension
	Phonetics 3		Lucida Bright Math Symbol
	Religious		Lucida Bright Math Italic
	Seals 1		Adobe Symbol
	Seals 2		Math Pi Set #0
	Special Alphabets 1		Math Pi Set #1
	Special Alphabets 2		Math Pi Set #2
	Special Alphabets 3		Math Pi Set #3
	Special Alphabets 4		Math Pi Set #4
	Special Alphabets 5		Math Pi Set #5
	Special Alphabets 6		Math Pi Set #6
	Special Alphabets 7		Math Pi Set #7

	Special Alphabets 8		Math Pi Set #8
	Special Alphabets 9		Math Pi Set #9
	Special Alphabets 10		ISO 8859/1 Latin 1
	Special Alphabets 11		ISO 8859/2 Latin 2
	Television 1		ISO 8859/3 Latin 3
	Television 2		ISO 8859/4 Latin 4
	Transportation 1		ISO 8859/9 Latin 5
	Transportation 2		ISO 8859/10 Latin 6
	Varityper Logos		ISO 8859/5 Latin/Cyrillic
	Varityper Logos		ISO 8859/6 Latin/Arabic
	Varityper Logos		ISO 8859/7 Latin/Greek
	Varityper Logos		ISO 10646-1
	Varityper Logos		OCR-A

	Varityper Logos		OCR-B
	Varityper Logos		OCR-M
	Varityper Logos		MICR (E13B)
	Varityper Logos		International Phonetic Alph.
	Varityper Logos		American Phonetic Alphabet
	Varityper Logos		Typewriter Paired APL
	Varityper Logos		Bit Paired APL
	Varityper Logos		Premier
	Varityper Logos		Premier Subset
	Varityper Logos		Expert
	Varityper Logos		Alternate
	Varityper/Marking Numbers-Sqrs		Fraktur
	Lucida Arrows		Phonetic Pi Set
	Lucida Icons		Xerox Prudential Sig. 1

	Symbol Set Name		Symbol Set Name
	Xerox Prudential Sig. 2		Legal
	reserved (Specials)		ISO 2: International Reference
	Cyrillic ASCII (8859/5-1986)		HPL
	Cyrillic		OEM-1
	PC Cyrillic		Roman-8
	Windows 3.1 Latin/Cyrillic		Windows 3.0 Latin 1
	PC Latin/Cyrillic (don't use)		PC-8, Code Page 437
	Extended Cyrillic (OEM)		PC-8 D/N, Danish/Norwegian
	USSR Gost (Russian, OEM)		PC-850, Multilingual
	Bulgarian based on PC-850		Pi Font
	ISO 11: Swedish		PC Latin 5 (not PC Latin/Turkish)
	HP Spanish		PC-852, Latin 2
	ISO 17: Spanish		PC Latin 3
	ISO 10: Swedish		Windows 3.1 Latin 1
	ISO 16: Portuguese		PC-860 Portugal
	ISO 84: Portuguese		PC-861 Iceland

	ISO 85: Spanish		PC-863 Canada-French
	HP European Spanish		PC-865 Norway
	HP Latin Spanish		PC-775 Baltic
	HP-GL Download		special Latvian/Russian set
	HP-GL Drafting		special Lithuanian/Russian set
	HP-GL Special Symbols		special Baltic/Russian set
	Sonata		Arabic (Professor McKay's version)
	1st Row simplified Chinese GB		Arabic-8
	94th row simp. Chinese GB		Windows 3.1 Latin/Arabic
	Thai-8		Code Page 864 Latin/Arabic
	TISI 620-2533 (Thai)		Varityper/Traditional Arabic
	Windows 3.1 Latin 5		Varityper/Simplified Arabic
	Turkish-8		Varityper/Display Arabic
	PC-8 T, Turkish		Varityper/Traditional Farsi
	Teletex		Varityper/Simplified Farsi
	CNS 11643 (1986 Taiwan)		Varityper/Display Farsi
	Big 5 (Taiwan)		3 of 9 Barcode

	Windows 3.1 Latin/Thai		Industrial 2 of 5 Barcode
	TCA (Taiwan)		Matrix 2 of 5 Barcode
	Mixed Pi Set #0		Interleaved 2 of 5 Barcode
	Mixed Pi Set #1		CODABAR Barcode
	Mixed Pi Set #2		MSI/Plessey Barcode
	Mixed Pi Set #3		Code 11 Barcode
	Mixed Pi Set #4		UPC/EAN Barcode
	Mixed Pi Set #5		CMC-7 MICR
	Mixed Pi Set #6		USPS ZIP
	Mixed Pi Set #7		Varityper/Postal Barcodes
	Mixed Pi Set #8		Varityper/UPC Barcodes
	Mixed Pi Set #9		Varityper/Code39
	1st row Traditional Chinese Big 5		Agfa Interleaved 2 of 5 1996
	94th row Traditional Chinese Big 5		Agfa Interleaved 2 of 5 1861
	ISO 6: ASCII		Agfa Interleaved 2 of 5 1863

For convenience, the above table is repeated below with the symbol set names in alphanumeric order.

Symbol Set Name		Symbol Set Name	
1st row Korean KS C 5601 (1987		Chemical Pi Set #6	
1st set KS C 5601 1993		Chemical Pi Set #7	
1st Row simplified Chinese GB		Chess	
1st row Traditional Chinese Big 5		CMC-7 MICR	
3 of 9 Barcode		CNS 11643 (1986 Taiwan)	
94th row Korean KS C 5601 (1987		CODABAR Barcode	
94th row simp. Chinese GB		Code 11 Barcode	
94th row Traditional Chinese Big 5		Code Page 864 Latin/Arabic	
114th set KS C 5601 1993		Commercial 1	
Adobe Cyrillic		Commercial 2	
Adobe East European		Communication 1	
Adobe Greek		Communication 2	
Adobe Symbol		Communication 3	

Agfa Interleaved 2 of 5 1861		Communication 4	
Agfa Interleaved 2 of 5 1863		Communication 5	
Agfa Interleaved 2 of 5 1996		Communication 6	
Alternate		Communication 7	
Alternate Caps		Communication 8	
American Phonetic Alphabet		Credit Cards	
Animals		Cyrillic	
Arabic (Professor McKay’s version)		Cyrillic ASCII (8859/5-1986)	
Arabic-8		DeskTop	
Astrology 1		Devanagari	
Astrology 1		Document	
Astrology 2		Ecology	
Astrology 2		Engravers	
Astrology 3		Engravers Subset	

Big 5 (Taiwan)		Epson Latin/Greek based on 437	
Bit Paired APL		Expert	
Borders & Ornaments 1		Extended Cyrillic (OEM)	
Borders & Ornaments 2		Fractions	
Borders & Ornaments 3		Fraktur	
Borders & Ornaments 4		Games & Sports 1	
Borders & Ornaments 5		Games & Sports 2	
Borders & Ornaments 6		Games & Sports 3	
Bulgarian based on PC-850		Games & Sports 4	
Business & Services 1		GB 2312 (1980 mainland China)	
Business & Services 2		General Symbols 1	
Business & Services 3		General Symbols 2	
Carta		General Symbols 3	
Chemical Pi Set #0		General Symbols 4	

Chemical Pi Set #1		General Symbols 5	
Chemical Pi Set #2		Greek-8	
Chemical Pi Set #3		Hebrew-7	
Chemical Pi Set #4		Hebrew-8	
Chemical Pi Set #5			

Symbol Set Name		**Symbol Set Name**	
Holidays 1		ISO 8859/7 Latin/Greek	
HP Block Characters		ISO 8859/8 Latin/Hebrew	
HP European Spanish		ISO 8859/9 Latin 5	
HP German		ITC Zapf Dingbats Series 100	
HP Large Characters		ITC Zapf Dingbats Series 200	
HP Latin Spanish		ITC Zapf Dingbats Series 300	
HP Spanish		JIS 208 (1990)	
HP-GL Download		JIS C 6226-1978	
HP-GL Drafting		JIS flavored Unicode	
HP-GL Special Symbols		JIS Kanji 6226-1983 Level 1	
HPL		JIS Kanji 6226-1983 Level 2	
Illuminae Kyrillikae		JIS X 0208-1983	
Industrial 2 of 5 Barcode		Kana-8 (JIS 210)	
Industry & Engineering 1		Kanji Ward #0 - Bitstream	
Industry & Engineering 2		Kanji Ward #99 - Bitstream	
Interleaved 2 of 5 Barcode		Korean-8	
International Phonetic Alph.		KS C 5601 (1987 Korean)	

International Symbols 1		KS C 5601 (1993 Korean)	
International Symbols 2		Legal	
ISO 10: Swedish		Legal Trademarks	
ISO 10646-1		Line Draw-7	
ISO 11: Swedish		Line Draw-7 (same as 0L)	
ISO 13: Katakana		Line Draw-8	
ISO 14: JIS ASCII		Lining Figures	
ISO 15: Italian		Logo: Services 1	
ISO 16: Portuguese		Logo: Services 2	
ISO 17: Spanish		Logo: Services 3	
ISO 2: International Reference		Logos: Company 1	
ISO 21: German		Logos: Company 10	
ISO 25: French (obsolete)		Logos: Company 11	
ISO 4: United Kingdom		Logos: Company 12	
ISO 57: Chinese		Logos: Company 13	

ISO 6: ASCII		Logos: Company 14	
ISO 60: Danish/Norwegian		Logos: Company 15	
ISO 61: Norwegian Version 2		Logos: Company 16	
ISO 69: French		Logos: Company 17	
ISO 84: Portuguese		Logos: Company 18	
ISO 85: Spanish		Logos: Company 19	
ISO 8859/1 Latin 1		Logos: Company 2	
ISO 8859/10 Latin 6		Logos: Company 20	
ISO 8859/2 Latin 2		Logos: Company 21	
ISO 8859/3 Latin 3		Logos: Company 22	
ISO 8859/4 Latin 4		Logos: Company 23	
ISO 8859/5 Latin/Cyrillic		Logos: Company 24	
ISO 8859/6 Latin/Arabic		Logos: Company 25	

Symbol Set Name		Symbol Set Name	
Logos: Company 26		Math Pi Set #8	
Logos: Company 27		Math Pi Set #9	
Logos: Company 28		Math-7	
Logos: Company 29		Math-7 (same as 0M)	
Logos: Company 3		Math-8	
Logos: Company 4		Matrix 2 of 5 Barcode	
Logos: Company 5		MC Text	
Logos: Company 6		Medical & Pharmaceutical 1	
Logos: Company 7		Medical & Pharmaceutical 2	
Logos: Company 8		MICR (E13B)	
Logos: Company 9		Microsoft Publishing	
Lotus LICS set		Military 1	
Lotus LMBCS Group 0		Military 2	

Lotus LMBCS Set		Mixed Pi Set #0	
Lucida Arrows		Mixed Pi Set #1	
Lucida Bright Math Extension		Mixed Pi Set #2	
Lucida Bright Math Italic		Mixed Pi Set #3	
Lucida Bright Math Symbol		Mixed Pi Set #4	
Lucida Icons		Mixed Pi Set #5	
Lucida Stars		Mixed Pi Set #6	
Math & Technical 1		Mixed Pi Set #7	
Math & Technical 10		Mixed Pi Set #8	
Math & Technical 11		Mixed Pi Set #9	
Math & Technical 12		MSI/Plessey Barcode	
Math & Technical 13		Musical	
Math & Technical 14		Numerics 1	
Math & Technical 15		Numerics 10	

Math & Technical 16		Numerics 11	
Math & Technical 17		Numerics 2	
Math & Technical 2		Numerics 3	
Math & Technical 3		Numerics 4	
Math & Technical 4		Numerics 5	
Math & Technical 5		Numerics 6	
Math & Technical 6		Numerics 7	
Math & Technical 7		Numerics 8	
Math & Technical 8		Numerics 9	
Math & Technical 9		OCR-A	
Math Pi Set #0		OCR-B	
Math Pi Set #1		OCR-M	
Math Pi Set #2		OEM-1	
Math Pi Set #3		Old Style Figures	

Math Pi Set #4		Ornaments	
Math Pi Set #5		PC Cyrillic	
Math Pi Set #6		PC Latin 3	
Math Pi Set #7		PC Latin 5 (not PC Latin/Turkish)	

Symbol Set Name		Symbol Set Name	
PC Latin/Cyrillic (don't use)		Small Caps and Old Style Figures	
PC-1004		Sonata	
PC-775 Baltic		Special Alphabets 1	
PC-8 D/N, Danish/Norwegian		Special Alphabets 10	
PC-8 Latin/Greek		Special Alphabets 11	
PC-8 T, Turkish		Special Alphabets 2	
PC-8, Code Page 437		Special Alphabets 3	
PC-850, Multilingual		Special Alphabets 4	
PC-851 Latin/Greek		Special Alphabets 5	
PC-852, Latin 2		Special Alphabets 6	
PC-860 Portugal		Special Alphabets 7	
PC-861 Iceland		Special Alphabets 8	
PC-862 Latin/Hebrew		Special Alphabets 9	

PC-863 Canada-French		special Baltic/Russian set	
PC-865 Norway		special Latvian/Russian set	
PC-869 Latin/Greek		special Lithuanian/Russian set	
Phonetic Pi Set		Studio Script	
Phonetics 1		Swash Characters	
Phonetics 2		Tax Line Draw	
Phonetics 3		TCA (Taiwan)	
Pi Font		Tech-7	
Pi Set #0		Teletex	
Pi Set #1		Television 1	
Pi Set #2		Television 2	
Pi Set #3		TEX Math Extension	
Pi Set #4		TEX Math Italic	
Pi Set #5		TEX Math Symbol	
Pi Set #6		Thai-8	
Pi Set #7		TISI 620-2533 (Thai)	

Pi Set #8		Transportation 1	
Pi Set #9		Transportation 2	
Premier		Turkish-8	
Premier Subset		Typewriter Paired APL	
PS ISO Latin 1		Universal Greek & Math Pi	
PS ITC Zapf Dingbats		Universal News & Commercial Pi	
PS Math		UNIX Cyrillic set	
PS Text		UPC/EAN Barcode	
Religious		USPS ZIP	
reserved (Specials)		USSR Gost (Russian, OEM)	
reserved 33-255 for future Asian		Varityper Logos	
reserved 33-255 for future Asian		Varityper Logos	
reserved 33-255 for future Asian		Varityper Logos	
reserved 33-255 for future Asian		Varityper Logos	
reserved 33-255 for future Asian		Varityper Logos	

reserved 33-255 for future Asian		Varityper Logos	
Roman Extension		Varityper Logos	
Roman-8		Varityper Logos	
Seals 1		Varityper Logos	
Seals 2		Varityper Logos	
Small Caps and Lining Figures		Varityper Logos	

Symbol Set Name		Symbol Set Name	
Varityper Logos		Ventura Math	
Varityper Logos		Ventura US	
Varityper Logos		Windows 3.0 Latin 1	
Varityper Logos		Windows 3.1 Baltic	
Varityper Logos		Windows 3.1 Latin 1	
Varityper/Ancient Greek		Windows 3.1 Latin 2	
Varityper/Code39		Windows 3.1 Latin 5	
Varityper/Display Arabic		Windows 3.1 Latin/Arabic	
Varityper/Display Farsi		Windows 3.1 Latin/Cyrillic	
Varityper/Kivun Hebrew		Windows 3.1 Latin/Greek	
Varityper/Marking Numbers-Sqrs		Windows 3.1 Latin/Hebrew	
Varityper/Postal Barcodes		Windows 3.1 Latin/Thai	
Varityper/Simplified Arabic		Windows 3.1J DBCS	
Varityper/Simplified Farsi		Wingdings	

Varityper/Traditional Arabic		Wingdings 2	
Varityper/Traditional Farsi		Wingdings 3	
Varityper/UPC Barcodes		Xerox Prudential Sig. 1	
Ventura International		Xerox Prudential Sig. 2	
Ventura ITC Zapf Dingbats			

NOTE: The *WP PS Character* symbol set (ID=20J) is HP Confidential.

Font Spacing (Primary) *Esc (s # p/P*
Font Spacing (Secondary) *Esc) s #p/P*

Value(#)	=	0	Fixed Spacing
	=	1	Proportional Spacing
	=	2	Dual-Fixed Spacing
Default		=	0
Range		=	0 to 2

If proportional spacing is specified and a proportional font is unavailable in the requested symbol set, a fixed spacing font with the current pitch specification is chosen. If fixed spacing is requested but unavailable, proportional is chosen. If dual-fixed spacing is requested but unavailable, a fixed spacing font with the current pitch is chosen.

For fixed space bitmap fonts, both pitch and height are used for selection of font character size. For proportional bitmap and scalable fonts, only height is used for selection.

Dual-fixed pitch spacing is designed for fonts that have full-width characters with one spacing and half-width characters with a different spacing. The *nominal space* is the largest width for a space character in a dual-fixed pitch font. CMI (*Esc&k#H*) directly affects the nominal space of the font. The other spacings are linearly scaled according to the current CMI value: character widths are multiplied by the ratio of the CMI to nominal width.

Font Pitch (Primary) *Esc (s # h/H*
Font Pitch (Secondary) *Esc) s # h/H*

Designates the horizontal spacing of a fixed-space (bitmap or scalable) font in characters per inch (cpi).

Value(#)	=	Pitch
Default	=	10 cpi
Range	=	> 0.00 (fractional values are allowed; valid to 2 decimal places)

If the exact pitch is unavailable, the next larger pitch is selected. If a larger pitch is unavailable, the closest smaller pitch is selected.

Pitch is ignored when selecting proportional fonts, but saved in the font select table and available when a fixed space font is selected.

For fixed space bitmap fonts, both pitch and height are used for font selection. For proportional scalable fonts, only height is used for selection.

INTELLIFONT FIXED PITCH TO POINT SIZE CONVERSION: The pitch of a fixed-space scalable font is converted to a corresponding point size (height. Fonts are scaled in quarter-point increments. The formula for converting pitch to point size for scalable fixed-pitch fonts is:

$$\frac{1}{\text{Height} = (\text{Desired Pitch})(\text{Master Design Pitch in Font Header} \div \text{Scale Factor})(0.01383)}$$

The resulting height value is then plugged into the range from .25 point to 999.75 point, with the closest point size being selected.

TRUETYPE FIXED PITCH TO POINT SIZE CONVERSION: Truetype scales to a requested point size by first rounding the number of pixels per em (ppm) requested. The following calculations are necessary to arrive at the correct point size from a user-requested pitch, and at the same time account for the ppem rounding.

First calculate the user units per em (units of measure per em) and truncate the result:

$$\text{UUpem} = \text{TRUNC}\left[\left(\frac{\text{UserUnits} \times \text{ScaleFactor}}{\text{UserPitch}}\right) \div \text{Pitch}\right]$$

Next, compute the real point size:

$$\text{real ps} = \text{ROUND}\left[\frac{72\ (\text{pts})}{1\ (\text{in})} \times \frac{\text{UUpem}\ (\text{UserUnits})}{1\ (\text{em})} \times \frac{1\ (\text{in})}{\text{UserUnits}} \times \frac{16\ (16\text{ ths})}{1\ (\text{pts})}\right]$$

DEVICE NOTE: On LJIIIs and above, all fonts within .05 cpi of the requested pitch are treated as the same pitch.

Font Height (Primary) *Esc (s # v/V*
Font Height (Secondary) *Esc) s # v/V*

Specifies font height in points (1/72.0 inch).

Value(#)	=	Height in points
Default	=	12 points
Range	=	> 0.00 (fractional values are allowed; valid to 2 decimal places)

If the requested height is unavailable, the closest heights are chosen for the next selection criteria. The closest value is in terms of absolute difference: e.g., if 6, 8, and 12 point fonts are available and 10 is requested, both 8 and 12 point fonts are considered.

Fractional values may be specified when requesting height. All bitmap fonts within 0.25 points of the closest absolute difference are considered. If a 12 point font is requested, fonts between 11.75 and 12.25 are considered. Or, to put it another way, if 6, 8, and 11.75 point fonts are available and 10 is requested, both the 8 and 11.75 fonts are considered.

Height is ignored when selecting a fixed space scalable font; but the value is saved in the font select table and available when a proportional font is selected.

NOTE: The Height command should be sent whenever selecting a scalable proportional font with an ID number; otherwise, the current height in the font select table determines the point size.

DEVICE NOTE: For scalable fonts on LJIIIs and above, the value is from .25 to 999.75 points in increments of 0.25 point.

DEVICE NOTE: If no font within .25 points is available, DeskJets below 1200 select the next smaller height. If no smaller height is available, the next larger height is selected. When two fonts have heights that are equidistant from the designated height, the smaller is chosen.

Font Style (Primary) *Esc (s # s/S*
Font Style (Secondary) *Esc) s # s/S*

Identifies the posture, width, and structure of the font symbols. The partial sums for posture, width, and structure are added together to determine the desired value (#). The composition of the style word is shown below.

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

15	14 – 10	9 – 5	4 – 2	1 – 0
X	reserved	structure	width	posture

Value(#)	=	**Posture** (style word partial sum)		
		0	-	Upright
		1	-	Italic
		2	-	Alternate Italic
		3	-	Reserved
	=	**Width** (style word partial sum multiplied by 4)		
		0	-	Normal
		1	-	Condensed
		2	-	Compressed or extra condensed
		3	-	Extra compressed
		4	-	Ultra compressed
		5	-	Reserved
		6	-	Extended or expanded
		7	-	Extra extended or extra expanded
	=	**Structure** (style word partial sum multiplied by 32)		
		0	-	Solid
		1	-	Outline
		2	-	Inline
		3	-	Contour, Edge effects
		4	-	Solid with shadow
		5	-	Outline with shadow
		6	-	Inline with shadow
		7	-	Contour with shadow
		8-11	-	Patterned (complex patterns, subjective to typeface)
		12-15	-	Patterned with shadow
		16	-	Inverse
		17	-	Inverse in open border
		18-30	-	Reserved
		31	-	Unknown structure

Default = 0
Range = 0 to 32767 (values greater than 32767 are clamped)

DEVICE NOTE: Pre-DJ 500 and pre-LJII printers cannot "see" bits 8-15.

The reserved bits (10 to 15) should be set to 0.

MATCHING ALGORITHM

The following procedure matches requested styles with available styles:

1. Printers recognizing only style values 0, 1, and 2 discard requests for larger values.

2. Printers recognizing style values from 0 to 255 convert requests for larger values to 255. The request is discarded if an exact match is not found after conversion.

3. An exact match is required for style selection. If the requested value is within the range of the printer and a match is not made, the request is ignored, but saved in the font select table, available for the next selection.

Whenever a requested font parameter is ignored, the parameter is retained in the font select table for future font selections; and the current font selection process continues as if the parameter had never been requested.

In some PCL machines, the style word may be an operator: slanting, condensing, expanding, outlining, and shadowing operations are all theoretically possible. For example, if italic is unavailable, slant may be added to the upright face. No new command sequence is required to support such operations.

EXAMPLE

Assume a font style of "italic compressed contour" is desired. Then the value(#) would be

$$1 + (2 \times 4) + (3 \times 32) = 105$$

Font Stroke Weight (Primary) *Esc (s # b/B*
Font Stroke Weight (Secondary) *Esc) s # b/B*

Designates the thickness of the strokes that compose the characters of a font.

Value (#)	=	-7	Ultra thin
		-6	Extra thin
		-5	Thin
		-4	Extra light
		-3	Light
		-2	Demi-light
		-1	Semi-light
		0	Medium ("Book" or "Text" weight)
		1	Semi-bold
		2	Demi-bold
		3	Bold
		4	Extra bold
		5	Black
		6	Extra black
		7	Ultra black
Default	=	0	
Range	=	-7 to 7 (less than -7 maps to -7; greater than 7 maps to 7)	

If the designated stroke weight is unavailable and 0 or greater, the closest thicker weight is selected. If a thicker weight is unavailable, the closest thinner weight is selected.

If the designated stroke weight is unavailable and less than 0, the closest thinner weight is selected. If a thinner weight is unavailable, the closest thicker weight is selected.

NOTE: Many typefaces were designed for advertising, and a "medium" was used to describe the standard treatment. Later, additional treatments were designed for text use. Therefore, the typeface treatment designation "medium" may not always take a PCL value of 0. This value may be assigned to "book" or "text" instead.

DEVICE NOTE: DeskJets below 800 provide algorithmic bolding on fixed medium internal fonts when a value of 1 is used.

Font Typeface (Primary) *Esc (s # t/T*
Font Typeface (Secondary) *Esc) s # t/T*

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

CURRENT VERSION

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

DEVICE NOTE: LJs use only the lower 9 bits if the requested typeface value is less than 512.

15 12	11 0
Vendor	Typeface Family

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

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

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

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

A requested font parameter that is ignored is retained for future font selections; and the current font selection process continues as if the parameter had never been requested.

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

SINGLE-BYTE VERSION

Pre-DeskJet 500s and pre-LaserJet IIIs used only a 1-byte field (for a range of 0-255). Printers recognizing only values of 0-255 treat requests for larger numbers as 255.

LASERJET III / DESKJET 500 VERSION

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

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

MSB LSB

15	14 11	10 9	8 0
0	Vendor	Version	Typeface Base Value

Typeface Base Value

0	Line Printer or Line Draw
3	Courier
4	Helvetica
5	Times Roman
6	Letter Gothic

... (See the table below for a complete list)

Version (typeface word partial sum multiplied by 512)

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

Vendor (typeface word partial sum multiplied by 2048)

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

Default	3
Range	0 to 65535

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

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

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

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

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

The following is a list of typeface values as of 1/01/95. These typeface names may be registered trademarks of a third party. Use of these fonts may be conditional upon a license grant from the owners of the fonts. Hewlett-Packard makes no representation as to the quality or performance of the fonts, and any reference to the fonts does not grant license or right to use the fonts.

	Typeface		**Typeface**
	Line Printer		ITC Zapf Dingbats
	Elite		Cooper
	Courier		ITC Bookman
	Helvetica		Noparat (Thai)
	Times Roman		Stick
	Letter Gothic		HP-GL Drafting
	Script		HP-GL Spline
	Prestige		Gill Sans
	Caslon 540 & No. 3		Unesco (Thai)
	Caslon Antique (contour)		Univers
	Caslon Open Face (inline)		Bodoni
	Orator		Poster Bodoni (black)
	Presentation		Greek Apla
	Serifa		Rockwell
	Futura		Melior
	Greek Futura		ITC Tiffany
	Palatino		ITC Clearface
	ITC Souvenir		Amelia
	ITC Souvenir Greek		Park Avenue (italic)
	Optima		Falstaff (black)
	Safeer (Arabic)		Handel Gothic
	Komain (Thai)		Dom Casual
	Greek Oracle		ITC Benguiat
	ITC Garamond		ITC Cheltenham

	Coronet (italic)		Century Expanded
	Chevalier (bold expanded pattern 0)		Franklin Gothic
	Broadway		Paetai (Thai)
	Century Schoolbook		Plantin
	Greek & Math Serif		Trump Mediaeval
	University Roman		Futura Black
	ITC Korinna		ITC American Typewriter
	Naskh		Antique Olive
	Cloister Black		Greek Antique Olive
	ITC Galliard		Uncial
	ITC Avant Garde Gothic		ITC Bauhaus
	Tom (Thai)		Century Old Style
	Brush (italic)		ITC Eras
	Stop		Friz Quadrata (ITC)
	Blippo (black)		ITC Lubalin Graph
	Tea Chest (condensed)		Eurostile
	Hobo		Intanon (Thai)
	Windsor		Greek Microstyle
	Peignot		Mincho (Japanese)
	Baskerville		Myoungjo (Korean)
	Trade Gothic		ITC Serif Gothic
	Pemai (Thai)		Saemmul (Korea)
	CG Trade		Sammul (Korea)
	Goudy Old Style		Snell Roundhand
	ITC Zapf Chancery		Pilgy (Korean)
	Clarendon		Souvenir Gothic

	Typeface		**Typeface**
	Stymie		Post Antiqua
	Bernhard Modern		Aerospace Pi
	Excelsior		Devanagari (Hindi)
	Gando Ronde Script		Maritime Pi
	Ondine		Krishna (Gujarati)
	EACT (Thai)		Bits Pic Pi
	P. T. Barnum		Ranjit (Gurmukhi)
	Kaufmann		Keycap Pi
	U-Thong (Thai)		Raj Raja (Tamil)
	ITC Bolt (extended)		Tieman
	ITC Machine (condensed)		Gyosho
	Revue		David
	Garamond (Stempel)		Nork
	Garth Graphic		Ousbouh
	ITC Ronda		Koufi
	OCR-A		Italia (ITC)

	Cochin		Hadassah
	Englische Schreibschrift (italic)		Bembo
	Mister Earl (condensed)		Sharif
	Flash (italic)		Aachen
	Woodstock		Malik
	Gothic (numbered)		Americana
	Stencil (ATF)		Arnold Boecklin
	OCR-B		Copperplate Gothic (text)
	Akzidenz-Grotesk		Belwe
	Black White (patterned, outline, inline)		ITC Berkeley Oldstyle
	Logos		Frutiger
	Shannon		Candida
	ITC Stone Informal		Folio
	ITC Stone Sans		Corona
	ITC Stone Serif		ITC Kabel
	Schneidler Mediaeval		Zeppelin (inline)
	ITC Symbol		Garamond No. 3

	ITC Weidemann		Sabon
	Copperplate Gothic (display)		ITC Novarese
	Trajan		Weiss
	Concorde		Hiroshige
	Janson Text		French Script
	Linotype Centennial		Meridien
	Life		Mistral
	Minister		Aster
	New Century Schoolbook		Caledonia
	Maru Gosikku (round gothic Japan)		Nuptial Script
	Gosikku (Kaku, gothic Japan)		Lucida
	Gothic (Japan)		Song (China)
	Socho		Adobe Wood Series 1
	Kyokasho (text book)		Memphis
	Kaisho		Lucida Sans
	Traditional Arabic Script		Syntax
	Arabic News		Utopia

	Typeface		Typeface
	Berthold Walbaum Buch		Communication 6
	Minion		Modern
	Marigold		PL Modern
	ITC Tiepolo		Games & Sports 1
	Versailles		Artistik
	ITC Leawood		Games & Sports 2
	ITC Caslon No. 224		Flintstones
	ITC Cushing		Games & Sports 3
	ITC Fenice		SnowCap
	ITC Usherwood		Games & Sports 4
	ITC Benguiat Gothic		Bedrock
	Spartan		Holidays 1
	ITC Ozwald (fatface)		Star Fleet
	Neuzeit Grotesk		Industry & Engineering 1
	PMN Caecilia		Star Trek Film
	ITC Busorama		Industry & Engineering 2

	Agfa Wile Roman		Star Trek
	ITC Zapf International		Transportation 1
	Poppl-Pontifex		Hei (China)
	ITC Quay Sans		Star Trek Pi
	Arial		Transportation 2
	Fairfield		ITC Mendoza
	ITC Zapf Book		Boton
	Lucida Casual		Jaeger Daily News
	Linotype Technical Pi 1 & 2		ITC Officina Serif
	Graphite		ITC Officina Sans
	Linotype Textil Pi 1 & 2		Goudy Modern
	Poetica		Scotch Roman
	Century Schoolbook Monospace		Temporary-Only Font
	Berliner Grotesk		-reserved
	Christiana		Bar Codes
	Comenius-Antiqua		Hadriano
	Delta		Joanna

	Italian Old Style		Onyx
	Zingo		Cyrillic Helvetica
	Octavian		Greek Helvetica
	Borders & Ornaments 1		East Asian Helvetica
	Footlight		Cyrillic Times
	Borders & Ornaments 4		Greek Times
	Apollo		East Asian Times
	Borders & Ornaments 5		ITC Quorum
	Bremen		Engravers' Old English
	Borders & Ornaments 6		Kennerley
	Oranda		Adobe Caslon
	Communication 1		Albertus
	Nubian		New Aurora Grotesque
	Communication 2		TBG Omnia
	Cataneo		Glypha
	Communication 3		Tempo
	Wittenberger Fraktur		Umbra (open shadow)

	Typeface		Typeface
	American Text		Letraset Bramley
	Pasquale		Isabella
	ITC Elan		Cascade Script
	Monotype Goudy Sans		VAG Rounded
	Lutheresche Fraktur		Russell Square
	Universal News & Commercial Pi		Liberty
	Thunderbird (extra condensed)		ITC Esprit
	ITC Honda (black)		Clairvaux
	Shelley		Raphael
	Mr. Big		ITC Franklin Gothic
	Macbeth		Murray Hill
	Universal Greek & Math Pi		Baker Signet
	ITC Century		Mythos
	Vineta		Gambling Pi
	TBG Duc de Berry		San Marco
	Times Europa		Typo Roman

	ITC Jamille		Engravers Text (inline)
	Flyer		New Berolina (italic)
	Wedding Text		Orbit-B
	Carolina		McCollough
	Avenir		ITC Isadora
	Lucia		Giddyup
	Tekton		Audio Pi
	Charme		Letraset Crillee
	ITC Flora		Agfa Nadianne
	Basilica		Compliment
	Auriol		ITC Giovanni
	Kuenstler Script		Neuzeit S
	ITC New Baskerville		Erbar
	Berling		Parisian
	News Gothic		Nofret
	Critter		City
	Linotype Holiday Pi 1, 2, & 3		Old Style 7

	Medici Script		Bell Centennial
	Aurora		Lydian
	Carta		Monotype Ellington
	Adobe Symbol		Impressum
	Insignia		Reporter No. 2
	Perpetua		Freestyle Script
	Raleigh		Serpentine
	Romic		Lithos
	Formata		Basilia
	Cyrillic Univers		Simplified Arabic
	Chuan Pim (like Univers)		Maximus
	Narkis Tam (like Univers)		ITC Slimbach
	Greek Univers II		Berthold Garamond
	Bauer Bodoni		Rad
	Industria		Land Pi
	Cutout		Oxford (italic)
	Decoration Pi		Kino (bold condensed)

	Typeface		Typeface
	Looney Tunes		Akzidens Grotesk Buch Schulbuch
	E13B MICR		Bookman
	Imperial		Bruce Old Style
	CMC-7 MICR		Bulmer
	Charlemagne		Madison
	Present Script		Textype
	Repro Script (italic)		Primer
	Matura (bold)		Garamond (Simoncini)
	Baskerville No. 2		Adobe Wood Series 2
	Engravers' Roman		Rotis Serif
	VGC Egyptian 505		Caravan LH One
	TBG Herculanum		Rotis Semiserif
	Clearface Gothic		Caravan LH Two
	Studz		Rotis Sans Serif
	Border Pi 1515-9		Caravan LH Three
	Toolbox		Rotis Semisans

	Bundesbahn Pi		Caravan LH Four
	Quake		Arcadia
	Chemical Pi		ITC Veljovik
	Neuland		Armenian Aramian
	Warning Pi		Armenian Barz
	Harry		Helvetica Rounded
	Alternate Gothic (numbered)		Olympian
	Figaro		DIN Engschrift (condensed)
	Formal Script		DIN Mittelschrift
	Holland Title		Granjon
	ITC Barcelona		Guardi
	Cartier		Impact
	Deepdene		Sassoon Primary
	Delphin		Packard
	Parsons		Baskerville Book
	Brighton		ITC Pacella
	Berthold Barmeno		Rusticana

	Berthold Colossalis		Eccentric
	Berthold Cosmos		Embassy Script
	ITC Isbell		Greek Florentine Script II
	ITC Mixage		PL Latin Bold
	Sonata		PL Latin Elongated (condensed)
	Badr, or Bayaan II		Latin Antique
	ITC Newtext		Latin Wide (extended)
	Happening		ITC Modern 216
	Menue		Serlio
	Doric		Piranesi
	S'maragd		Imago
	Pierrot		Wilke
	Ornaments		Cyrillic 22
	Berthold Bodoni Old Face		Adobe Garamond
	Schadow		Seagull
	Akzidens Grotesk Buch		Latin MT
	Akzidens Grotesk Buch Stencil		Runic MT

	Typeface		**Typeface**
	Moore Computer		Catull
	Commercial Script		Cremona
	Dominante		Audrey No. 2
	Wilhelm Klingspor Gotisch		Lo-Type
	Trajanus		Madame (patterned with shadow)
	TSI Caxton		Roundy
	Letraset Caxton		Animals
	Fette Fraktur		Ruling Script
	Sapphire (pattern 0)		Business & Services 1
	Saphir (pattern 0)		Sho
	Rainbow Bass (pattern 0)		Business & Services 2
	European Pi		Wiesbaden Swing
	Banco		Commercial 1
	Bodoni Antiqua		Star Trek Next
	Sallwey Script		Commercial 2
	Mathematical Pi		ITC Highlander

	Congress		Ecology
	Cheq		Helios II
	Berthold Walbaum Buch (B.metrics)		General Symbols 1
	Huxley Vertical		Kai Medium
	Grayda		General Symbols 2
	Penfield No. 3		Medical & Pharmaceutical 1
	Michelangelo		Space
	Neo Didot		Musical
	Berthold Caslon Buch		Special Alphabets 4
	Sans No. 1		Special Alphabets 5
	Torino		Special Alphabets 6
	Photina		Inflex
	Calligraphiques		Monotype Old Style
	Concorde Nova		Ming
	Franco		FangSong
	Goudy Text		Helinda Rook
	Balloon (italic)		Original Script

	Eusebius		Citadel Script
	Eusebius Open (inline)		Old Fashion Script
	Digital		ITC Legacy Serif
	Noris Script (italic)		ITC Legacy Sans
	Poppl-Pontifex (B.metrics)		Athenaeum
	Amigo		Athenaeum Negative (pattern 0)
	Pelican (italic)		Athenaeum Positive (pattern 1)
	Visigoth (bold italic)		ITC Anna (condensed)
	Letraset Arta		ITC Beesknees (black)
	Post Mediaval		ITC Studio Script (italic)
	Adsans		ITC Mona Lisa Recut (inline)
	Ariadne		ITC Mona Lisa Solid (upright)
	Calligraphy		Sackers Square Gothic
	Didot		Sackers English Script
	Ashley Script (italic)		Heritage
	Ashley Crawford (bold)		Sackers Gothic
	Ashley Inline (inline)		Greek Helios II

	Typeface		**Typeface**
	Times (Ten, New, etc.)		Broadpen
	Berthold Script		Amazone
	Bernhard Tango (italic)		Frank Ruehl
	Castellar (inline)		Cloe
	Else		Discus
	Basque (condensed)		Myriad
	Palace Script (italic)		WTC Our Bodoni
	Centaur		Ideal Schreibschrift
	Fine Hand		Print
	Linotype Astrology Pi		Lucida Blackletter
	Sackers Roman		Lucida Calligraphy
	Kompakt (ultra black italic)		Data 70
	Monoline Script (italic)		Compacta (expanded)
	Othello (bold condensed)		Helvetica Inserat (condensed)
	Sackers Classic Roman		Lucida Handwriting
	Sackers Italian Script (italic)		Milestones

	Musketeer		Biffo
	Riviera (inline)		Calvert
	Poppl-Residenz		Cantoria
	Rotation		Dorchester Script
	Bank Gothic		Grotesque
	Delphian (inline)		Pepita
	Greeting Monotone		Vectora
	Sackers Antique Roman		Script Bold
	Schwabacher		Spectrum
	Egyptienne (condensed)		Boulevard
	Artisan Roman (inline)		Cheltenham
	Forte (bold italic)		De Vinne
	Burin Roman		London Text (inline)
	Burin Sans (light)		Profil (bold italic inline)
	Hellenic Wide (extended)		Imprint
	Thompson Quillscript		Allegro (bold italic)
	Kartoon		Engraver's Gothic (text)

	Classic Roman		Bernhard (bold condensed)
	AG Old Face		Eckmann (text)
	Lucian		Cloister Open Face (outline)
	Della Robbia		Davida (text)
	Libra		Klang (italic)
	Brody (bold upright)		Fry’s Baskerville
	Ad Lib (bold)		Metro
	Choc (black)		Mandate
	Handle Oldstyle		Star Trek Gen
	Roman		Virile
	Antique Roman		Bingham Script (text)
	Goudy Catalogue, addt’l Old Style faces		Block (bold)
	Goudy Handtooled (inline)		ITC Gorilla (text)
	Goudy Heavyface (black)		ITC Pioneer (outline shadow)
	Calligrapher		Ruzicka
	Lucida Bright		Bodoni Campanile
	Pi Collection		Linotype Modern

	Typeface		Typeface
	Monterey Script (italic)		Eclipse (pattern 0)
	Playbill (condensed)		Capone Light
	Normande		Victorian Silhouette (contour)
	Wave		Dynamo (extra bold)
	Bernhard Fashion (extra light)		Modernistic (inline)
	Mercurius		Gallia (inline)
	Stuyvesant (inline)		Skjald
	Impuls (italic)		Bell Gothic
	Romana (text & bold)		Gillies Gothic Bold (italic)
	Shotgun		Quaint Roman
	Ehrhardt		Chic (inline)
	ITC Grizzly		PL Westerveldt Light (condensed)
	ITC Grouch		PL Davison Americana
	ITC Tom's New Roman		TC Jasper
	Palette (italic)		Poppl-Laudatio
	Hanseatic (ultrabold condensed)		TC Europa Bold

	Bison		Siena Black (italic)
	Jefferson		Yearbook
	Electra		Koloss (extra bold)
	Antique No. 3		Phenix American (extra condensed)
	Flemish Script (italic)		PL Bernhardt
	Hallmark Bodoni		Orlando Caps
	Modern #20		PL Barclay Outline (outline)
	Westinghouse Gothic		PL Britannia Bold
	Bloc (outline)		PL Fiorello Condensed
	Empire (ultra condensed)		Fluidum Bold (italic)
	Oscar		Woodblock (bold)
	Eagle Bold		Sinaloa (pattern 0)
	Joanna Solotype (inline)		Stratford Extra Bold
	Akzidenz-Grotesk (B.metrics)		Matra (pattern 0)
	Koch Antiqua		PL Tower Condensed
	Mirarae		Section Bold Condensed
	Horley Old Style		Miehle Condensed

	Tango		Phyllis
	Pifont Circle Numbers		Modernique (extra bold)
	Pifont OCRA Numbers		Egyptienne F
	Pifont Square Numbers		Post Antiqua (B.metrics)
	Pifont Triangle Numbers		Diotima
	Bank Script (italic)		Aldus
	Serlio Dekoration (pi numbers)		Chaplin (italic)
	Concorde (B.metrics)		Uncle Sam Stars (pattern 0, shadow)
	Jets		Uncle Sam Stripes (pattern 1, shadow)
	Jetsons		Wildstyle
	Looney Type		Logan (pattern 0)
	Pompeijana		Eon Age (pattern 1)
	Rusticana (Frutiger)		System X3 (pattern 2)
	Notre Dame		Galaxy Run (pattern 3)
	Beverly Hills (inline)		Jukebox (bold condensed)
	Lotus (pattern 0)		Marking Numbers Squares
	Advertisers Gothic Light		Al Harf Al Jadid

	Typeface		**Typeface**
	Vivaldi		Hollandse Mediaeval
	Codex		Holland Seminar
	Metronome Gothic (bold extra condensed)		CG Cloister
	Salut (bold)		Adroit
	Lucida Fax		Claire News
	Bellevue		Triplett
	Architect		Accolade
	Beton Extra Bold		Claridge
	Metropolis (extra bold)		Alpin Gothic
	PL Davison Zip Bold		Geometric
	Neon (Nebiolo)		Heldustry
	PL Benguiat Frisky		Busorama
	PL Bartuska Trophy Oblique		Salto
	Cable		Fehrle Display
	PL Brazilia		Kismet
	PL Radiant		Digi Fraktur

	Ritmo Bold (italic)		Anglia
	PL Fiedler Gothic Bold		Jiffy
	Egiziano Black		Rosewood
	Studio		Zebrawood
	PL Futura Maxi		Pepperwood (condensed)
	Solemnis		Copal (solid)
	Quirinus Bold (condensed)		Copal (outline, patterned)
	PL West Behemoth Semi Condensed (XBd Cd)		Motter Corpus (extrabold)
	Renault		Cerigo
	Forbes Bold		Caflisch Script
	Mobil		Mezz
	Becket		Nueva
	Lucida Sans Typewriter		Penumbra
	Cartoon Script Roman		Sanvito
	Campanula		Viva
	Odilia		Alexa (italic)
	Lino Letter		Balzano

	Henche		Caliban (condensed italic)
	Mahlau (condensed)		Ex Ponto
	Aquarias No. 8 (bold)		LiShu (China)
	CG Frontiera		Yuang (Yuan, XiYuang - China)
	Globe Gothic		Miryam
	Signature		Ryadh
	Sans Serif Stencil		Strider
	Boldface PS		Akzidenz Grotesk Buch Rounded
	Title PS		Isil Gothic
	Hess Neobold		Wingdings

For convenience, the above table is shown below with typefaces sorted in alphanumeric order.

Typeface		**Typeface**	
-reserved		Aster	
Aachen		Athenaeum	
Accolade		Athenaeum Negative (pattern 0)	
Ad Lib (bold)		Athenaeum Positive (pattern 1)	
Adobe Caslon		Audio Pi	
Adobe Garamond		Audrey No. 2	
Adobe Symbol		Auriol	
Adobe Wood Series 1		Aurora	
Adobe Wood Series 2		Avenir	
Adroit		Badr, or Bayaan II	
Adsans		Baker Signet	
Advertisers Gothic Light		Balloon (italic)	
Aerospace Pi		Balzano	
AG Old Face		Banco	
Agfa Nadianne		Bank Gothic	
Agfa Wile Roman		Bank Script (italic)	

Akzidens Grotesk Buch		Bar Codes	
Akzidens Grotesk Buch Schulbuch		Basilia	
Akzidens Grotesk Buch Stencil		Basilica	
Akzidenz Grotesk Buch Rounded		Baskerville	
Akzidenz-Grotesk		Baskerville Book	
Akzidenz-Grotesk (B.metrics)		Baskerville No. 2	
Al Harf Al Jadid		Basque (condensed)	
Albertus		Bauer Bodoni	
Aldus		Becket	
Alexa (italic)		Bedrock	
Allegro (bold italic)		Bell Centennial	
Alpin Gothic		Bell Gothic	
Alternate Gothic (numbered)		Bellevue	
Amazone		Belwe	
Amelia		Bembo	
American Text		Berliner Grotesk	
Americana		Berling	

Amigo		Bernhard (bold condensed)	
Anglia		Bernhard Fashion (extra light)	
Animals		Bernhard Modern	
Antique No. 3		Bernhard Tango (italic)	
Antique Olive		Berthold Barmeno	
Antique Roman		Berthold Bodoni Old Face	
Apollo		Berthold Caslon Buch	
Aquarias No. 8 (bold)		Berthold Colossalis	
Arabic News		Berthold Cosmos	
Arcadia		Berthold Garamond	
Architect		Berthold Script	
Ariadne		Berthold Walbaum Buch	
Arial		Berthold Walbaum Buch (B.metrics)	
Armenian Aramian		Beton Extra Bold	
Armenian Barz		Beverly Hills (inline)	
Arnold Boecklin		Biffo	
Artisan Roman (inline)		Bingham Script (text)	
Artistik		Bison	

Ashley Crawford (bold)		Bits Pic Pi	
Ashley Inline (inline)		Black White (patterned, outline, inline)	
Ashley Script (italic)		Blippo (black)	

Typeface		**Typeface**	
Bloc (outline)		Catull	
Block (bold)		Centaur	
Bodoni		Century Expanded	
Bodoni Antiqua		Century Old Style	
Bodoni Campanile		Century Schoolbook	
Boldface PS		Century Schoolbook Monospace	
Bookman		Cerigo	
Border Pi 1515-9		CG Cloister	
Borders & Ornaments 1		CG Frontiera	
Borders & Ornaments 4		CG Trade	
Borders & Ornaments 5		Chaplin (italic)	
Borders & Ornaments 6		Charlemagne	
Boton		Charme	
Boulevard		Cheltenham	
Bremen		Chemical Pi	
Brighton		Cheq	

Broadpen		Chevalier (bold expanded pattern 0)	
Broadway		Chic (inline)	
Brody (bold upright)		Choc (black)	
Bruce Old Style		Christiana	
Brush (italic)		Chuan Pim (like Univers)	
Bulmer		Citadel Script	
Bundesbahn Pi		City	
Burin Roman		Claire News	
Burin Sans (light)		Clairvaux	
Business & Services 1		Clarendon	
Business & Services 2		Claridge	
Busorama		Classic Roman	
Cable		Clearface Gothic	
Caflisch Script		Cloe	
Caledonia		Cloister Black	
Caliban (condensed italic)		Cloister Open Face (outline)	
Calligrapher		CMC-7 MICR	
Calligraphiques		Cochin	

Calligraphy		Codex	
Calvert		Comenius-Antiqua	
Campanula		Commercial 1	
Candida		Commercial 2	
Cantoria		Commercial Script	
Capone Light		Communication 1	
Caravan LH Four		Communication 2	
Caravan LH One		Communication 3	
Caravan LH Three		Communication 6	
Caravan LH Two		Compacta (expanded)	
Carolina		Compliment	
Carta		Concorde	
Cartier		Concorde (B.metrics)	
Cartoon Script Roman		Concorde Nova	
Cascade Script		Congress	
Caslon 540 & No. 3		Cooper	
Caslon Antique (contour)		Copal (outline, patterned)	

Caslon Open Face (inline)		Copal (solid)	
Castellar (inline)		Copperplate Gothic (display)	
Cataneo		Copperplate Gothic (text)	

Typeface		Typeface	
Corona		Engravers Text (inline)	
Coronet (italic)		Engravers' Old English	
Courier		Engravers' Roman	
Cremona		Eon Age (pattern 1)	
Critter		Erbar	
Cutout		European Pi	
Cyrillic 22		Eurostile	
Cyrillic Helvetica		Eusebius	
Cyrillic Times		Eusebius Open (inline)	
Cyrillic Univers		Ex Ponto	
Data 70		Excelsior	
David		Fairfield	
Davida (text)		Falstaff (black)	
De Vinne		FangSong	
Decoration Pi		Fehrle Display	
Deepdene		Fette Fraktur	

Della Robbia		Figaro	
Delphian (inline)		Fine Hand	
Delphin		Flash (italic)	
Delta		Flemish Script (italic)	
Devanagari (Hindi)		Flintstones	
Didot		Fluidum Bold (italic)	
Digi Fraktur		Flyer	
Digital		Folio	
DIN Engschrift (condensed)		Footlight	
DIN Mittelschrift		Forbes Bold	
Diotima		Formal Script	
Discus		Formata	
Dom Casual		Forte (bold italic)	
Dominante		Franco	
Dorchester Script		Frank Ruehl	
Doric		Franklin Gothic	
Dynamo (extra bold)		Freestyle Script	

E13B MICR		French Script	
EACT (Thai)		Friz Quadrata (ITC)	
Eagle Bold		Frutiger	
East Asian Helvetica		Fry’s Baskerville	
East Asian Times		Futura	
Eccentric		Futura Black	
Eckmann (text)		Galaxy Run (pattern 3)	
Eclipse (pattern 0)		Gallia (inline)	
Ecology		Gambling Pi	
Egiziano Black		Games & Sports 1	
Egyptienne (condensed)		Games & Sports 2	
Egyptienne F		Games & Sports 3	
Ehrhardt		Games & Sports 4	
Electra		Gando Ronde Script	
Elite		Garamond (Simoncini)	
Else		Garamond (Stempel)	
Embassy Script		Garamond No. 3	
Empire (ultra condensed)		Garth Graphic	

Englische Schreibschrift (italic)		General Symbols 1	
Engraver’s Gothic (text)		General Symbols 2	

Typeface		**Typeface**	
Geometric		Hobo	
Giddyup		Holidays 1	
Gill Sans		Holland Seminar	
Gillies Gothic Bold (italic)		Holland Title	
Globe Gothic		Hollandse Mediaeval	
Glypha		Horley Old Style	
Gosikku (Kaku, gothic Japan)		HP-GL Drafting	
Gothic (Japan)		HP-GL Spline	
Gothic (numbered)		Huxley Vertical	
Goudy Catalogue, addt'l Old Style faces		Ideal Schreibschrift	
Goudy Handtooled (inline)		Imago	
Goudy Heavyface (black)		Impact	
Goudy Modern		Imperial	
Goudy Old Style		Impressum	
Goudy Text		Imprint	
Granjon		Impuls (italic)	

Graphite		Industria	
Grayda		Industry & Engineering 1	
Greek & Math Serif		Industry & Engineering 2	
Greek Antique Olive		Inflex	
Greek Apla		Insignia	
Greek Florentine Script II		Intanon (Thai)	
Greek Futura		Isabella	
Greek Helios II		Isil Gothic	
Greek Helvetica		Italia (ITC)	
Greek Microstyle		Italian Old Style	
Greek Oracle		ITC American Typewriter	
Greek Times		ITC Anna (condensed)	
Greek Univers II		ITC Avant Garde Gothic	
Greeting Monotone		ITC Barcelona	
Grotesque		ITC Bauhaus	
Guardi		ITC Beesknees (black)	
Gyosho		ITC Benguiat	

Hadassah		ITC Benguiat Gothic	
Hadriano		ITC Berkeley Oldstyle	
Hallmark Bodoni		ITC Bolt (extended)	
Handel Gothic		ITC Bookman	
Handle Oldstyle		ITC Busorama	
Hanseatic (ultrabold condensed)		ITC Caslon No. 224	
Happening		ITC Century	
Harry		ITC Cheltenham	
Hei (China)		ITC Clearface	
Heldustry		ITC Cushing	
Helinda Rook		ITC Elan	
Helios II		ITC Eras	
Hellenic Wide (extended)		ITC Esprit	
Helvetica		ITC Fenice	
Helvetica Inserat (condensed)		ITC Flora	
Helvetica Rounded		ITC Franklin Gothic	
Henche		ITC Galliard	
Heritage		ITC Garamond	

Hess Neobold		ITC Giovanni	
Hiroshige		ITC Gorilla (text)	

Typeface		**Typeface**	
ITC Grizzly		Jetsons	
ITC Grouch		Jiffy	
ITC Highlander		Joanna	
ITC Honda (black)		Joanna Solotype (inline)	
ITC Isadora		Jukebox (bold condensed)	
ITC Isbell		Kai Medium	
ITC Jamille		Kaisho	
ITC Kabel		Kartoon	
ITC Korinna		Kaufmann	
ITC Leawood		Kennerley	
ITC Legacy Sans		Keycap Pi	
ITC Legacy Serif		Kino (bold condensed)	
ITC Lubalin Graph		Kismet	
ITC Machine (condensed)		Klang (italic)	
ITC Mendoza		Koch Antiqua	
ITC Mixage		Koloss (extra bold)	

ITC Modern 216		Komain (Thai)	
ITC Mona Lisa Recut (inline)		Kompakt (ultra black italic)	
ITC Mona Lisa Solid (upright)		Koufi	
ITC New Baskerville		Krishna (Gujarati)	
ITC Newtext		Kuenstler Script	
ITC Novarese		Kyokasho (text book)	
ITC Officina Sans		Land Pi	
ITC Officina Serif		Latin Antique	
ITC Ozwald (fatface)		Latin MT	
ITC Pacella		Latin Wide (extended)	
ITC Pioneer (outline shadow)		Letraset Arta	
ITC Quay Sans		Letraset Bramley	
ITC Quorum		Letraset Caxton	
ITC Ronda		Letraset Crillee	
ITC Serif Gothic		Letter Gothic	
ITC Slimbach		Liberty	
ITC Souvenir		Libra	
ITC Souvenir Greek		Life	

ITC Stone Informal		Line Printer	
ITC Stone Sans		Lino Letter	
ITC Stone Serif		Linotype Astrology Pi	
ITC Studio Script (italic)		Linotype Centennial	
ITC Symbol		Linotype Holiday Pi 1, 2, & 3	
ITC Tiepolo		Linotype Modern	
ITC Tiffany		Linotype Technical Pi 1 & 2	
ITC Tom's New Roman		Linotype Textil Pi 1 & 2	
ITC Usherwood		LiShu (China)	
ITC Veljovik		Lithos	
ITC Weidemann		Lo-Type	
ITC Zapf Book		Logan (pattern 0)	
ITC Zapf Chancery		Logos	
ITC Zapf Dingbats		London Text (inline)	
ITC Zapf International		Looney Tunes	
Jaeger Daily News		Looney Type	
Janson Text		Lotus (pattern 0)	

Jefferson		Lucia	
Jets		Lucian	

Typeface		**Typeface**	
Lucida		Monoline Script (italic)	
Lucida Blackletter		Monotype Ellington	
Lucida Bright		Monotype Goudy Sans	
Lucida Calligraphy		Monotype Old Style	
Lucida Casual		Monterey Script (italic)	
Lucida Fax		Moore Computer	
Lucida Handwriting		Motter Corpus (extrabold)	
Lucida Sans		Mr. Big	
Lucida Sans Typewriter		Murray Hill	
Lutheresche Fraktur		Musical	
Lydian		Musketeer	
Macbeth		Myoungjo (Korean)	
Madame (patterned with shadow)		Myriad	
Madison		Mythos	
Mahlau (condensed)		Narkis Tam (like Univers)	
Malik		Naskh	

Mandate		Neo Didot	
Marigold		Neon (Nebiolo)	
Maritime Pi		Neuland	
Marking Numbers Squares		Neuzeit Grotesk	
Maru Gosikku (round gothic Japan)		Neuzeit S	
Mathematical Pi		New Aurora Grotesque	
Matra (pattern 0)		New Berolina (italic)	
Matura (bold)		New Century Schoolbook	
Maximus		News Gothic	
McCollough		Noparat (Thai)	
Medical & Pharmaceutical 1		Noris Script (italic)	
Medici Script		Nork	
Melior		Normande	
Memphis		Notre Dame	
Menue		Nubian	
Mercurius		Nueva	
Meridien		Nuptial Script	

Metro		OCR-A	
Metronome Gothic (bold extra condensed)		OCR-B	
Metropolis (extra bold)		Octavian	
Mezz		Odilia	
Michelangelo		Old Fashion Script	
Miehle Condensed		Old Style 7	
Milestones		Olympian	
Mincho (Japanese)		Ondine	
Ming		Onyx	
Minion		Optima	
Minister		Oranda	
Mirarae		Orator	
Miryam		Orbit-B	
Mister Earl (condensed)		Original Script	
Mistral		Orlando Caps	
Mobil		Ornaments	
Modern		Oscar	
Modern #20		Othello (bold condensed)	

Modernique (extra bold)		Ousbouh	
Modernistic (inline)		Oxford (italic)	

Typeface		Typeface	
P. T. Barnum		Poppl-Laudatio	
Packard		Poppl-Pontifex	
Paetai (Thai)		Poppl-Pontifex (B.metrics)	
Palace Script (italic)		Poppl-Residenz	
Palatino		Post Antiqua	
Palette (italic)		Post Antiqua (B.metrics)	
Parisian		Post Mediaval	
Park Avenue (italic)		Poster Bodoni (black)	
Parsons		Present Script	
Pasquale		Presentation	
Peignot		Prestige	
Pelican (italic)		Primer	
Pemai (Thai)		Print	
Penfield No. 3		Profil (bold italic inline)	
Penumbra		Quaint Roman	
Pepita		Quake	
Pepperwood (condensed)		Quirinus Bold (condensed)	

Perpetua		Rad	
Phenix American (extra condensed)		Rainbow Bass (pattern 0)	
Photina		Raj Raja (Tamil)	
Phyllis		Raleigh	
Pi Collection		Ranjit (Gurmukhi)	
Pierrot		Raphael	
Pifont Circle Numbers		Renault	
Pifont OCRA Numbers		Reporter No. 2	
Pifont Square Numbers		Repro Script (italic)	
Pifont Triangle Numbers		Revue	
Nofret		Ritmo Bold (italic)	
Pilgy (Korean)		Riviera (inline)	
Piranesi		Rockwell	
PL Barclay Outline (outline)		Roman	
PL Bartuska Trophy Oblique		Romana (text & bold)	
PL Benguiat Frisky		Romic	
PL Bernhardt		Rosewood	

PL Brazilia		Rotation	
PL Britannia Bold		Rotis Sans Serif	
PL Davison Americana		Rotis Semisans	
PL Davison Zip Bold		Rotis Semiserif	
PL Fiedler Gothic Bold		Rotis Serif	
PL Fiorello Condensed		Roundy	
PL Futura Maxi		Ruling Script	
PL Latin Bold		Runic MT	
PL Latin Elongated (condensed)		Russell Square	
PL Modern		Rusticana	
PL Radiant		Rusticana (Frutiger)	
PL Tower Condensed		Ruzicka	
PL West Behemoth Semi Condensed (XBd Cd)		Ryadh	
PL Westerveldt Light (condensed)		S’maragd	
Plantin		Sabon	
Playbill (condensed)		Sackers Antique Roman	
PMN Caecilia		Sackers Classic Roman	

Poetica		Sackers English Script	
Pompeijana		Sackers Gothic	

Typeface		Typeface	
Sackers Italian Script (italic)		Star Trek Film	
Sackers Roman		Star Trek Gen	
Sackers Square Gothic		Star Trek Next	
Saemmul (Korea)		Star Trek Pi	
Safeer (Arabic)		Stencil (ATF)	
Sallwey Script		Stick	
Salto		Stop	
Salut (bold)		Stratford Extra Bold	
Sammul (Korea)		Strider	
San Marco		Studio	
Sans No. 1		Studz	
Sans Serif Stencil		Stuyvesant (inline)	
Sanvito		Stymie	
Saphir (pattern 0)		Syntax	
Sapphire (pattern 0)		System X3 (pattern 2)	
Sassoon Primary		Tango	

Schadow		TBG Duc de Berry	
Schneidler Mediaeval		TBG Herculanum	
Schwabacher		TBG Omnia	
Scotch Roman		TC Europa Bold	
Script		TC Jasper	
Script Bold		Tea Chest (condensed)	
Seagull		Tekton	
Section Bold Condensed		Tempo	
Serifa		Temporary-Only Font	
Serlio		Textype	
Serlio Dekoration (pi numbers)		Thompson Quillscript	
Serpentine		Thunderbird (extra condensed)	
Shannon		Tieman	
Sharif		Times (Ten, New, etc.)	
Shelley		Times Europa	
Sho		Times Roman	
Shotgun		Title PS	

Siena Black (italic)		Tom (Thai)	
Signature		Toolbox	
Simplified Arabic		Torino	
Sinaloa (pattern 0)		Trade Gothic	
Skjald		Traditional Arabic Script	
Snell Roundhand		Trajan	
SnowCap		Trajanus	
Socho		Transportation 1	
Solemnis		Transportation 2	
Sonata		Triplett	
Song (China)		Trump Mediaeval	
Souvenir Gothic		TSI Caxton	
Space		Typo Roman	
Spartan		U-Thong (Thai)	
Special Alphabets 4		Umbra (open shadow)	
Special Alphabets 5		Uncial	
Special Alphabets 6		Uncle Sam Stars (pattern 0, shadow)	
Spectrum		Uncle Sam Stripes (pattern 1,	

		shadow)	
Star Fleet		Unesco (Thai)	
Star Trek		Univers	

Typeface		**Typeface**	
Universal Greek & Math Pi		Weiss	
Universal News & Commercial Pi		Westinghouse Gothic	
University Roman		Wiesbaden Swing	
Utopia		Wildstyle	
VAG Rounded		Wilhelm Klingspor Gotisch	
Vectora		Wilke	
Versailles		Windsor	
VGC Egyptian 505		Wingdings	
Victorian Silhouette (contour)		Wittenberger Fraktur	
Vineta		Woodblock (bold)	
Virile		Woodstock	
Visigoth (bold italic)		WTC Our Bodoni	

Viva		Yearbook	
Vivaldi		Yuang (Yuan, XiYuang - China)	
Warning Pi		Zebrawood	
Wave		Zeppelin (inline)	
Wedding Text		Zingo	

Expanded Style descriptor makes these family codes obsolete:

	Typeface		**Typeface**
	Pica		ITC Garamond Condensed
	Helvetica Condensed		Franklin Gothic Condensed
	Cooper Black		Franklin Gothic Extra Condensed
	Bauer Bodoni Condensed		Univers Condensed
	Helvetica Outline		Univers Extended
	Futura Condensed		ITC American Typewriter Condensed
	Helvetica Compressed		Antique Olive Compact
	Helvetica Extra Compressed		Helvetica Monospaced

The following family codes are only for Type Director use:

	Typeface		**Typeface**
	-reserved-Limited Serif Proportional		-reserved

		-reserved-Limited Serif Fixed Pitch		-reserved
		-reserved-Limited Sans Serif Proportionl		-reserved
		-reserved-Limited Sans Serif Fixed Pitch		-reserved
		-reserved-Universal		-reserved
		-reserved-Limited Weight Sensitive Prop		-reserved
		-reserved-Universal Sans Fixed		-reserved
		-reserved-Universal Serif Fixed		-reserved
		-reserved-future buckets		-reserved
		-reserved-future buckets		-reserved
		-reserved-future buckets		-reserved
		-reserved-Cour		-reserved
		-reserved-Helv		-reserved
		-reserved-Univ		-reserved
		-reserved		-reserved

	-reserved		-reserved
	-reserved (other Times)		-reserved
	-reserved		-reserved
	-reserved		-reserved

The following family codes are assigned to font vendors and are not to be distributed outside the company in any literature.

	Typeface		**Typeface**
	-reserved-Arabic		Old Dreadful
	Naskh		Carmina
	Naskh Hollow		Arrus
	Advertisers Naskh		Oz
	-reserved-Arabic		-reserved Bitstream Proprietary
	Koufi		Iowan Old Style
	Oberon		-reserved Bitstream Proprietary
	Callisto		-reserved Bitstream Proprietary

	Charter		-reserved Bitstream Proprietary
	Serif Proportional		-reserved Bitstream Proprietary
	Sans Serif Monospace		-reserved Bitstream Proprietary
	Amerigo		Copal(??)
	PiFont		

NOTE: Some of the typeface names in the above tables may be registered trademarks of a third party. Use of these fonts may be conditional upon a license grant from the owners of the fonts. Hewlett-Packard makes no representation as to the quality or performance of the fonts, and references to the fonts does not grant any license or right to use the fonts.

9.4 Font Selection by ID

Downloaded fonts can be specified by their ID numbers (see Chapter 10 for the Font ID command).

Font Selection by ID (Primary) *Esc (# X*
Font Selection by ID (Secondary) *Esc) # X*

Designates a downloaded font by ID as primary or secondary.

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

The designated font, if present, is selected as the primary/secondary font; and all the attributes in the primary/secondary font select table are changed to match those of the designated font. However, Pitch is not changed if the designated font is proportional.

No action occurs if the designated font is unavailable or if the device does not implement Font IDs.

NOTE: Pitch and height are unchanged if the designated font is scalable.

NOTE: If a proportional-space scalable font is selected by ID, the Height command should be sent **prior** to the Font Selection ID command to specify point size. Otherwise, the size will be determined by the height of the former font (as listed in the font select table).

NOTE: For an unbound font, symbol set is determined from the font select table. To specify a different symbol set, the Symbol Set selection command should be sent prior to the Font Selection by ID command.

NOTE: For shared or multi-user environments, HP recommends that soft fonts be selected by attribute rather than by ID.

Chapter 10: Downloading Fonts

Contents of this Chapter

This chapter describes the following PCL commands:

10.1 Introduction to Soft Fonts

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

Font Downloading

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

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

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

Font ID
Font Definition
Character Code$_1$
Character Definition$_1$
Character Code$_2$
Character Definition$_2$
...
Character Code$_n$
Character Definition$_n$

Definitions

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

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

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

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

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

Temporary and Permanent Fonts

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

Unbound Fonts

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

Bitmap and Scalable Fonts

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

10.2 The Font ID

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

Font ID *Esc * c # d/D*

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

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

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

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

EXAMPLE

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

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

10.3 The Font Definition

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

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

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

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

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

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

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

Six font definitions are shown on the following pages:

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

Bitmap Font Definition

The bitmap font definition is shown below.

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

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

Resolution-Specified Bitmap Font Definition

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

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Font Descriptor Size (≥68)		1
2	Format (20)	Symbol Set Type	3
4	Style MSB	Reserved	5
6	Baseline Position		7
8	Cell Width		9
10	Cell Height		11
12	Orientation	Spacing	13
14	Symbol Set		15
16	Pitch (Default CMI)		17
18	Height		19
20	x-Height		21
22	Width Type	Style LSB	23
24	Stroke Weight	Typeface LSB	25
26	Typeface MSB	Serif Style	27
28	Quality	Placement	29
30	Underline Position	Underline Thickness	31
32	Text Height		33
34	Text Width		35
36	First Code		37
38	Last Code		39
40	Pitch Extended	Height Extended	41
42	Cap Height		43

44-46	Font Number	45-47
48-62	Font Name	49-63
64	X Resolution	65
66	Y Resolution	67
68	Copyright (optional)	69
n	**...**	n+1

Intellifont Bound Scalable

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

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

Intellifont Unbound Scalable

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

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Font Descriptor Size (≥88)		1
2	Format (11)	Symbol Set Type (10)	3
4	Style MSB	Reserved	5
6	Baseline Position		7
8	Cell Width		9
10	Cell Height		11
12	Orientation	Spacing	13
14	Symbol set		15
16	Pitch		17
18	Height		19
20	x-Height		21
22	Width Type	Style LSB	23
24	Stroke Weight	Typeface LSB	25
26	Typeface MSB	Serif Style	27
28	Quality	Placement	29
30	Underline Position	Underline Thickness	31
32	Text Height		33
34	Text Width		35
36	Reserved (0)		37
38	Number of Contours (characters)		39
40	Pitch Extended	Height Extended	41
42	Cap Height		43

44-46	Font Number	45-47
48-62	Font Name	49-63
64	Scale Factor	65
66	Master X Resolution	67
68	Master Y Resolution	69
70	Master Underline Position	71
72	Master Underline Height	73
74	OR Threshold	75
76	Global Italic Angle	77
78-84	Character Complement	79-85
DescSize-2	Global Intellifont Data Size	DescSize-1
DescSize	Global Intellifont Data	DescSize+1

n	Copyright (optional) ...		n + 1
	Reserved (0)	Checksum	

TrueType Scalable

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

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

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

Universal

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

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

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

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

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

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

Font Descriptor Size (UINT)

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

Descriptor Format (UBYTE)

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

Unrecognized values invalidate font creation.

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

Symbol Set Type (UBYTE)

Describes the font's relationship to symbol sets.

Symbol Set Organization

Bound font, 7-bit (96 characters) — Character codes 32-127 [decimal] are printable.*
Bound, 8-bit (192 characters) — Character codes 32-127 and160-255 printable.*
Bound, 8-bit (256 characters) — All codes are printable except 0, 7-15, and 27.*
Bound, 16-bit (65535 characters) — All are printable except 0, 7-15, 27, 65279, 65534, 65535
Unbound — Character codes correspond to MSL numbers (Intellifont).

Unbound — Character codes correspond to Unicode numbers (TrueType).

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

Style MSB (UINT16)

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

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

15	14 10	9 5	4 2	1 0
X	reserved	structure	width	posture

Value(#)	=	**Posture** (style word partial sum)		
		0	-	Upright
		1	-	Italic
		2	-	Alternate Italic
		3	-	Reserved
	=	**Width** (style word partial sum multiplied by 4)		
		0	-	Normal
		1	-	Condensed
		2	-	Compressed or extra condensed
		3	-	Extra compressed
		4	-	Ultra compressed
		5	-	Reserved
		6	-	Extended or expanded
		7	-	Extra extended or extra expanded
	=	**Structure** (style word partial sum multiplied by 32)		
		0	-	Solid
		1	-	Outline
		2	-	Inline
		3	-	Contour, Edge effects
		4	-	Solid with shadow
		5	-	Outline with shadow
		6	-	Inline with shadow
		7	-	Contour with shadow
		8-11	-	Patterned (complex patterns, subjective to typeface)
		12-15	-	Patterned with shadow
		16	-	Inverse
		17	-	Inverse in open border
		18-30	-	Reserved
		31	-	Unknown structure

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

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

Baseline Position (UINT16)

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

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

TrueType - Set to 0.

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

Cell Width (UINT16)

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

Bitmap Font - Specified in PCL coordinate system dots.

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

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

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

Cell Height (UINT16)

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

Bitmap Font - Specified in PCL coordinate system dots.

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

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

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

Orientation (UBYTE)

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

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

Intellifont - Set to 0.

TrueType - Set to 0.

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

Spacing (UBYTE)

Value	Format
0	Fixed
1	Proportional
2	Dual-fixed

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

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

Symbol Set (UINT16)

Bound Font

Specifies the symbol set characteristic of the font.

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

$$\text{Symbol Set Descriptor Field} = (\# \times 32) + (L - 64)$$

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

$$\text{Symbol Set Descriptor Field} = (0 \times 32) + (85 - 64) = 21$$

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

Unbound Font

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

Pitch (UINT16)

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

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

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

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

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

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

Height (UINT16)

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

$$(10 \text{ point}) \times (1 \text{ inch} / 72 \text{ point}) \times (300 \text{ dots/inch}) \times (4 \text{ quarter-dots/dot}) = 166.667$$

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

TrueType - Set to 0.

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

xHeight (UINT16)

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

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

Width Type (SBYTE)

Specifies the proportionate width of characters in the font.

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

Style LSB (UBYTE)

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

Stroke Weight (SBYTE)

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

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

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

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

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

CURRENT VERSION

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

15 12	11 0
Vendor	Typeface Family

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

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

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

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

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

SINGLE-BYTE VERSION

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

LASERJET III / DESKJET 500 VERSION

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

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

MSB LSB

15	14 11	10 9	8 0
0	Vendor	Version	Typeface Base Value

Typeface Base Value

0	Line Printer or Line Draw
3	Courier
4	Helvetica
5	Times Roman
6	Letter Gothic

... (See Chapter 9 for a complete list)

Version (typeface word partial sum multiplied by 512)

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

Vendor (typeface word partial sum multiplied by 2048)

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

Default	3
Range	0 to 65535

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

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

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

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

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

Serif Style (UBYTE)

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

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

Values for bits 6 and 7	
64	Sans Serif
128	Serif
192	Reserved

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

Quality (UBYTE)

Specifies the quality or density of the font.

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

DEVICE NOTE: LaserJets ignore this field.

Placement (SBYTE)

Specifies the position of character patterns relative to the baseline.

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

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

Scalable Font - Set to 0.

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

Underline Position (SBYTE)

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

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

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

Underline Thickness (UBYTE)

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

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

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

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

Text Height (UINT16)

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

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

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

DEVICE NOTE: DJ5xx ignores this field.

Text Width (UINT16)

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

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

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

DEVICE NOTE: DJ5xx ignores this field.

First Code (UINT16)

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

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

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

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

Last Code / Number of Characters (UINT16)

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

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

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

Pitch Extended (UBYTE)

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

Scalable Font - This field is set to zero.

Height Extended (UBYTE)

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

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

Cap Height (UINT)

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

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

$$0.7087 \times 65535 = 46445$$

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

$$\% = (\text{Cap Height Data} / 65535) \times 100$$

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

Font Number (UINT32)

Bitmap Font - Should be ignored and set to 0.

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

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

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

MSB LSB

	Byte 44	Byte 45	Byte 46	Byte 47
	43H	01H	69H	59H
	"C"	92505 Decimal		

Font Name (ASC16)

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

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

Scale Factor (UINT16)

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

Master X Resolution (UINT16)

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

X Resolution (UINT16)

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

Master Y Resolution (UINT16)

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

Y Resolution (UINT16)

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

Master Underline Position (SINT16)

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

Master Underline Thickness (UINT16)

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

Font Scaling Technology (UBYTE)

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

An undefined value in this byte invalidates the font.

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

Variety (UBYTE)

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

OR Threshold (UINT16)

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

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

Global Italic Angle (SINT16)

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

Character Complement (Array of UBYTE)

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

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

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

MSL Symbol Index Character Complements

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

Individually defined bits include:

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

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

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

Unicode Symbol Index Character Complements

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

Individually defined bits include:

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

22	0	Code page.
	1	Insufficient symbols for PC-8, PC-8 D/N.
2,1,0	110	Unicode Symbol Index Indicator.

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

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

Global Intellifont Data Size (UINT16)

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

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

DEVICE NOTE: LJs accept 0.

Global Intellifont Scalable Data

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

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

Checksum (UBYTE)

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

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

Copyright (ASCII)

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

Data Segments

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

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

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

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

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

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

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

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

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

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

AP — Application Support Segment

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

BR — Bitmap Resolution Segment (Format 16 only)

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

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

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

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

CC — Character Complement Segment

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

CE — Character Enhancements Segment (Format 16 only)

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

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

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

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

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

Style Word = Posture + Structure

31 12	11 4	3 0
Structure	Reserved	Posture

Bit Positions	**Posture**
1	Italics
0,2,3	Reserved
	Structure
12	Outline
13	Shad

14-31	ow Reser ved

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

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

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

15

Bold	Light

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

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

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

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

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

Sizing Word = Reduction + Expansion

15 8	7 0
Expansion	Reduction

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

	dimension
12-15	Reserved

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

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

CP — Copyright Segment

This optional segment consists of ASCII data.

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

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

The structure is shown below:

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

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

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

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

GC — Galley Character Segment (Format 16 only)

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

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

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

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

The GC segment definition is shown below:

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

This data segment is used only with Format 16 fonts.

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

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

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

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

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

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

GI — Global Intellifont Data

Reserved for future use.

GT — Global TrueType Data

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

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

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

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

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

IF — Intellifont Face Data Segment

Reserved for future use.

PA — PANOSE 1 Description Segment

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

PF — PS-compatible Font Name Segment

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

TF — Typeface String Segment (Format 16 only))

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

B y t e	**15 (MSB) 8**	**7 (LSB) 0**	
0	TF (21574)		
2 4	Data Segment Size (2 x n+2)		
6	Embedded Font Name Flag	Substitute String Length	
8	Substitute String Character List ...		

This data segment is used only with Format 16 fonts.

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

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

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

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

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

VR — Vertical Rotation Segment (Format 16 only)

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

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

Format (UINT16) — Set to 0.

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

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

This data segment is used only with Format 16 fonts.

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

VT — Vertical Substitution Segment (Format 16 only)

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

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

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

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

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

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

This data segment is used only with Format 16 fonts.

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

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

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

Older DeskJet Formats

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

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

DeskJet/DeskJet Plus Bitmap Font Definition

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

Byte	15 (MSB) 8	7 (LSB) 0	Byte
0	Font Descriptor Size (72)		1
2	Descriptor Format (DJ=5)(DJ+=9)	Symbol Set Type	3
4	Reserved (0)		5
6	Baseline Position		7
8	Cell Width		9
10	Cell Height		11
12	Orientation	Spacing	13
14	Symbol set		15
16	Pitch		17
18	Height		19
20	x Height		21
22	Width Type (0)	Style	23
24	Stroke Weight	Typeface	25
26	Slant	Serif Style	27
28	Quality	Placement	29
30	Underline Position	Underline Thickness	31
32	Text Spacing		33
34	Text Width		35
36	First Code		37
38	Last Code		39
40	Pitch Extended (0)	Height Extended (0)	41
42	CAP Height (0)		43
44-46	Font Number		45-47
48-62	Font Name		49-63
64	Horizontal Pixel Resolution (600)		65
66	Vertical Pixel Resolution (300)		67
68	Top Double Underline Position	Top Double Underline Height	69
70	Bottom Double Underline Position	Bottom Double Underline Height	71
72	Printer Specific Block Size (20)		73
74	Font Data Size		75
76	Unidirection Flag	Compressed Flag	77

78	Reserved (0)	No Half-Pitch Flag	79
80	No Double-Pitch Flag	No Half-Height Flag	81
82	No Bold Flag	No Draft Flag	83
84	Bold Method	Reserved (0)	85
86	Pass 2 Baseline Offset		87

DeskJet 500 Bitmap Font Definition

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

	15 (MSB) 8	**7 (LSB) 0**	**Byte**
	Font Descriptor Size (74)		1
	Descriptor Format (12)	Symbol Set Type	3
	Reserved		5
	Baseline Position		7
	Cell Width		9
	Cell Height		11
	Orientation	Spacing	13
	Symbol set		15
	Pitch		17
	Height		19
	x Height		21
	Width Type	Style LSB	23
	Style MSB	Stroke Weight	25
	Typeface		27
	Slant	Serif Style	29
	Quality	Placement	31
	Underline Position	Underline Thickness	33
	Text Height		35
	Text Width		37
	First Code		39
	Last Code		41

	Pitch Extended	Height Extended	43
	Reserved		45
	Font Number		47-49
	Font Name		51-65
	Horizontal Pixel Resolution (600)		67
	Vertical Pixel Resolution (300)		69
	Top Double Underline Position	Top Double Underline Height	71
	Bottom Double Underline Position	Bottom Double Underline Height	73
	Printer Specific Block Size (36)		75
	Font Data Size		77
	Unidirection Flag	Compressed Flag	79
	Hold Time Factor	No Half-Pitch Flag	81
	No Double-Pitch Flag	No Half-Height Flag	83
	No Bold Flag	No Draft Flag	85
	Bold Method	Reserved	87
	Pass 2 Baseline Offset		89
	Pass 3 Baseline Offset		91
	Pass 4 Baseline Offset		93
	Font Name Extension		95-109

Baseline Offsets (UINT)

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

Bold Method (BOOL)

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

Bottom Double Underline Height (UBYTE)

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

Bottom Double Underline Position (SBYTE)

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

Compressed Flag (BOOL)

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

Font Data Size (UINT)

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

Hold Time Factor (UBYTE)

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

Horizontal Pixel Resolution (UINT)

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

No Bold Flag (BOOL)

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

No Double Pitch Flag (BOOL)

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

No Draft Flag (BOOL)

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

No Half Height Flag (BOOL)

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

No Half Pitch Flag (BOOL)

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

Printer Specific Block Size (UINT)

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

Slant (UBYTE)

Specifies the slant of the font.

Top Double Underline Height (UBYTE)

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

Top Double Underline Position (SBYTE)

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

Unidirectional Flag (BOOL)

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

Vertical Pixel Resolution (UINT)

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

10.4 Managing Fonts

Font Control *Esc * c # f/F*

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

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

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

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

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

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

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

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

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

Chapter 11: Downloading Characters

Contents of this Chapter

This chapter describes the following PCL commands:

11.1 Introduction

The Second Part of Font Creation

As described in Chapter 10, font downloading consists of (1) downloading the font definition and (2) downloading the individual characters in the font. Chapter 10 described font definition; this chapter describes character definition.

Each character is treated as a separate entity that can be downloaded and deleted. Individual characters are identified by a **character code** and defined by a **character definition**.

Character Code

The character code is an identification number — analogous to Font ID — attached to each character by *Esc*c#E*. A unique decimal character code (e.g., ASCII 33) must be designated prior to downloading a character's definition. If the font being downloaded already contains a character with this code, the existing character is deleted when the character is downloaded.

Character Definition

A character is defined by *Esc(s#W[character definition]*. This command can send a maximum of 32767 bytes. If more bytes are needed, the command is used again as many times as necessary to send the additional data. The data sent by a single command is called a **block**. Additional data for a given character is sent as ***continuation blocks***.

Character Definition

A ***character definition*** is composed of the following:

- Header (one per block)
- Descriptor
- Data

The character ***header*** consists of the first two byte fields (Format and Continuation) in every character definition. These two bytes are repeated in the continuation blocks, if any. The Format field describes the type of character to follow (e.g., bitmap, Intellifont, TrueType, etc). The Continuation field indicates previous blocks in the character definition: zero means none.

The character ***descriptor*** contains information such as character width and height, left and top offsets, etc.

The binary ***data*** following the descriptor specifies the character shape — raster data for bitmap fonts or outline data for scalable fonts.

11.2 The Character Code

Before each individual character is downloaded, a unique identification code must be assigned so the character can be referenced.

Character Code *Esc * c # e/E*

Establishes a decimal ASCII code for the next character downloaded.

Value(#)	=	Character code (decimal)
Default	=	0
Range	=	0 to 2^{32}-1

The character code is a state variable that must be designated prior to the download of a character descriptor. Any existing characters with the same code are deleted.

Unbound Intellifont Fonts - For unbound Intellifont fonts, the character code for a given character will equal the MSL (HP Master Symbol List) number for that character (see the *Book of Characters* for character code MSL assignments).

Unbound TrueType Fonts - For unbound TrueType fonts, the character code for a given character will equal its Unicode index as provided in the cmap table of the TrueType font file. A special code (0xffff) must be used to download vertical substitution characters, missing character glyphs, and glyph pieces that never stand alone as characters.

EXAMPLE: *Esc*c103E* sets the character code to 103. If followed by the Character Descriptor command (*Esc(s#W*) with a valid character descriptor and data, a character is defined in the code location corresponding to the ASCII lower case "g".

DEVICE NOTE: DJs below 1000 and pre-LJIIs have a range of 255. LJIII has a range of 32767.

11.3 The Character Definition

After downloading the font definition, each character in the font must be defined.

Download Character *Esc (s # W [Character Definition]*

Downloads a character definition with the character code assigned by *Esc*c#E*.

Value(#)	=	Number of bytes following the terminating character
Default	=	NA
Range	=	0 to 2^{32} -1

The value field (#) contains the number of bytes to be downloaded. If more bytes are needed, this command is used again as many times as necessary. The group of bytes sent by one command is called a ***block***. A character definition consists of a first block and zero or more ***continuation blocks***.

An unsupported or invalid character definition is ignored and discarded. An out-of-memory condition during character download deletes the entire font. Reserved fields should be set to 0.

LaserJet Bitmap Character Definition

The format for the LaserJet bitmap character definition and continuation block is shown below. Format is set to 4, and Descriptor Size is set to 14. The character descriptor is grayed.

	15 (MSB) 8	**7 0 (LSB)**
	Format (4)	Continuation (0)
	Descriptor Size (14)	Class (1)
	Orientation	Reserved (0)
	Left Offset	
	Top Offset	
	Character Width	
	Character Height	
	Delta X	
	Bitmap Character Data: (in bytes) ...	

	15 (MSB) 8	**7 0 (LSB)**
	Format (4)	Continuation (non-zero)
	Bitmap Character Data: (in bytes) ...	

LaserJet Bitmap Character Definition and Continuation Block Format

DeskJet Bitmap Character Definition

The format for the DeskJet5xx series (except 540) character definition is shown below. These DeskJets do not allow continuation blocks.

	15 (MSB) 8	**7 0 (LSB)**
	Format (5, 9, 12)	Continuation (0)
	Descriptor Size	Character Type
	Width (normal)	Compressed Width
	Left PS Pad	Right PS Pad

DeskJet Character Definition

Intellifont Character Definition

The format for the Intellifont character definition and continuation block is shown below. Format is set to 10, and Descriptor Size is set to 2. The character descriptor is grayed.

	15 (MSB) 8	**7 0 (LSB)**
	Format (10)	Continuation (0)[1]
	Descriptor Size (2)	Class (3 or 4)
	Contour Character Data: (in bytes) (see the "Class" description) ...	
	Reserved (0) [2]	Checksum [2]

	15 (MSB) 8	**7 0 (LSB)**
	Format (10)	Continuation (non-zero)[1]
	Contour Character Data — resumed: (in bytes) (see the "Class" description) ...	
	Reserved (0) [2]	Checksum [2]

[1] Continuation is supported for classes 1, 2, 3, and 15 only (compound characters cannot be continued.

[2] These bytes appear only on the last (continuation) block of data.

Intellifont Character Definition and Continuation Block Format

TrueType Character Definition

The format for the TrueType character definition and continuation block format is shown below. The Format field is set to 15. Descriptor Size, which includes everything after the continuation byte but prior to the Character Data Size field, may vary; the minimum is 2. The character descriptor is grayed.

Byte	**15 (MSB) 8**	**7 (LSB) 0**	**Byte**
0	Format (15)	Continuation (0)	1
2	Descriptor Size	Class (15)	3
4	{Additional descriptor data may be inserted here}		
...	...		1+Desc Size
2+Desc Size	Character Data Size		3+Desc Size
4+Desc Size	Glyph ID		5+Desc Size
6+Desc Size	TrueType Glyph Data		
	...		
			# - 3
# - 2	Reserved (0)	Checksum	# - 1

TrueType Character Descriptor (no continuation block required)

Byte	**15 (MSB) 8**	**7 (LSB) 0**	**Byte**
0	Format (15)	Continuation (0)	1
2	Descriptor Size	Class (15)	3
4	{Insert additional descriptor data here}		
...	...		1+Desc Size
2+Desc Size	Character Data Size		3+Desc Size
4+Desc Size	Glyph ID		5+Desc Size
6+Desc Size	Beginning of TrueType Glyph Data		
	...		# - 1

Byte	**15 (MSB) 8**	**7 (LSB) 0**	**Byte**
0	Format (15)	Continuation (1)	1
2	Conclusion of TrueType Glyph Data		
	...		
			# - 3

# - 2	Reserved (0)	Checks um	# - 1

TrueType Character Descriptor (multiple character blocks)

Format (UBYTE)

Specifies the character descriptor format.

Value	Format
0	82906A
1	82450A
3	QuietJet
4	LaserJet bitmap
5	DeskJet
6	PaintJet
7	PaintJet XL
8	RuggedWriter
9	DeskJet Plus
10	Intellifont
12	DeskJet 500
15	TrueType

The character is discarded if the format is different from that expected by the device.

Continuation (BOOL)

Specifies whether the subsequent data is a character descriptor block (0) or a continuation (non-zero) of the data associated with the previous character descriptor.

If the byte count in the value field of the Download Character command exceeds 32767, the character must be sent in two or more blocks. The additional bytes are sent in as many *continuation blocks* as needed (except compound characters).

NOTE: Compound characters (e.g. accented characters) cannot be continued.

A character that has not received all the character data is an "incomplete" character. There is at most one incomplete character at a time. If an incomplete character is deleted, any subsequent continuation downloads are ignored.

A continuation block that is downloaded before the first block was received is ignored.

DEVICE NOTE: Set to 0 for DJs below 1200.

Descriptor Size (UBYTE)

Specifies character descriptor size in bytes. The descriptor follows the character header, which consists of the first two bytes of the character definition (the Format and Continuation fields). For bitmap characters, the descriptor size includes Descriptor Size through Delta X. For Intellifont characters, the descriptor size includes only Descriptor Size and Class. For TrueType characters, the descriptor size includes Descriptor Size and Class, but additional descriptor information can follow; therefore, the minimum TrueType descriptor size is 2.

Value	Device
8	DeskJet 5xx except 540 (Format 5 or 9 character descriptor)
8	DeskJet 5xx except 540 (Format 12 character descriptor)
2	Intellifont
2+	TrueType (additional descriptor information can be added)
14	LaserJet bitmap

Class (UBYTE)

Specifies the format of the character data.

Value	Class
1	Bitmap
2	Compressed bitmap
3	Intellifont
4	Compound Intellifont
15	TrueType

DEVICE NOTE: LJ+, LJII, LJ2000 use a value of 1. DJs below 1200 do not use this field.

Class 1: Bitmap Data — Class 1 character data is a string of bytes containing the dot-per-bit image of the character, with no compression. A "1" bit causes the dot to be printed.

The data is grouped in dot rows describing a one-dot high strip of the character from left to right in the direction of the printer’s raster scan. The dot rows are organized from top to bottom of the character (in portrait orientation): the first dot row of data corresponds to the first dot row of the character. The end of each row is padded with zero bits so it contains an integral number of bytes.

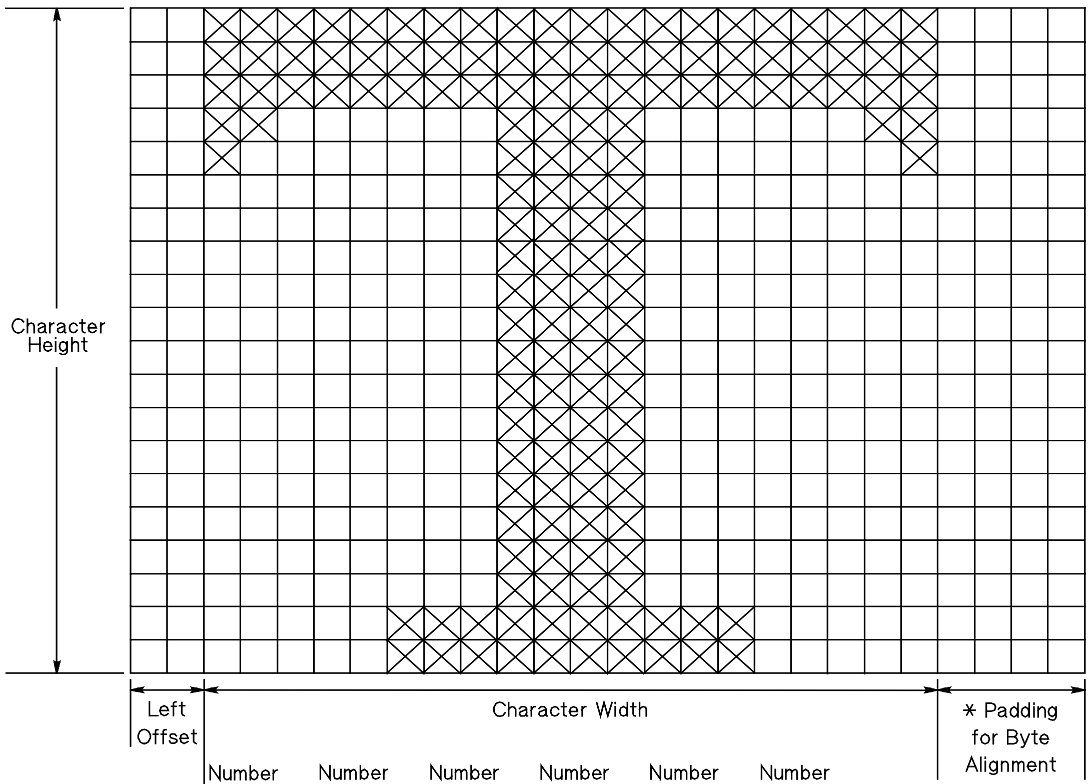
The number of bytes of character data should be: $\mathrm{TRUNC}\left(\frac{\mathrm{CharWidth}+7}{8}\right)\times \mathrm{CharLength}$

Additional data is discarded. The character will consist only of the downloaded character data, even if this is insufficient; the missing part is undefined.

Class 2: Compressed Bitmap Character Data — The string of bytes composing Class 2 character data is in compressed run-length-with-line-repetition format. A row's first byte tells how many times the row is repeated. The second byte tells how many white dots start the row (it is 0 if the first dot is black). The third byte tells how many black dots follow, the fourth byte tells how many white dots follow that, etc.

The *character width* (dots) field in the character descriptor determines the row width. The dot count for each row in the character cell must equal the character width. In the following figure the cell width is 20, so each row (excluding the repetition count byte) adds up to 20.

More than 255 dots of the same type is indicated by a byte containing 255 followed by a byte containing 0 (meaning there are none of the opposite type of dots), followed by a byte containing the count of the remaining dots of the current type.



Line Repetition	Number White Pixels	Number Black Pixels	Number White Pixels	Number Black Pixels	Number White Pixels	Number Black Pixels
2	0	20	–	–	–	–
0	0	2	6	4	6	2
0	0	1	7	4	7	1
12	8	4	8	–	–	–
1	5	10	5	–	–	–

* Byte alignment is necessary only for raster data (ie, not for compressed raster data)

Uncompressed – 60 Bytes
Compressed – 25 Bytes

Class 2: Compressed Bitmap Character Data

Class 3: Intellifont Scalable Character Contour Data — The Intellifont character data is organized as shown below. The data area is grayed.

Byte	15 - MSB 8	7 LSB - 0	Byte

0	Format (10)	Continuation[1]	1
2	Descriptor Size (2)	Class (3)	3
4	Contour Data Size		5
6	Metric Data Offset		7
8	Character Intellifont Data Offset		9
10	Contour Tree Offset		11
12	XY Data Offset		13
14	Tangent Data Offset ...		15
	Metric Data ...		
	Character Intellifont Data ...		
	Contour Tree Data ...		
	XY Coordinate Data ...		
	Reserved (0)	Checksum	

[1] Continuation is supported for classes 1, 2, 3, and 15 only.

Class 3: Intellifont Contour Character Data Format

Class 4: Intellifont Scalable Compound Character Data — A class 4 compound character is a combination or two or more different characters (e.g., accented characters). Continuation should be set to 0, since compound characters cannot be continued. The character data is grayed.

Byte	**15 - MSB 8**	**7 LSB - 0**
0	Format (10)	Continuation (0)[1]
2	Descriptor Size (2)	Class (4)
4	Compound Character Escapement	
6	Number of Components	
8	Component List (See the "Component List" description for the format) ...	
	Reserved (0)	Checksum

[1] Continuation is not supported for class 4.

Class 4: Intellifont Compound Character Descriptor and Data Format

Class 15: TrueType - Since all TrueType scalable characters are handed to the TrueType font scaler in the same format, this field does not provide vital information. For TrueType characters, set this field to 15.

Character Type (UBYTE)

DeskJets below 1200 only. Specifies the type of character.

Value	Class
0	Normal (single pass)
2	Pass 1 of multi-pass
3	Pass 2 of multi-pass
4	Pass 3 of multi-pass
5	Pass 4 of multi-pass

Orientation (UBYTE)

Bitmap fonts only. Specifies the orientation of the character. Character orientation must match the orientation in the font descriptor, as follows:

Value	Orientation
0	Portrait
1	Landscape
2	Reverse-portrait
3	Reverse-landscape

The character is discarded if the orientation is unsupported or different than font orientation.

DEVICE NOTE: LJ, LJ+, LJII support only 0,1.

Left Offset (SINT16)

Bitmap fonts only. Specifies the distance in dots from the reference point to the left side of the character pattern on the physical page coordinate system (i.e.. this value is orientation dependent). The left and top offsets locate the character reference point about CAP.

DEVICE NOTE: The left-offset range is -16384 to 16384 on LJIIIs later, -4200 to 4200 on LJII, and -128 to 127 for LJ+. Values outside these ranges cause the character to be discarded.

Top Offset (SINT16)

Bitmap fonts only. Specifies the distance in dots from the reference point to the top of the character pattern on the physical page coordinate system (i.e., this value is orientation dependent). The left and top offsets locate the character reference point about CAP.

DEVICE NOTE: The legal range is -16384 to 16384 on LJIIIs and later, -4200 to 4200 on LJII, and -128 to 127 for LJ+. Values outside these ranges cause the character to be discarded.

Character Width (UINT16)

Bitmap fonts only. Specifies the width of the character in dots on the physical coordinate system (i.e., this value is orientation dependent). Generally, this width is from the farthest left black dot to the farthest right black dot.

DEVICE NOTE: The legal range for character width is 1 to 16384 on LJIII and LJ2000, 1 to 4200 on LJII, and 1 to 128 for LJ+. Values outside these ranges cause the character to be discarded.

Character Height (UINT16)

Bitmap fonts only. Specifies the height of the character in dots on the physical coordinate system (i.e., this value is orientation dependent).

DEVICE NOTE: The legal range is 1 to 16384 on LJIIIs and later, 1 to 4200 on LJII, and 1 to 128 for LJ+. Values outside these ranges cause the character to be discarded.

Delta X (SINT16)

Bitmap fonts only. Specifies the number of quarter-dot units (radix dots) by which the horizontal position within the PCL logical page coordinate system is incremented after printing the character. If the value field is negative, the value is set to 0. This value is used by the printer only when the font is proportionally spaced.

DEVICE NOTE: The legal range is -32768 to 32767 on LJIIIs and later, and 0 to 16800 on LJII and LJ2000. Values outside these ranges cause the character to be discarded.

Width (UBYTE)

DeskJet 5xx (except 540) only. This field holds the normal width of the character (in 600ths of an inch). The normal width of a character can take on values from 0 to 254. A value of 255 specifies that the character is a non-printing, non-spacing character. Non-printing, non-spacing characters print a blank space in transparent print mode and display functions mode.

Compressed Width (UBYTE)

DeskJet 5xx (except 540) only. If the font supports algorithmic compressed printing, this field holds the compressed width of the character (in 1/600ths of an inch). If the width of the character is equal to 255 (designating a non-spacing, non-printing character), this value is also 255. In landscape fonts, this field is the left pad.

Left PS PAD (UBYTE)

DeskJet 5xx (except 540) only. If the font is proportionally spaced, this field holds the left pad value of the character (in 600ths of an inch). The value can range from 0 to 255 or -127 to +127.

DEVICE NOTE: This field is a UBYTE for DeskJet Classic and Plus, but an SBYTE for all other DeskJets in the 5xx series, except the 540, which does not use it.

Right PS PAD (UBYTE)

DeskJet 5xx (except 540) only. If the font is proportionally spaced, this field holds the right pad value of the character (in 600ths of an inch). The value ranges from 0 to 255 or -127 to +127.

DEVICE NOTE: This field is a UBYTE for DeskJet Classic and Plus, but an SBYTE for all other DeskJets in the 5xx series, except the 540, which does not use it.

Character Data

The character data is in the format specified by the class field.

Contour Data Size (UINT16)

Intellifont only. The size of the contour data including this size field.

Metric Data Offset (SINT16)

Intellifont only. The offset to the Metric Data relative to the Contour Data Size field address.

Character Intellifont Data Offset (SINT16)

Intellifont only. The offset to the Character Intellifont Data relative to the address of the Contour Data Size field.

Contour Tree Offset (SINT16)

Intellifont only. The offset to the contour Tree Data relative to the address of the Contour Data Size field.

XY Data Offset (SINT16)

Intellifont only. The offset to the XY data relative to the address of the Contour Data Size field.

Tangent Data Offset (SINT16)

Intellifont only. The offset to the Tangent Data relative to the address of the Contour Data Size field. This field should be set to -1. This field appears in HQ2 characters but not in HQ3 characters. HQ2 and HQ3 formats are described in *Intellifont SubSystem Specification 2.0 version 6*.

Metric Data

Intellifont only. See *Intellifont SubSystem Specification 2.0 version 6*.

Character Intellifont Data

Intellifont only. See *Intellifont SubSystem Specification 2.0 version 6*.

Contour Tree Data

Intellifont only. See *Intellifont SubSystem Specification 2.0 version 6*.

XY Coordinate Data

Intellifont only. See *Intellifont SubSystem Specification 2.0 version 6*.

Compound Character Escapement (SINT16)

Intellifont only. The escapement in dimensional units of a resulting compound character.

Number of Components (UBYTE)

Intellifont only. The number of components of the compound character.

Component List

Intellifont only. The component list contains a 6-byte record (shown below) for each compound character component. The number of 6-byte records is determined by the Number of Components field described above.

Byte	15 (MSB)8	7 (LSB) 0	Byte
0	Character Code		1
2	X Offset		3
4	Y Offset		5

The *Character Code* is the character code number of a component of a compound character. *X Offset* is the offset of a component from the origin in the x direction in design window units (as defined in the Scale Factor field of the font descriptor).
Y Offset is the offset of a component in the y direction from the origin in design window units.

NOTE: The character code may be greater than the last code of the symbol set that is implied by the font type, since a compound character can include components that are not part of the symbol set.

Character Data Size (UINT16)

TrueType only. This value should equal the sum of the sizes of the Character Data Size, Glyph ID, and TrueType Glyph Data fields. It allows the PCL interpreter to know when a continuation block is needed. The minimum value is 4.

The value of Character Data Size plus Descriptor Size plus 4 (for the Format, Continuation, Reserved and Checksum bytes) is never less than the value in the Define Character command. If the sum is equal to the value in the command, no continuation block is expected for the given character; if the sum is greater, a continuation block is needed.

The validity of a downloaded TrueType character requires that the sum of the Define Character command values for all of that character's blocks equals the sum of Descriptor Size and Character Data Size plus 2 (for Reserved and Checksum), plus 2 times the number of character blocks (for Format and Continuation bytes).

Glyph ID (UINT16)

TrueType only. This field is used by the TrueType font scaler as an ID number for the glyph data associated with the given character.

TrueType Glyph Data

TrueType only. This field contains the data segment associated with the given character as found in the glyph table of the original TrueType font file. (See the *TrueType Font File Specification*).

Checksum (UBYTE)

Intellifont. The checksum of all the contour character data. The checksum is for all the blocks added together.

TrueType. The value of this byte, when added to the sum of all of the bytes in the Character Data Size, Glyph ID, and TrueType Glyph Data fields, should equal 0 (modulo 256). The Checksum is found in the last block for a given character.

Chapter 12: Unbound Fonts and Downloaded Symbol Sets

Contents of this Chapter

This chapter describes the following PCL commands:

12.1 Introduction

Bound and Unbound Fonts

A ***bound*** font is restricted to a single symbol set. An **unbound** font contains the union of multiple symbol sets described by a Symbol Index such as Hewlett-Packard's Master Symbol List (MSL) or Unicode (TrueType). Unbound fonts typically contain 300, 400, or even more characters.

Downloading Symbol Sets

Symbol sets may be downloaded like fonts. Both customized and standard HP-supported symbol sets may be downloaded.

12.2 Unbound Font Operations

Font Selection and Unbound Fonts

When a font is requested, the printer selects an available font that most closely matches the current font selection characteristics — symbol set, spacing, pitch, height, style, stroke weight, and typeface. (See Chapter 9 for font selection.)

Since the symbol set attribute has the highest priority, the printer first makes a list of all the available fonts — bitmap, bound, and unbound — that contain the requested symbol set. Bitmap and bound fonts are easily checked because they contain only one symbol set. However, to determine which unbound fonts match a symbol set, the printer must identify the appropriate *symbol collections*.

Symbol Collections

The symbols in an unbound font can be divided into symbol collections that classify symbols according to language or application. The symbols in each collection do not change from one unbound font to another — that is, the Basic Latin collection always contains the same symbols. However, different fonts may contain different collections. For example, the internal Univers font in LaserJet IIIP contains the Latin, Math, and Semi-graphic collections, which together contain all the symbols required for the 35 symbol sets that Univers provides. On the other hand, the Dingbats font contains only the Dingbats collection, which contains all the symbols required for the five supported Dingbat symbol sets.

Matching Unbound Fonts to Requested Symbol Sets

The *Character Complement* number, which is a 64-bit field in an unbound font definition (see Chapter 11), identifies the symbol collections contained in the font. Each bit in this field corresponds to a symbol collection. For example, if bit 63 is cleared, an unbound Intellifont contains the Basic Latin collection. If bit 31 is cleared, an unbound TrueType font contains ASCII.

The *Character Requirements* number, which is a 64-bit number provided to the printer for each symbol set, identifies the character collections needed by the symbol set. The printer matches bit-by-bit the Character Requirements number for the requested symbol set with the Character Complement number of every unbound font in the printer.

After matching the Character Complement number with the required symbol collections, the printer will contain a list of all fonts (bitmap, bound, and unbound) that support the requested symbol set. If no fonts are found for the requested symbol set, the printer continues with the font selection algorithm described in Chapter 9.

Summary of Symbol Set Matching

Symbol set matching for unbound fonts uses the following general procedure:

1. The host requests a font through font attribute selection by updating the font attribute table.
2. Since the symbol set attribute has the highest priority, the printer looks for the requested symbol set.
3. The printer uses the *Character Requirements* number of the requested symbol set to identify the symbol collections needed by the symbol set.
4. The printer checks the *Character Complement* field of available fonts for symbol collections that contain symbols in the requested symbol set.
5. Fonts whose symbol collections together contain the necessary symbols are included in the next round of attribute matching.

Symbol set or point size are not specified when an unbound scalable font is downloaded. Designation by ID is equivalent to a PCL font select string containing parameters with fixed values (spacing, stroke weight, style, and typeface), but not point size and symbol set. To designate an unbound scalable font by ID, first select the desired symbol set and point size, and then select the font by ID.

Symbol Indexes (MSL and Unicode)

Symbol Indexes identify HP printer symbols by a unique number. Two symbol indexes are used for unbound fonts. Unbound Intellifonts use MSL (Master Symbol List) numbers; unbound TrueType fonts use Unicode numbers. Character collections differ between Intellifont and TrueType fonts.

Since the printer identifies symbols by their Symbol Index Number (0-65535), but receives character codes (0-255), the printer has a symbol set mapping table for each available symbol set. Using this mapping, the printer identifies which indexed character will be printed for the requested character code. A partial mapping for Roman-8 set is shown below.

Character Code	**MSL Number**	**Unicode Number (hex)**
32	0000	(no HMI variable space)
33	0001	0x0021
34	0002	0x0022
35	0003	0x0023
36	0004	0x0024
37	0005	0x0025
38	0006	0x0026
39	0008	0x2019
40	0009	0x0028
41	0010	0x0029
...	...	
252	0189	0x25a0
253	0190	0x00bb
254	0191	0x00b1

Printing a Character

After an unbound font has been selected and the printer receives a character code for printing, the printer must access the mapping table to get the Symbol Index Number for that character. The printer then searches the selected unbound font for the correct MSL or Unicode number and prints that character.

For example, if Roman-8 is requested and the printer receives character code 254, the printer accesses the Roman-8 mapping table. In the Roman-8 mapping table, character code 254 is mapped to MSL number 191 or Unicode number 0x00b1, corresponding to the plus-over-minus symbol, which is then printed.

12.3 Downloading Symbol Sets

Symbol sets may be downloaded and used to bind an unbound font. The downloaded symbol set may be one of the standard HP-supported symbol sets listed in Chapter 9 that is not internal to the printer, or it may be a user-defined or modified symbol set. The following process is used to download a symbol set:

1. Specify an identification number for the symbol set — *Esc*c#R*.
2. Download the symbol set definition — *Esc(f#W [data]*.
3. Select the symbol set for printing — *Esc(ID*.
4. Delete the symbol set or designate it as permanent — *Esc*c#S*.

Symbol Set Identification: Before a symbol set can be downloaded or manipulated as an individual entity, it must be assigned a unique identification number. *Esc*c#R* designates an identification number prior to downloading the symbol set. An existing symbol set with this code is deleted at download.

Symbol Set Definition: *Esc(f#W* downloads a group of symbol set attributes as well as a list that maps each symbol set character code to a character ID number by which the given character may be specified in unbound scalable fonts.

Symbol Set Selection: After downloading the symbol set definition, the user may select the symbol set by *Esc(ID*.

Symbol Set Management: Once a user-defined symbol set is downloaded, the Symbol Set Control command (*Esc*c#S*) can assign symbol sets as temporary or permanent, or delete them.

Symbol Set Code *Esc * c # r/R*

Assigns an identification code to a downloadable symbol set.

Value(#)	=	Identification code
Default	=	0
Range	=	0 to 65535

DEVICE NOTE: LJIIIP and 4 have an upper range of 32767, corresponding to 1023Z. They do not allow values above 1023Z for user-defined symbol sets.

The Symbol Set Code is analogous to Font ID and Character Code. It is used to download and manage symbol sets.

Any downloaded symbol set already associated with this code is deleted when the symbol set definition is received.

The value (#) used for this command must match the Encoded Symbol Set Designator field in the downloaded symbol set definition.

The Symbol Set Code is derived from the identification number (ID) used by *Esc(ID* in font selection:

$$\text{Symbol Set Code} = (\# * 32) + (L - 64)$$

where # is the number portion of the ID, and L is the ASCII value of the letter portion. (See Chapter 9 for a list of symbol set IDs.)

EXAMPLE

Assume that a symbol set has an ID of 1Q, which will be used as a font select parameter by *Esc(ID* — that is, *Esc(1Q*. The Symbol Set Code command for this symbol set would be *Esc*c49R* — that is, (1 * 32) + (81 - 64) = 49.

Then the Download Symbol Set command, *Esc(#W,* will create a symbol set with a symbol set code of 49. And the Symbol Set Control command (*Esc*c#S*) will execute the specified action for any symbol set with a symbol set code of 49.

Download Symbol Set *Esc (f # W [symbol set definition]*

Defines the characters and character mapping for a downloaded symbol set.

Value(#)	=	Number of bytes in symbol set definition
Default	=	NA
Range	=	0 to 32767 (command ignored if invalid definition or out-of-memory)

This command must be sent subsequent to Symbol Set Code (*Esc*c#R*). The last symbol set code sent is used; if no code has been sent, the default (0) is assigned.

A previously downloaded symbol set with the same symbol set code is deleted. An internal symbol set with the same code is overridden by the new symbol set.

If the symbol set definition is invalid, if the Encoded Symbol Set Designator field of the definition does not match the symbol set code, or if there is insufficient memory to create the symbol set, the command is ignored and the symbol set discarded.

The format for a downloaded symbol set definition is shown below:

Byte	15 (MSB) 8	7 0 (LSB)	Byte
0	Header Size		1
2	Encoded Symbol Set Designator		3
4	Format	Symbol Set Type	5
6	First Code		7
8	Last Code		9
10 to 16	Character Requirements		11 to 17
#-2	Symbol Map [Last Code - First Code +1] ...		#-1

User-defined Symbol Set Definition Format

Header Size (UINT16)

Specifies the number of bytes in the header of the symbol set definition. This is the number of bytes preceding the Symbol Map.

For a format 1 (MSL) symbol set definition, the header size must be 18 or greater; otherwise, the symbol set is invalidated.

Encoded Symbol Set Designator (UINT16)

This field must match the value designated by Symbol Set Code (*Esc*c#R*).

Format (UBYTE)

This field specifies the symbol index and format of the symbol set definition:

Value	Format
1	MSL (Intellifont)
3	Unicode (TrueType)

Unrecognized values invalidate symbol set creation.

DEVICE NOTE: LJIIIP recognizes only a format of 1.

Symbol set Type (UBYTE)

Defines printable and unprintable codes for the symbol set:

Value	Symbol Set Organization
0	7-bit (32-127 are printable) *
1	8-bit (32-127 and 160-255 are printable) *
2	PC-8 (All codes are printable except 0, 7 - 15, and 27) *

* All character code positions print in transparency mode.

First Code (UINT16)

Designates the first character code in the set. In a Format 1 or 3 symbol set, this value can be 0 to 255. The symbol set is invalid unless the First Code is less than or equal to the Last Code.

Last Code (UINT16)

Designates the last character code in the set. The value must be between 0 and 255 inclusive, and must not be smaller than the First Code.

Together, the First Code through the Last Code identify the range of character codes that map to the Symbol Index numbers (characters) in the Symbol Map field.

Character Requirements (Array of UBYTE)

This 8-byte field, in conjunction with the Character Complement field in the unbound font definition, determines the compatibility of the unbound font to a symbol set. Each bit in the field represents a specific collection of symbols. Setting a bit to 1 indicates that collection is required; setting the bit to 0 indicates that collection is not required. (Bit 63 is the most significant bit of the first byte; bit 0 is the least significant bit of the 8-byte field.) A font and a symbol set are compatible only if the result of AND'ing the Character Complement field of the font definition with the Character Requirements field of the symbol set definition is 8 bytes of zero.

As described below, Character Requirements differ between MSL-based symbol sets and Unicode-based symbol sets. Unbound Intellifonts are ordered in MSL numbers; unbound TrueType fonts are ordered in Unicode numbers.

MSL Symbol Index Character Requirements

Bit Field	Designated Use
58-63	Reserved for Latin fonts
55-57	Reserved for Cyrillic fonts
52-54	Reserved for Arabic fonts
50-51	Reserved for Greek fonts
48-49	Reserved for Hebrew fonts
3-47	Miscellaneous Uses (South Asian fonts, East Asian fonts, Armenian, Georgian, Amharic, other alphabets, bar codes, OCR, Math, PC Semi-graphics, Dingbats, etc.)
0-2	Symbol Index Indicator

Individually defined bits include:

Bit	Value	Meaning
63	1	Basic Latin required (such as Roman-8 and ISO8859/1 Latin 1)
	0	Basic Latin not required

62	1	East European Latin required (such as ISO8859/2 Latin 2);
	0	East European Latin not required
61	1	Turkish required (such as ISO 8859/9 Latin 5)
	0	Turkish not required
57	1	Cyrillic required (such as ISO 8859 Latin/Cyrillic)
	0	Cyrillic not required
34	1	Math required (such as Math-8)
	0	Math not required
33	1	Semi-graphic required (such as PC-8 D/N)
	0	Semi-graphic not required
32	1	Dingbats required (such as ITC Zapf Dingbats series 100)
	0	Dingbats not required
2,1,0	000	MSL Symbol Index

Unicode Symbol Index Character Complements

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

Individually defined bits include:

Bit	Value	Meaning
31	1	ASCII symbols required (various definitions).
	0	No basic Latin alphabet required.
30	1	Latin 1 extensions required.
	0	No West European added symbols required.
29	1	Latin 2 extensions required.
	0	No East European added symbols required.
28	1	Latin 5 extensions required.
	0	No Turkish and West European added symbols required.
27	1	Publishing symbols required.
	0	No added publishing symbols required.
26	1	Accents (often missing from some sets) required.
	0	No added accents required.
25	1	PCL (Roman-8, Legal, etc.) required.
	0	No added symbols for the above required.
24	1	Macintosh symbols required.
	0	No added symbols for MacIntosh text required.
23	1	PostScript required
	0	No added symbols for PostScript text required.
22	1	Code page required.
	0	No added symbols for PC-8, PC-8 D/N required.
2,1,0	001	Unicode symbol index indicator.

There are no invalid Character Requirements field values. Examples of values are:

Value (Hex)	Meaning
0x0000000000 000000	Default requirement (MSL); symbol set can be used with any typeface indexed by MSL.
0x8000000000 000000	Symbol set (MSL) requires only the Basic Latin Symbol Collection. (e.g., Roman-8)
0x8000000400 000000	Symbol set (MSL) requires both Latin and math characters. (e.g., Math-8)
0x8000000200 000000	Symbol set (MSL) requires both Latin and PC semi-graphic characters. (e.g., PC-8)
0x0000000100 000000	Symbol set (MSL) requires only the Dingbat Collection.
0x0000000000 000001	Default requirement (Unicode); symbol set can be used with typeface indexed by Unicode.
0x00000000a0 000001	Symbol set (Unicode) requires ASCII and East European collections (ISO 8859/2 Latin 2)
0x0000000088 000001	Symbol set (Unicode) requires ASCII and publishing collections (e.g., Ventura US)

Symbol Map (Array of UINT16)

Maps each character code to a symbol index number. The number of symbol index characters in the array must match the number of character codes in the range, First Code through Last Code. If no printable symbol is associated with a given character code (e.g., codes 128 through 160 of Roman-8), the corresponding entry in the Symbol Map is 65535 (0xffff).

12.4 Managing User-Defined Symbol Sets

Symbol set management involves the same types of operations as font management:

- Designating a symbol set temporary or permanent
- Deleting a symbol set

Symbol set management operations are performed on the symbol set designated by the current symbol set code (*Esc*c#R*).

Symbol Set Control *Esc * c # s/S*

Designates user-defined symbol sets as permanent or temporary, or deletes them.

Value(#)	=	0	Delete all temporary and permanent user-defined symbol sets
	=	1	Delete all temporary user-defined symbol sets
	=	2	Delete current user-defined symbol set (last Symbol Set Code specified)
	=	4	Make current user-defined symbol set temporary
	=	5	Make current user-defined symbol set permanent
Default		=	NA
Range		=	0 to 2, 4, 5

A downloaded symbol set is temporary by default. Internal symbol sets cannot be deleted or made temporary; however, a downloaded symbol set can *overlay* (redefine) an internal symbol set, but this is not recommended. An overlaying symbol set must be deleted to access an overlaid ROM-based symbol set. The priority scheme is (highest priority to lowest):

1. Downloaded symbols set (lowest ID)
2. Read/Write removeable disk (lowest ID)
3. Read/Write removeable flash (lowest ID)
4. Read/Write permanent disk (lowest ID)
5. Read/Write permanent flash (lowest ID)
6. Cartridge (lowest unit to highest unit)[1]
7. SIMM (lowest ID)
8. Internal

[1] DEVICE NOTE: On LJIII the left cartridge has priority over the right cartridge. On DeskJets below 1200, the back cartridge has priority over the front.

Chapter 13: Raster

Contents of this Chapter

The commands in this chapter are also used for color raster printing (described in Chapter 14).

This chapter describes the following PCL commands:

Compression Method	*Esc*b#M*	13-12
End Raster	*Esc*rC*	13-6
Destination Raster Height	*Esc*t#V*	13-22
Destination Raster Width	*Esc*t#H*	13-22
Raster Presentation	*Esc*r#F*	13-7
Raster Resolution	*Esc*t#R*	13-6
Scale Algorithm	*Esc*t#K*	13-23
Seed Row Source	*Esc*b#S*	13-21
Source Raster Height	*Esc*r#T*	13-9
Source Raster Width	*Esc*r#S*	13-8
Start Raster	*Esc*r#A*	13-5
Transfer Raster by Plane	*Esc*b#V[data]*	13-10
Transfer Raster by Row/Block	*Esc*b#W[data]*	13-11
Raster Y Offset	*Esc*b#Y*	13-9

13.1 Introduction

This chapter describes raster (bitmap) graphics. A raster image is composed of rows of dots or pixels. A pixel row is transferred to the printer as a string of bytes. Rows are organized from top to bottom: the first data row becomes the top pixel row of the image.

Dot

A dot is the smallest mark a printer can make. Its size and spacing are device specific.

Pixel

The smallest definable picture element in the source image. At maximum machine resolution, a pixel consists of one dot. When scaling up, or at lower resolutions, a pixel may consist of more than one dot.

Bit

Bits refer to the binary 1's and 0's in the source data. In black and white printers, a single bit defines a pixel; a "1" bit prints a black pixel and a "0" bit prints a white pixel.

Bits	**Pixel Rows**
11111111	• • • • • • • •
10101010	• • • •

A color pixel may require more bits per pixel: e.g., two bits for four colors, three bits for eight, etc.

Raster Mode

Start Raster (*Esc*r#A*) begins a restricted state called *raster mode*, which locks out commands that would affect rendering. Raster mode continues until an implicit or explicit End Raster (*Esc*rC*). The Transfer Raster (*Esc*b#V*, *Esc*b#W*) and Raster Y Offset (*Esc*b#Y*) commands can implicitly start raster mode, using the current raster resolution and left raster margin. Any text or non-raster command can implicitly terminate raster mode.

Raster Area

The PCL raster system uses the concept of a bounded raster picture, or *raster area*. Within the boundary, the printer zero-fills missing and incomplete rows and clips data that would fall outside.

Raster height extends vertically from CAP to one of the following:

- the distance specified by Source Raster Height if raster scaling is off (see section below).
- the row preceding an implicit or explicit End Raster (*Esc*rC*).
- the lower edge of the printable area.

Raster width extends from the *left raster margin* (CAP or the left edge of the logical page), and is limited to one of the following:

- the distance specified by Source Raster Width (*Esc*r#S*) if scaling is off (see section below).
- the right edge of the logical page.

DEVICE NOTE: LJs, DJ1200C, and PJXL300 clip at the right logical page boundary if Source Raster Width is specified, and at the right printable area boundary if width is not specified.

Raster is independent of text margins and perforation skip, and never causes a page eject: at the end of the page, CAP moves to the bottom logical page boundary and the rest of the graphic is discarded.

Raster Scaling

If scaling is on, the size of the raster picture is determined by Destination Raster Width (*Esc*t#H*) and Destination Raster Height (*Esc*t#V*). To use raster scaling, Source sizes (*Esc*r#S, Esc*r#T*) must be specified, but Destination sizes are optional (they default to logical page and printable area boundaries).

Raster Resolution

Machine resolution is the dots per inch (dpi) a device is capable of printing. Some devices allow different *requested resolutions* up to the machine resolution. At lower requested resolutions, each pixel consists of more than one dot; but dot-spacing remains the same and the same number of dots-per-inch are printed. When 150 dpi is requested in a 300 dpi machine, 300 dpi is still printed; but the pixel is now composed of four dots and is twice as wide and twice as tall as a pixel printed at 300 dpi. Since dot spacing is always the same, changing the resolution changes the size of the printed raster image.

bit	300 dpi	150 dpi
1	•	•• ••

Requesting a lower resolution does not by itself cause less detail to be printed. The printer still prints at its highest resolution; it just prints the image larger. Using the same data, a 150 dpi image is printed four times as large as a 300 dpi image. To keep the destination image the same size, it is necessary to send the printer one-fourth the information.

Data Compression

Once a rectangular raster area is defined, data transfer can be minimized. The Raster Compression Method command (*Esc*b#M*) provides several types of compression.

Raster Command Sequence

The normal execution sequence for raster (without color) commands is shown below.

Raster Presentation	*Esc*r#F*	
Raster Resolution	*Esc*t#R*	
Source Raster Height	*Esc*r#T*	
Source Raster Width	*Esc*r#S*	
Destination Raster Height	*Esc*t#V*	(if using raster scaling)
Destination Raster Width	*Esc*t#H*	(if using raster scaling)
Configure Image Data	*Esc*v#W[data]*	(if using raster scaling)
Start Raster *Esc*r#A*		
Raster Compression	*Esc*b#M*	
Transfer Raster	*Esc*b#W[raster data]* or *Esc*b#V[raster data]*	
...		
Raster Y Offset	*Esc*b#Y*	
Raster Compression	*Esc*b#M*	
Transfer Raster	*Esc*b#W[raster data]* or *Esc*b#V[raster data]*	
...		
Raster Compression	*Esc*b#M*	
Transfer Raster	*Esc*b#W[raster data]*	
...		
End Raster	*Esc*rC*	

Note that presentation, resolution, height, and width are set outside the *start...data...end* sequence. The picture is sent, using the most effective compression method for each row, during *start...data...end*. Raster presentation, resolution, height, and width are all true PCL modes (see Chapter 4): they are not changed when raster mode is terminated; they remain in their specified state until explicitly changed by issuing the command again, or until reset to their default values by *EscE* or a power cycle. On the other hand, raster compression and left raster margin are reset when raster mode is terminated.

13.2 Raster Mode

The Start Raster command (*Esc*r#A*) begins a restricted state called *raster mode*. Start Raster (*Esc*r#A*) explicitly enters raster mode, and End Raster (*Esc*rC*) explicitly ends raster mode. Implicit start and end raster is allowed, **but is not recommended and should not be documented externally**.

Implicit Start Raster Mode

The Transfer Raster (*Esc*b#V*, *Esc*b#W*) and Raster Y Offset (*Esc*b#Y*) commands implicitly start raster mode, using the current raster resolution and left raster margin.

Implicit End Raster Mode

Text or non-raster commands end raster mode, retaining the current resolution and left raster margin.

Raster Commands that are Ignored in Raster Mode

Raster mode locks out the following commands that might affect rendering of the image.

Assign Color Index (*Esc*v#I*) *	Render Algorithm (*Esc*t#J*)
Color Component (first) (*Esc*v#A*) *	Scale Algorithm (*Esc*t#K*)
Color Component (second) (*Esc*v#B*) *	Simple Color (*Esc*r#U*) * **
Color Component (third) (*Esc*v#C*) *	Source Raster Height (*Esc*r#T*)
Configure Image Data (*Esc*v#W*)	Source Raster Width (*Esc*r#S*) **
Configure Raster Data (*Esc*g#W*)	Start Raster (*Esc*r#A*)
Destination Raster Height (*Esc*t#H*)	Download Dither Matrix (*Esc*m#W*)
Destination Raster Width (*Esc*t#V*)	Monochrome Print Mode (*Esc&b#M*)
Foreground Color (*Esc*v#S*) * **	Color Lookup Tables (*Esc*l#E*)
Gamma Correction (*Esc*t#I*)	Viewing Illuminant (*Esc*i#W*)
Logical Operation (*Esc*l#O*)	Pixel Placement (*Esc*l#R*)
Push/Pop Palette (*Esc*p#P*)	Select Palette (*Esc&p#S*)
Raster Presentation (*Esc*r#F*)	Palette Control ID (*Esc&p#I*)
Raster Resolution (*Esc*t#R*)	Palette Control (*Esc&p#C*)

* DEVICE NOTE: DJ850C does not lock out these commands.

** DEVICE NOTE: DJ540 does not lock out these commands.

Raster Commands that are Allowed in Raster Mode

The following raster commands may be used in raster mode:

Compression Method (*Esc*b#M*)
Transfer Raster Data by Row/Block (*Esc*b#W*)
Transfer Raster Data by Plane (*Esc*b#V*)
Raster Y Offset (*Esc*b#Y*)
Seed Row Source (*Esc*b#S*)

Start Raster *Esc * r # a/A*

Starts raster mode and specifies the starting position of the raster image.

Value(#)	=	0	Start raster at logical page left boundary (current vertical position)
	=	1	Start raster at CAP (current vertical and horizontal position)
	=	2	Turn on scale mode and start raster at logical page left boundary
	=	3	Turn on scale mode and start raster at CAP
Default		=	0
Range		=	0 to 3

DEVICE NOTE: Only Color LJ, DJ1200C, and DJ1600C use values 2 and 3.

This command sets the left raster margin (see the table below). A value of 0 starts the upper left raster at the current vertical position on the left edge of the logical page. A value of 1 starts the upper left raster corner at CAP (the current vertical and horizontal position). Values of 2 or 3 are equivalent to 0 and 1, but additionally enable resolution-independent scaling.

NOTE: To use raster scaling, send Configure Image Data (*Esc*v#W*) before Start Raster (*Esc*r#A*).

The default left raster margin is defined as:

Raster Presentation Mode	**Orientation**	**Default Left Raster Margin**
0	Portrait	Logical page left boundary
0	Reverse portrait	Logical page left boundary
0	Landscape	Logical page left boundary
0	Reverse landscape	Logical page left boundary
3	Portrait	Logical page left boundary
3	Reverse portrait	Logical page left boundary
3	Landscape	50 dots from the logical page top boundary
3	Reverse landscape	50 dots from the logical page top boundary

Using presentation mode 3 and landscape or reverse landscape orientation, each raster row runs along the vertical direction of the logical page.

This command locks out the commands listed on the previous page and clears the seed row for raster compression.

This command fixes raster height, width, resolution, and presentation mode until the next Start Raster command. If raster mode is started implicitly, the current left raster margin, height, width, compression, resolution, and presentation mode are used.

End Raster *Esc * r C*

This command ends Raster Mode. It signifies the end of the transfer of a raster image and ends the current raster row. It performs the following functions:

- Zeros the seed row (see Compression Method *Esc*b#M*).
- Re-enables commands locked out of raster mode by *Esc*r#A*.
- Defaults the compression method to 0.
- Resets the left raster margin to 0.
- Moves CAP to the row immediately following the end of the raster area and zero fills empty rows if source raster height was specified; otherwise to the next row.
- Resets the plane counter.

Raster Resolution *Esc * t # r/R*

Defines the resolution at which raster data is to be printed

Value(#)	=	Resolution in pixels per inch (ppi)
Default	=	Device dependent
Range	=	0 to 2^{32} -1

To assure that the picture is printed without data loss, an unsupported resolution maps to the next highest supported resolution.

This command is ignored during raster mode. If raster mode is entered with scaling ON (*Esc*r2A*, *Esc*r3A*), resolution is ignored but saved.

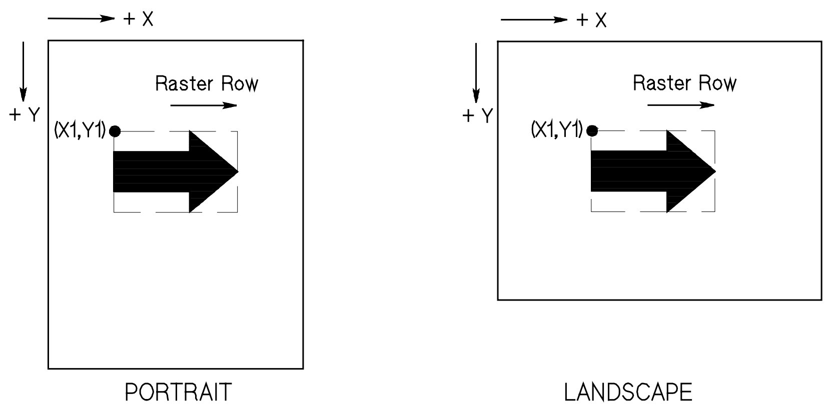
Raster Presentation *Esc * r # f/F*

Specifies the orientation of the raster image on the logical page.

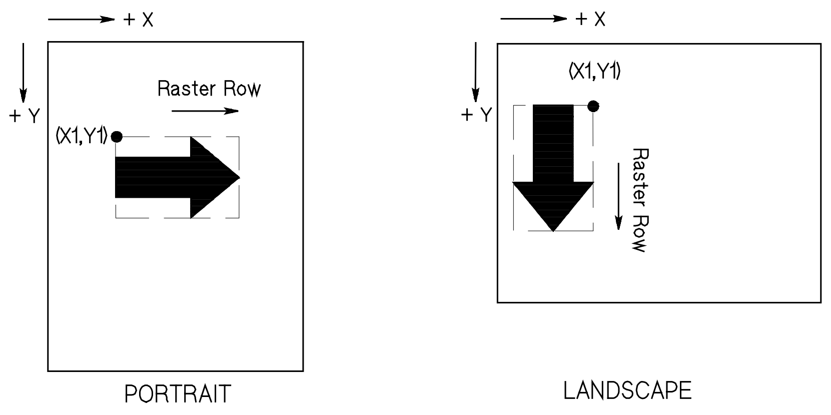
Value(#)	=	0	Prints in the orientation of logical page
	=	3	Prints along the width of the physical page
Default	=	3	
Range	=	0, 3	

A value of 3 prints the raster graphic along the width of the physical page, regardless of logical page orientation. In portrait orientation, a raster row is printed in the positive X-direction, and the next raster row is printed in the positive Y-direction. In landscape orientation, a raster row is printed in the positive Y-direction, and the next row is printed in the negative X-direction.

NOTE: The (X1,Y1) location below is the position of CAP prior to the raster data transfer. The location of (X1,Y1) depends on the value specified by the Start Raster command (*Esc*r#A*).



PRESENTATION MODE 0



PRESENTATION MODE 3

(X1,Y1) = current active position prior to the raster data transfer

Source Raster Width *Esc * r # s/S*

Specifies the width in pixels of the raster area.

Value(#) = Width in pixels of the specified resolution
Default = Right logical page boundary minus the left raster margin (depends on raster presentation mode)
Range = 0 to 2^{32} -1 (clamped to the right logical page boundary minus the left raster margin)

This command specifies a distance from the left raster margin. The left raster margin may be set at CAP (implicit or explicit Start Raster) or at the left edge of the logical page (explicit Start Raster).

The printer clips data that would extend outside the specified width or beyond the right logical page boundary. In other words, raster data that would appear outside the intersection of the logical page, the printable area, and the raster width and height if specified is clipped.

Incompletely-filled rows are zero-filled to the specified width. The color of zero-filled areas depends on the color configuration (i.e. direct vs indexed, white/black references, the color at index 0, source transparency mode, etc).

Since width is in the direction that the raster rows are laid down, source raster width is relative to the current raster presentation mode.

This command is required for raster scaling and must precede a Start Raster command with a value of 2 or 3 (*Esc*r2A* or *Esc*r3A*).

This command is ignored after raster mode starts.

A specified source raster height or width of zero causes subsequent raster data to be ignored.

DEVICE NOTE: LJs, DJ1200C, and PJXL300 clip at the right logical page boundary if Source Raster Width is specified, and at the right printable area boundary if width is not specified.

DEVICE NOTE: LJs, DJ1200C, and PJXL300 zero-fill to the right printable area boundary if source raster width is not specified.

DEVICE NOTE: DJ540 does not lock out this command in raster mode.

DEVICE NOTE: DJs below 800 round Source Raster Widths that are not a multiple of 8 to the next higher multiple of 8 (i.e., the next byte boundary).

Source Raster Height *Esc * r # t/T*

Specifies the height of the raster area in pixels.

Value(#)	=	Height in pixels
Default	=	NA
Range	=	0 to 2^{32} -1 (clamped to logical page length minus the Y position of CAP)

This command specifies a vertical distance from CAP.

If not specified, height extends either to the row preceding an implicit or explicit End Raster (*Esc*rC*) or to the lower edge of the printable area.

The printer clips rows if they are outside the specified height or if the specified height extends outside the printable area with scale mode off.

The printer zero-fills unsent rows to the specified height; but no padding occurs if height is not specified. A specified height or width of zero clips the entire raster graphic. If raster mode is started and stopped before a non-zero specified area is filled, the unsent area is zero-filled. The printer zero-fills rows that incompletely fill the specified width. The color of zero-filled areas depends upon the color configuration (i.e., direct vs indexed, white/black references, the color at index 0, source transparency mode, etc).

"Height" is perpendicular to the direction that raster rows are laid down; therefore, height is relative to the current raster presentation mode (*Esc*r#F*).

This command is required for raster scaling and must precede a Start Raster command having a value of 2 or 3 (*Esc*r2A* or *Esc*r3A*).

This command is ignored once raster mode starts.

Raster Y Offset *Esc * b # y/Y*

Moves CAP vertically relative to CAP. Moves may extend outside the raster area. The amount of movement depends on the device resolution setting.

Value(#)	=	Number of raster lines of vertical movement
Default	=	NA
Range	=	0 to 2^{32} -1

This command zero-fills the offset area. The color of zero-filled areas depends upon the current color configuration (i.e., direct vs indexed, white/black references, the color at index 0, etc).

For Delta Row compression (methods 3 and 9), this command zeros the seed row. For Adaptive compression (method 5), this command applies to the entire raster block of data.

13.3 Raster Data Transfer

The Transfer Raster commands (*Esc*b#V*, *Esc*b#W*) define how many bytes are to be interpreted as binary raster data, rather than as ASCII data.

Transfer Raster by Plane (*Esc*b#V*) is used when the data is encoded by plane, as specified by Simple Color (*Esc*r#U*), Configure Image Data (*Esc*v#W*), or the HP-GL/2 *IN* command. Usually, *Esc*b#V* sends each plane in the row except the last; then *Esc*b#W* sends the last plane and advances the row.

The Transfer Raster by Row command (*Esc*b#W*) moves CAP to the next pixel row after execution. It is used in single-color mode, for the last plane in a multi-plane row, or for color raster transfer when the data is encoded by pixel.

Transfer Raster by Plane *Esc * b # v/V [Data]*

Sends a plane of data to the printer and advances to the next plane, but not to the next row.

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

The number of planes per row is specified by Simple Color (*Esc*r#U*), Configure Image Data (*Esc*v#W*), Configure Raster Data (*Esc*g#W*), or the HP-GL/2 command *IN*. The first plane sent represents the least significant bit in the pixel.

*Esc*b#V* does not advance the row. Only *Esc*b#W* can advance the row.

The amount of data sent is independent of raster width and may vary from plane to plane. Planes shorter than the raster width are zero-filled. Empty planes can be sent by *Esc*b0V* except in Compression Methods 3 or 9 (*Esc*b#M*). If fewer planes are sent than specified and the row ended early by *Esc*b0W*, the undefined planes are zero-filled (except in compression methods 3 and 9). The color of zero-filled areas depends on the current configuration (i.e., direct vs indexed, white/black references, the color at index 0, source transparency mode, etc).

If more planes are sent than specified, the extra planes are ignored and the binary data discarded. If *Esc*b#W* is one of the extra planes, its data is ignored and the row incremented.

If an End Raster command (*Esc*rC*) is received before the row is completed, the output is not rendered and the row is incremented.

The data field is interpreted according to the current Compression Method.

This command implicitly starts raster mode if sent in the absence of a Start Raster command (*Esc*r#A*).

DEVICE NOTE: Little "v" is not supported by LJs and DJ1200 and 1600.

Transfer Raster by Row/Block *Esc * b # w/W [data]*

Transfers the number of bytes specified in the value field to the printer in a row by row or block format, depending on the current compression method *(Esc*b#M)*, then moves CAP to the next row.

Value(#)	=	Number of bytes in the raster row/block
Default	=	NA
Range	=	0 to 2^{32} -1

This is the only data transfer command used in single-color devices, or in multi-color devices in which the data is not plane-encoded. This command increments the row pointer; therefore it is used to send the last plane in a multi-plane row.

In both *row* or *block* formats (see Compression Method 4), CAP is updated for each row. For row formats, CAP is reset to the left raster margin at the next raster row. For block formats, CAP is reset to the initial X-coordinate (left raster margin) of the block at the end of each row, and the Y-coordinate is incremented. For color data, the plane counter in a multi-plane row is reset to the first plane at the end of a row.

The data can vary from row to row and is independent of source raster width. If raster width is greater than data sent, the undefined area is zero-filled. The color of zero-filled areas depends upon the current color configuration (i.e., direct vs indexed, white/black references, the color at index 0, source transparency mode, etc). Data that would appear outside the intersection of the logical page, the printable area, and the specified raster width and height if specified, is clipped; however, the truncated data does affect the seed row.

The data field is interpreted according to the current Compression Method (*Esc*b#M*).

The compression seed row is zeroed upon receipt and completion of a raster block.

DEVICE NOTE: Little "w" is not supported by LJs, and DJ1200 and 1600.

13.4 Raster Compression

Compressed data formats can improve data transfer throughput and efficiency (however, the amount of printer memory required to produce an image is not reduced). The Compression Method command (*Esc*b#M*) provides several ways of compressing raster data.

NOTE: In addition to the compression methods described below, white-space transfer can also be reduced by sending empty rows and planes (*Esc*b0W, Esc*b0V*), or by offsetting the left graphic margin (*Esc*r1A)* or vertical starting point (*Esc*b#Y*); however, these methods are risky because the color of zero-fill depends on the current transparency modes and color configuration.

Compression Method *Esc * b # m/M*

Determines how raster data is interpreted in the Transfer Raster commands (*Esc*b#V, Esc*b#W*).

Value(#)	=	0	Unencoded
	=	1	Run-length
	=	2	Tagged Image File Format (TIFF) rev 4.0 "Packbits"
	=	3	Delta row
	=	4	Unencoded (block-based)
	=	5	Adaptive (block-based)
	=	6	CCITT Group 3 one-dimensional encoding (block-based)
	=	7	CCITT Group 3 two-dimensional encoding (block-based)
	=	8	CCITT Group 4 encoding (block-based)
	=	9	Compressed replacement delta row encoding
Default		=	0
Range		=	0 to 9 (out-of-range values default to 0)

DEVICE NOTE: On the HP5000 fanfold, Source Raster Width (*Esc*r#S*) is not used; and clipping and zero-fill are not supported. Instead, the first four bytes (32-bit unsigned integer field) of Transfer Raster Data by Row (*Esc*b#W*) specify the number of pixels in a row. The decompressed image must fit the dimensions bounded by this value and Source Raster Height (*Esc*r#T*); otherwise, an error occurs. The range for the number of pixels per row that may be specified by this command is 0-5400.

The compression method remains in effect until explicitly changed to another method, a reset (*EscE*) occurs, or raster mode is terminated.

Compression can be turned on or off within a given raster image; and the compression method can be changed for each row.

Methods 6, 7, and 8 are currently defined as monochrome compression methods: data is sent for one plane only. If other planes are sent, all data planes except the first are assumed to be zeroed: the colors defined for indices 0 and 1 are always rendered for 0 and 1 bits respectively, regardless of how many planes were specified by byte two (number of bits per index) of the Configure Image Data command (*Esc*v#W*).

Method 0 - Unencoded

This is a simple binary transfer: no compression. Each bit has a one-to-one mapping to each pixel that is to be printed. Bit 7 of the first byte is the most significant bit. It corresponds to the first dot within the raster row. Bit 0 is the least significant bit and corresponds to the eighth dot.

Method 1 - Run-Length Encoding

The raster data is sent as byte pairs: the first byte is the repetition count for the second byte. Thus, two bytes will be sent even if a byte is not repeated; however, if a byte is repeated, it need only be sent once, along with its repetition count byte. Therefore, method 1 is most useful if there are a large number of identical bytes within a raster row. The repetition count byte contains the total number of consecutive identical bytes minus 1. A count of 0 means the byte pattern occurs once and is not repeated; a count of 1 means the pattern occurs twice; and a count of 255 (the maximum) means the pattern occurs 256 times.

NOTE: Method 1 requires byte pairs: a Transfer Raster command (*Esc*b#V*, *Esc*b#W*) with an odd value field is ignored and the data discarded.

Method 2 - Tagged Image File Format Encoding (TIFF revision 4)

Method 2 combines methods 0 and 1, with blocks of repeated bytes and blocks of unrepeated (*literal*) bytes. A control byte in two's complement format precedes each block; it indicates the number of succeeding bytes in the block, and whether they are repeated or literal.

If bit 7 of the control byte is set (byte = -1 to -127), the following bytes are repeated. The control byte's absolute value is the number of repetitions. The actual number of occurrences is the repetition count + 1. A control byte of -5 means the subsequent byte is repeated 5 times (6 occurrences).

If bit 7 of the control byte is zero (byte = 0 to 127), the following bytes are literal. The number of succeeding data bytes is 1 + the value of the control byte. A control byte of 0 means 1 literal byte follows; a control byte of 6 means 7 literal bytes follow.

A value of -128 represents a non-operative (NOP) control byte. This byte is ignored and the subsequent byte is treated as a new control byte.

It is more efficient to code two consecutive identical bytes as a repeated byte, unless these bytes are preceded and followed by literal bytes.

NOTE: The byte count value (#) or row length specified by the Raster Transfer commands (*Esc*b#V*, *Esc*b#W*) has precedence if it is met before the literal run count.

EXAMPLES OF METHODS 0-2

The following examples show how methods 0-2 can be used to send the raster row below:

Byte Number	#1	#2	#3	#4	#5	#6	#7
Bits	01010101.	01010101.	01010101.	01010101.	01000001.	01010100.	01010100
ASCII	U	U	U	U	A	T	T

Compression method 0 - Unencoded

```
Esc*r1A
Esc*b0m7WUUUUATT
Esc*rC
```

Compression method 1 - Run-length encoding

```
Esc*r1A
Esc*b1m6W(3)U(0)A(1)T
Esc*rC
```

Compression method 2 - TIFF encoding

```
Esc*r1A
Esc*b2m6W(-3)U(0)A(-1)T     -or-     Esc*b2m6W(-3)U(2)ATT
Esc*rC
```

Method 3 - Delta Row Encoding

Delta row compression identifies bytes in a row that are different from the preceding row and then transmits only the data that is different. The previous row that has just been sent is called the **seed row**. The new row that reflects the changes is called the **delta** row. The delta row becomes the seed row when it is printed.

In delta row compression, a raster row consists of a command byte and one to eight replacement bytes:

[Command byte] [replacement bytes]

The command byte has two parts:

7 5	4 0
Number of replacement bytes (1-8)	Offset from the current byte (0-30)

Number of Delta Bytes:	Bits 5-7 indicate the number of consecutive replacement bytes that follow the command byte. The actual number of replacement bytes is always one more than the value (000 = 1, 111 = 8).
Offset:	Bits 0-4 show where to position the replacement byte string. This is the *offset*: it specifies a byte placement, counting from left to right from the current byte position. The "current" byte is the first unaltered byte that follows the last replacement byte; at the beginning of a row, the current byte immediately follows the left raster margin.

An offset of 0 indicates the current byte; an offset of 1 indicates the byte following the current byte. For example, assume the current byte is the first byte in the row. An offset of 7 skips bytes 0 through 6, and a replacement count of 5 replaces bytes 7 through 11. The new current byte is 12. A second offset of 3 skips bytes 12, 13, and 14; byte 15 is the next byte to be replaced.

Bits 0-4 allow maximum value of 31; but larger offsets are possible. A value of 0 to 30 indicates the delta bytes are offset from the 1st to the 31st byte. A value of 31 indicates that an additional *offset byte* follows the command byte. To summarize, bits 0-4 have the following meanings:

0 to 30	The offset is 0 to 30.
31	The offset is 31 or greater. If the offset is 31, an additional **offset byte** follows the command byte. The offset in the command byte is added to the offset byte. If the offset byte is 0, the offset is 31; if the offset byte is 255, additional offset bytes follow. The last offset byte will have a value less than 255. All the offset bytes are added to the offset in the command byte to get the offset value. For example, if there are two offset bytes, and the last byte contains 175, the total offset would be: 31 + 255 + 175 = 461.

If more than eight delta bytes are needed, additional "command byte/delta bytes" are added:

[(Command Byte) (1-8 Delta Bytes) **]** **[** (Command Byte) (1-8 Delta Bytes) **] . . .**

Upon entering raster mode, the seed row is zeroed and its width is set to the Source Raster Width (*Esc*r#S*). Every raster transfer (*Esc*b#V*, *Esc*b#W*) affects the seed row, regardless of the compression method. For example, *Esc*b0W* while in compression method 0 zeros the seed row. This allows method 3 to be combined with other methods in order to achieve better compression performance.

Method 3 operates on each plane independently; a separate seed row is maintained for each plane.

Raster Y Offset (*Esc*b#Y*) zeros the seed row as well as skipped rows. Raster Y Offset affects all planes and rows simultaneously.

The following examples show the effects of some escape sequences in method 3 compression:

*Esc*b0W*	Repeat the previous row. The seed row is unchanged.
*Esc*b1Y*	Move down one raster row. Zero the seed row.
*Esc*b0Y*	Zero the seed row.

The byte count in the value field of the Transfer Raster commands (*Esc*b#V*, *Esc*b#W*) has precedence over the byte count in the command byte. For example, if the byte count is 10, but the data field has 3 bytes, only 3 bytes are replaced. If the last byte indicated by the value field in a Transfer Raster command is a control byte, that byte is ignored; therefore, *Esc*b1W* does not affect the seed row, but causes the previous row to be repeated.

EXAMPLE OF METHOD 3:

The following binary raster data is compressed using method 3. Italicized bytes are those needing replacement — bytes different from the seed row. The seed row is zeroed upon raster mode entry.

Byte #:	1	2	3	4	5
Row 1	00000000	***11111111***	00000000	00000000	00000000
Row 2	00000000	11111111	***11110000***	00000000	00000000
Row 3	***00001111***	11111111	11110000	***10101010***	***10101010***

*Esc*r1A*

Start Raster initializes the seed row to all zeros.

Row 1 - *Esc*b3m2W*(00000001)(11111111)

The **3m** selects method 3, and **2W** indicates that two bytes of data will follow. The upper three bits of the command byte are zero, indicating one byte will be replaced. The lower five bits contain a relative offset of 1, indicating that the replacement occurs 1 byte in from the current position. The replacement byte follows and contains 11111111.

Row 2 - *Esc*b2W*(00000010)(11110000)

The first three bits of the command byte are 0 indicating one byte will be replaced. The lower five bits contain a relative offset of 2; so the replacement will occur 2 bytes from the current position. The replacement byte 11110000 follows.

Row 3 - *Esc*b5W*(00000000)(00001111)(00100010)(10101010)(10101010)

Three bytes are replaced using two commands. The first three bits of the first command byte are zero, indicating a single byte replacement, and the next five bits are zero, indicating a relative offset of zero. The replacement byte 00001111 follows.

The first three bits of the second command byte are 001, indicating the replacement of two bytes; and the lower five bits contain a relative offset of two. The two replacement bytes (10101010)(10101010) follow the command byte.

Method 4 - Unencoded (simple binary transfer by block)

Method 4 is the same as method 0 except that a single *Esc*b#W* command may transfer all or any part of the raster data for the entire image.

Rows start on a byte boundary. The number of pixels per row is specified in the first four bytes of the raster data as a 32-bit unsigned integer binary value. This value must be a multiple of 8; otherwise, the next larger multiple of 8 is used. The row is padded to or clipped at the source raster width (*Esc*r#S*). All planes for the first row are defined before the first plane of the second row, etc.

For both implicit and explicit row/plane divisions, the plane counter is incremented after the data for each plane is received. When the row is completely defined, the row counter is incremented, the plane counter is reset, and CAP is set to the left raster margin (i.e., CAP is reset to the initial X-coordinate of the block at the end of each row and the Y-coordinate is incremented).

If the last row of data specifies an incomplete row or if not all the planes have been defined, the unspecified data is assumed to be 0's. If source raster height (*Esc*r#T*) is specified, the same rules given for source raster width apply.

Method 5 - Adaptive Compression

Adaptive compression enables the combined use of methods 0-3: when the row data within a block is no longer optimally compressed by one method, the method may be changed to "adapt" to the data. It can specify empty (all 0's) or duplicate rows to skip white space or replicate identical rows.

Adaptive compression interprets a raster image as a block of data, rather than as individual rows. When using this method, the Transfer Raster command (*Esc*b#W*) is sent only once, at the beginning of a raster transfer; its value field specifies the number of bytes in the entire block, rather than the number of bytes in a row. The block size of the compressed data is limited to 32767 bytes. To transfer more bytes, more blocks can be sent.

Adaptive compression uses three **control bytes** at the beginning of each row within the block. The first of these bytes, the **command byte**, identifies the type of compression for that row. The following two bytes specify either the number of bytes within the row or the number of duplicate or empty rows. The format of an adaptive compression raster row is shown below:

```
                  msb                   lsb
<command byte> <# of bytes/rows><# of bytes/rows> <raster data>
```

As shown below, the command byte designates the compression method, empty row, or duplicate row:

Value	=	0	Unencoded
		1	Run-Length Encoding
		2	Tagged Image File Format (TIFF) revision 4.0
		3	Delta Row
		4	Empty row
		5	Duplicate row
Range	=	0 to 5 (out-of-range values and data are ignored, the remainder of the block is skipped, CAP is not updated, and the seed row is cleared)	

Command byte values 0-3 indicate the compression method to be used for a row. The two-byte field *<# of bytes/rows>* specifies the number of bytes in the row (**row length)**. The most significant byte (high byte) of this field is sent first, followed by the least significant byte (low byte). The maximum value for the two-byte field is 65535, but the image will be clipped to the logical page; therefore.

A command byte value of 4 prints one or more rows containing zeros. The two-byte field *<# of bytes/rows>*identifies the number of empty or duplicate rows to be encountered after the current row, including the current row. An empty-row operation resets the seed row to zero and updates CAP.

A command byte value of 5 causes the previous row to be printed again the number of times contained in the two-byte field *<# of bytes/rows>*. A duplicate-row operation updates CAP, but does not change the seed row.

USAGE GUIDELINES FOR ADAPTIVE COMPRESSION

- Compression methods cannot be mixed within one row.
- CAP is updated with each row of the raster block. CAP is also incremented for block counts less than 3.
- The Raster Y Offset command moves the entire block of raster data and zeros the seed row — even if the Raster Y Offset is zero.
- Block size takes precedence over row length. If the row length of any line exceeds the block size, the row size is truncated to the block size.
- For method 1, if the row length is odd, CAP is incremented and the row data is skipped (discarded), but the seed row is unchanged.
- For method 1, a row length of zero increments CAP and zeros the seed row.
- For method 2, CAP is always incremented and a new row started when row length is satisfied, regardless of the control byte value. "Extra" bytes following the control byte are used to start the next row.
- For method 2, a row length of 1 inputs one bye from I/O and increments CAP. The data is ignored and the seed row is zeroed.
- For method 3, CAP is always incremented and a new row started when row length is satisfied, regardless of the control byte value. "Extra" bytes following the control byte are used to start the next row.
- For method 3, a row length of zero duplicates the current row and increments CAP.
- For method 3, a row length of 1 inputs one byte from I/O, duplicates the current row, and increments CAP. The data is ignored.
- For duplicate and empty rows, a row length of zero does not update CAP; but the seed row is zeroed.
- An unsupported command byte for a row causes the remaining bytes in the block to be skipped and the seed row zeroed; CAP is not incremented.

EXAMPLE OF ADAPTIVE COMPRESSION

The following example demonstrates adaptive compression.

*Esc*r3F*	Set presentation method to 3; and print along the width of the physical page, regardless of the logical page orientation.
*Esc*t300R*	Set raster resolution to 300 dpi.
*Esc*r1A*	Start raster at CAP. The seed row is initialized to all zeros.
*Esc*b5M*	Set compression method to 5, Adaptive Compression.

*Esc*b84W* Transfer a raster block of data containing 84 bytes.

Ten rows of raster data are sent within the block. The first three rows are compressed using method 3, Delta Row, compression. The next row is compressed by compression method 1, Run Length Encoding. The next three rows of data are specified as duplicate rows of the previous Run Length Encode row. Finally, the last three rows are compressed using method 3. CAP is updated after each raster row within the block is processed.

The raster block has the following format and HEX data:

	Control byte	**# bytes per row**	**Raster data**
	03 hex	00 09 hex	E0 FF F0 00 FF FF 00 0F FF
	03 hex	00 09 hex	E0 00 00 FF F0 0F FF 00 00
	03 hex	00 09 hex	E0 FF F0 00 FF FF 00 0F FF
	01 hex	00 06 hex	00 FF 05 00 00 FF
	05 hex	00 03 hex	
	03 hex	00 09 hex	E0 FF F0 00 FF FF 00 0F FF
	03 hex	00 09 hex	E0 00 00 FF F0 0F FF 00 00
	03 hex	00 09 hex	E0 FF F0 00 FF FF 00 0F FF

*Esc*rC* End raster session.

Method 6: CCITT Group 3 One-Dimensional Encoding (Modified Huffman)

CCITT Group 3 one-dimensional encoding is a block-based compression method that uses a statistical encoding similar to Huffman encoding. The length of alternating white and black (zero and one bit) runs are calculated, and then a table look-up is employed to output the corresponding binary codes. Refer to *CCITT Fascicle VII.3 Recommendation T.4* for details.

Method 7: CCITT Group 3 Two-Dimensional Encoding (Modified READ)

Method 7 is similar to method 6, except that the first row is sent using one-dimensional encoding followed by up to K-1 two-dimensional encoded rows, where K is the K-factor used when the data was encoded. Refer to *CCITT Fascicle VII.3 Recommendation T.4* for details.

Method 8: CCITT Group 4 Encoding (Modified Modified READ)

Method 8 is similar to method 7, except that all encoding is two-dimensional, does not include end of line delimiters, and does not allow padding to byte boundaries. Refer to *CCITT Fascicle VII.3 Recommendation T.6* for details.

Method 9 - Compressed Replacement Delta Row Encoding

Like Method 3, this method replaces only bytes in the current row that differ from the preceding (seed) row. Unlike Method 3, the replacement (delta) bytes may be compressed.

The replacement byte string (delta compression string) consists of a command byte, optional offset bytes, optional replacement count bytes, and the replacement data.

Command Byte	Optional Offset Bytes	Optional Replacement Count Bytes	Data Bytes

The command byte itself has three parts:

Control Bit	Offset Count	Replacement Count

Control Bit Determines whether the replacement data is compressed, and also the bit boundaries of the command byte's other two fields.

Offset Count The offset (number of bytes) the replacement data is offset from the current byte position in the seed row.

Replacement Count The number of consecutive bytes to replaced. One more byte than the replacement count is replaced (i.e., 6 bytes are replaced by a replacement count of 5).

Like compression method 3, The "current" byte follows the last replacement byte; at the beginning of a row, the current byte immediately follows left *raster margin*. An offset of 0 indicates the current byte; an offset of 1 indicates the byte following the current byte.

The size of the offset count and replacement count fields depends on the value of the control bit.

CONTROL BIT = 0

7	6 3	2 0
Control Bit = 0	Offset Count	Replacement Count

If the control bit is 0, the replacement data is uncompressed. Bits 0-2 indicate the replacement count, and bits 3-6 indicate the offset count.

If the offset count is 15, an additional *offset count byte* follows and is added to total offset count. If the offset count byte is 255, another offset count byte follows. The last offset count byte is indicated by a value less than 255.

If the replacement count is 7, an additional *replacement count byte* follows and is added to the total replacement count. If the replacement count byte is 255, another replacement count byte follows. A value less than 255 indicates the last replacement count byte. One more than the total replacement byte count will be replaced. The number of data bytes immediately following the command and its optional bytes is replacement count+1. (That is, the replacement count is normalized from 0-7 to 1-8.)

CONTROL BIT = 1

7	6 5	4 0
Control Bit = 1	Offset Count	Replacement Count

If the control bit is 1, the replacement data is run length encoded. The bit boundaries are different than if the control bit is 0: bits 5-6 contain the offset count, and bits 0-4 contain the replacement count. As when the control bit is 0, optional offset bytes and replacement bytes may be added.

If the offset count is 3, an additional *offset count byte* follows and is added to total offset count. If the offset count byte is 255, another offset count byte follows. The last offset count byte is indicated by a value less than 255.

If the replacement count is 31, an additional *replacement count byte* follows and is added to the total replacement count. If the replacement count byte is 255, another replacement count byte follows. The last replacement count byte is indicated by a value less than 255. The single data byte to be replicated follows the command byte and its optional bytes; it is replicated replacement count+1 times, for a total number of occurrences of replacement count+2. (That is, the replacement count is normalized from 0-31 to 1-32.)

Seed Row Source *Esc * b # s/S*

Specifies the plane of the seed row for multi-plane raster in compression methods 3 and 9.

Value(#)	=	0	Use the corresponding plane of the previous row
	=	>0	Number of previous seed planes before the current plane.
Default	=	0	
Range	=	0 to the number of currently active planes	

For example, if the printer is in 3-plane mode and *Esc*b0S* is sent, then:

1. The Cyan plane would use the Cyan plane from the previous row as a seed plane.
2. The Magenta plane would use the Magenta plane from the previous row as a seed plane.
3. The Yellow plane would use the Yellow plane from the previous row as a seed plane.

If the printer is in 3-plane CMY mode and *Esc*b1S* is sent, then:

1. The Cyan plane would use the Yellow plane from the previous row as a seed plane.
2. The Magenta plane would use the Cyan plane from the current row as a seed plane.
3. The Yellow plane would use the Magenta plane from the current row as a seed plane.

13.5 Raster Scaling

Start Raster (*Esc*r#A*) with a value field of 2 or 3 turns on scale mode. The raster image is rendered in the desired size independently of device resolution.

NOTE: Configure Image Data (*Esc*v#W*) is necessary for raster scaling. It must be sent prior to Start Raster (*Esc*r#A*). The Start Raster command must have a value field of 2 or 3.

Source Raster Width (*Esc*r#S*) and Height (*Esc*r#T*) define the source image size (the amount of data sent). Destination Raster Width (*Esc*t#H*) and Destination Raster Height (*Esc*t#V*) define the size of the destination image that is placed on the page. The scale factor is implicitly determined from destination size, source size, and device resolution.

Specification of destination raster width and height is unnecessary for scaling, since these dimensions default to raster margin and printable area boundaries. If the destination dimensions are not specified, isotropic scaling is maintained such that the entire image is rendered on the page without clipping. If only one destination dimension is specified, that dimension prevails and the other dimension is implicitly determined to maintain isotropic scaling; in this case, clipping may occur.

Destination Raster Width *Esc * t # h/H*

Defines the width in decipoints of the destination raster picture.

Value(#)	=	Width in decipoints
Default	=	Right logical area boundary minus the left raster margin (CAP or left logical page boundary)
Range =	0 to 2^{32} -1	

Zero or absent values default destination width to a value that preserves isotropic scaling.

A specified width that crosses the right printable boundary is clipped; but the scale factor is maintained.

Destination Raster Height *Esc * t # v/V*

Defines the height in decipoints of the destination raster picture.

Value(#)	=	Height in decipoints
Default	=	Bottom printable area boundary minus vertical CAP
Range =	0 to 2^{32} -1	

DEVICE NOTE: Color LJ, DJ1200C, and PJXL300 default to the logical page boundary minus CAP.

Zero or absent values default destination height to a value that preserves isotropic scaling.

A specified height that crosses the lower printable area boundary is clipped; but the scale factor is maintained.

Scale Algorithm *Esc * t # k/K*

Selects an algorithm for enhancing details in downscaling color images with light or dark backgrounds. This command has no effect when scaling upward.

Value(#)	=	0	Enhances color source images with a light background
	=	1	Enhances color source images with a dark background
Default		=	0
Range		=	0,1

This command is valid only after the next Start Raster command (*Esc*r#A*) is sent with the scale mode ON.

This command is ignored for scale factors greater than or equal to 1. For example, assume the source image is 1024 × 1280 pixels and the destination image is 2160 × 2880 decipoints (3" × 4"). At 300 dpi machine resolution, this means a destination size of 900 × 1200 dots; so the specified scale algorithm would take effect.

If the scale factor of either dimension is less than 1, the scale algorithm affects that dimension only.

Chapter 14: Color

Contents of this Chapter

This chapter describes the following PCL color commands:

14.1 Introduction

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

Palette

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

Raster Color vs. Non-Raster Color

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

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

Device-Independence vs. Device-Dependence

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

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

Black and White References

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

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

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

Color Selection

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

Indexed

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

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

Direct

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

Data Encoding

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

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

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

Encoding by Plane

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

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

Encoding by Pixel

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

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

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

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

Color Modes

Four color modes are used in PCL:

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

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

Black and White Mode (Default)

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

Simple Color Mode

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

PCL Imaging Mode

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

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

HP-GL/2 Imaging Mode

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

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

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

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

Device-Independent Color

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

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

Device-Dependent Color

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

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

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

Device-independent Color

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

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

Color Matching

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

Color Appearance Matching

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

Color Lookup Tables

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

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

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

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

Illumination Models

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

Color Processing

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

The PCL language uses color in two different ways:

- Page marking primitives
- Color raster

Page Marking Primitives

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

Color Raster

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

A color raster pixel may be defined by either:

- Palette entry indices
- Direct color specifications

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

Color Processing Functions

Given these two color uses, color processing must:

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

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

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

Device-dependent Color Spaces

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

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

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

Device-independent Color Spaces

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

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

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

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

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

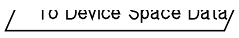
   - Nearest Intensity *Esc*t0J*
   - Device Best *Esc*t3J*
   - Error Diffusion *Esc*t4J*
   - Cluster Ordered Dither *Esc*t7J*
   - Ordered Dither *Esc*t11J*
   - Noise Ordered Dither *Esc*t10J*

   Monochrome render algorithm selections are also allowed.

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

Device-Dependent Pipeline

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



Device-dependent Pipeline

CIE L*a*b* Pipeline

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

Device-Independent Pipeline—CIEL*a*b*

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

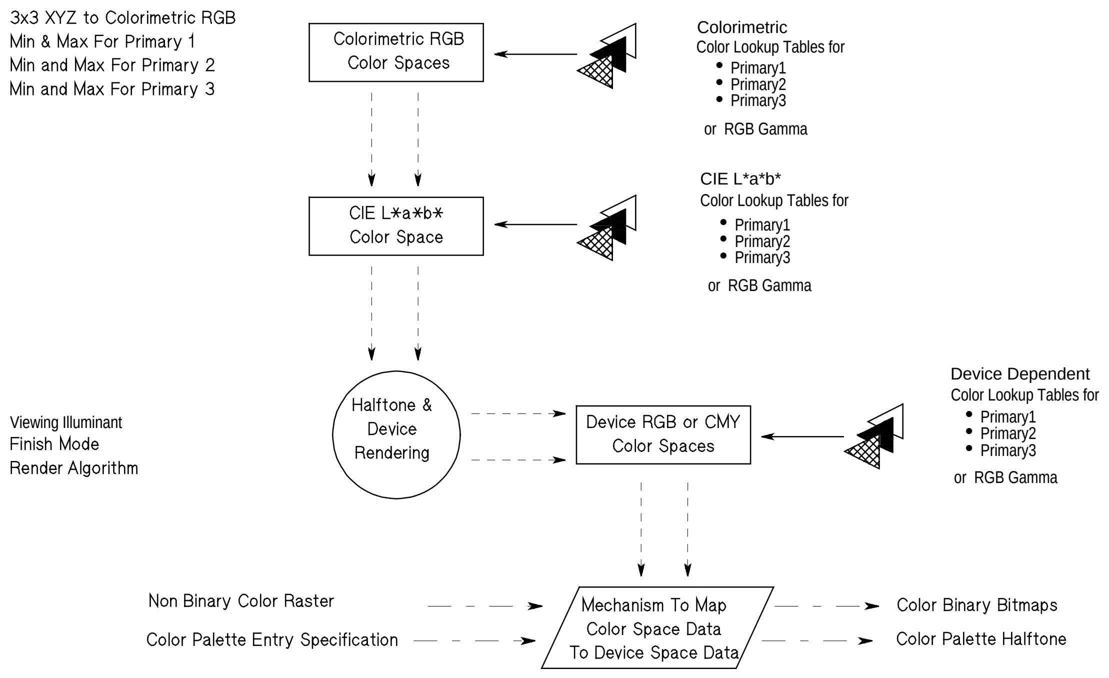
- Rendering algorithm (optional; restricted to the four aforementioned types)
- RGB gamma correction (optional)
- Device-dependent lookup tables for each of the three primaries (optional)
- CIE L*a*b* color lookup tables for each of the three primaries (optional)
- Viewing Illuminant (optional)
- Finish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

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

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

Colorimetric RGB Pipeline

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



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

Device-Independent Pipeline—Colorimetric RGB

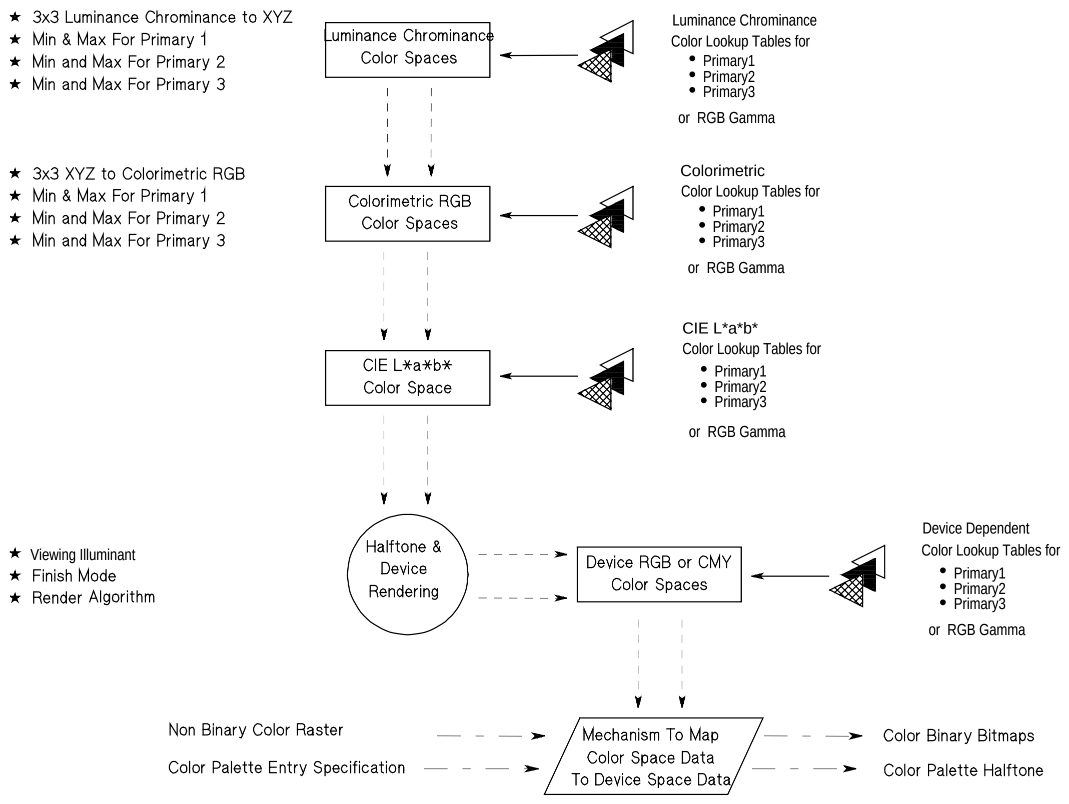
The addition of Colorimetric RGB spaces is merely an extension of the CIE L*a*b* pipeline. Again, this level can have color lookup tables attached to each primary, as well as different color lookup tables attached to different color spaces further down the pipeline. The only new extensions to this level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

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

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

Luminance-Chrominance Pipeline

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



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

Device-Independent Pipeline—Luminance-Chrominance

14.2 Simple Color Mode

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

Simple Color *Esc * r # u/U*

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

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

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

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

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

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

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

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

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

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

The four Simple Color palettes are shown below.

Single Plane (value = 1)

Index	Color
0	White
1	Black

3-Plane RGB (value = 3)

	Color
0	
1	
2	
3	
4	
5	
6	
7	White

3-Plane CMY (value = -3)

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

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

	Black Pen	**Color Pen**
	White	White
	Black	White
	White	Cyan
	Black	Cyan
	White	Magenta
	Black	Magenta
	White	Blue
	Black	Blue
	White	Yellow
	Black	Yellow
	White	Green
	Black	Green
	White	Red
	Black	Red
	White	Black
	Black	Black

14.3 PCL Imaging Mode

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

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

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

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

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

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

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

Common 6-Byte Header

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

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

Byte 0: Color Space

This byte specifies one of the following color spaces:

	Color Space
	Device RGB (default)
	Device CMY
	Colorimetric RGB spaces
	CIE L*a*b*
	Luminance-Chrominance spaces

DEVICE NOTE: DJs support only a value of 0.

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

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

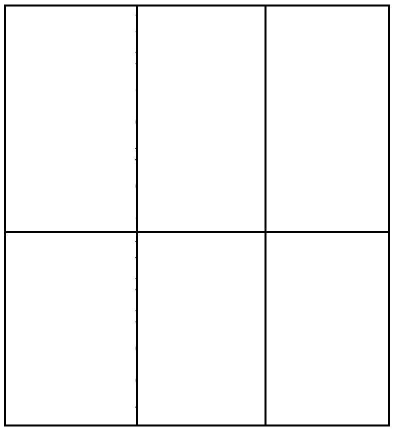
Luminance-Chrominance spaces are a 3x3 linear transformation from Colorimetric RGB. Like CIE L*a*b*, achromatic data is in one channel and chromatic data shares the other two channels.

Byte # 1: Pixel Encoding Mode

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

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

The possible combinations are shown below:



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

MODE 0: INDEXED BY PLANE

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

*Esc*b#V*	row 1	plane 1	i1 i1 ***<u>i1</u>*** i1 i1 ...
*Esc*b#V*		plane 2	i2 i2 ***<u>i2</u>*** i2 i2 ...
*Esc*b#W*		plane 3	i3 i3 ***<u>i3</u>*** i3 i3 ...
*Esc*b#V*	row 2	plane 1	i1 i1 i1 i1 i1 ...

EXAMPLE:

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

MODE 1: INDEXED BY PIXEL

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

*Esc*b#W*	row 1	i4 i3 i2 i1 ***<u>i4 i3 i2 i1</u>*** ...
*Esc*b#W*	row 2	i4 i3 i2 i1 i4 i3 i2 i1 ...
*Esc*b#W*	row 3	i4 i3 i2 i1 i4 i3 i2 i1 ...

EXAMPLE:

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

MODE 2: DIRECT BY PLANE

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

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

EXAMPLE:

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

MODE 3: DIRECT BY PIXEL

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

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

EXAMPLE:

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

Byte # 2: Number Of Bits Per Index

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

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

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

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

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

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

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

Short Form of CID

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

- **Device RGB** ***Esc*v6W[0x00,...]***
- **Device CMY** ***Esc*v6W[0x01,...]***
- **CIE L*a*b*** ***Esc*v6W[0x03,...]***

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

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

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

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

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

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

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

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

Data Range Scaling

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

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

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

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

Long Form of CID

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

Device RGB (Long Form)

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

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

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

Device CMY (Long Form)

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

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

	15 (msb) 8	7 (lsb) 0	
	Color space	Pixel encoding mode	
	Bits per index	Bits per primary #1	
	Bits per primary #2	Bits per primary #3	
	White reference for primary #1 (sint16)		
	White reference for primary #2 (sint16)		
	White reference for primary #3 (sint16)		
	Black reference for primary #1 (sint16)		
	Black reference for primary #2 (sint16)		
	Black reference for primary #3 (sint16)		

CIE L*a*b* (Long Form)

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

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

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

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

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

The white point is based on the standard D6500 illuminant.

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

31 30 23 22 0

	Exponent	Fractional Portion

	15 (msb) 8	7 (lsb) 0	Byte
	Color space	Pixel encoding mode	1
	Bits per Index	Bits per primary #1	3
	Bits per primary #2	Bits per primary #3	5
	Minimum L* value (most significant word)**		7
	Minimum L* value (least significant word)**		9
	Maximum L* value (msw)		11
	Maximum L* value (lsw)		13
	Minimum a* value (msw)		15
	Minimum a* value (lsw)		17
	Maximum a* value (msw)		19
	Maximum a* value (lsw)		21
	Minimum b* value (msw)		23
	Minimum b* value (lsw)		25
	Maximum b* value (msw)		27
	Maximum b* value (lsw)		29

** Floating point format

Colorimetric RGB (Long Form)

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

	15 (msb) 8	**7 (lsb) 0**	
	Color space	Pixel encoding mode	
	Bits per Index	Bits per primary #1	
	Bits per primary #2	Bits per primary #3	
	x Chromaticity for red primary (most significant word)**		
	x Chromaticity for red primary (least significant word)**		
	y Chromaticity for red primary (msw)		
	y Chromaticity for red primary (lsw)		
	x Chromaticity for green primary (msw)		
	x Chromaticity for green primary (lsw)		
	y Chromaticity for green primary (msw)		
	y Chromaticity for green primary (lsw)		
	x Chromaticity for blue primary (msw)		
	x Chromaticity for blue primary (lsw)		
	y Chromaticity for blue primary (msw)		
	y Chromaticity for blue primary (lsw)		
	x Chromaticity for white point (msw)		
	x Chromaticity for white point (lsw)		
	y Chromaticity for white point (msw)		
	y Chromaticity for white point (lsw)		
	Gamma for red primary (msw)		
	Gamma for red primary (lsw)		

	Gain for red primary (msw)	
	Gain for red primary (lsw)	
	Gamma for green primary (msw)	
	Gamma for green primary (lsw)	
	Gain for green primary (msw)	
	Gain for green primary (lsw)	
	Gamma for blue primary (msw)	
	Gamma for blue primary (lsw)	
	Gain for blue primary (msw)	
	Gain for blue primary (lsw)	
	Minimum red value (msw)	
	Minimum red value (lsw)	
	Maximum red value (msw)	
	Maximum red value (lsw)	
	Minimum green value (msw)	
	Minimum green value (lsw)	
	Maximum green value (msw)	
	Maximum green value (lsw)	
	Minimum blue value (msw)	
	Minimum blue value (lsw)	
	Maximum blue value (msw)	
	Maximum blue value (lsw)	

** Floating point format

Luminance-Chrominance (Long Form)

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

	15 (msb) 8	**7 (lsb) 0**	
	Color space	Pixel encoding mode	
	Bits per index	Bits per Primary #1	
	Bits per primary #2	Bits per Primary #3	
	Encoding for primary # 1 R (most significant word)**		
	Encoding for primary # 1 R (least significant word)**		
	Encoding for primary # 1 G (msw)		
	Encoding for primary # 1 G (lsw)		
	Encoding for primary # 1 B (msw)		
	Encoding for primary # 1 B (lsw)		
	Encoding for primary # 2 R (msw)		
	Encoding for primary # 2 R (lsw)		
	Encoding for primary # 2 G (msw)		
	Encoding for primary # 2 G (lsw)		
	Encoding for primary # 2 B (msw)		
	Encoding for primary # 2 B (lsw)		
	Encoding for primary # 3 R (msw)		
	Encoding for primary # 3 R (lsw)		
	Encoding for primary # 3 G (msw)		
	Encoding for primary # 3 G (lsw)		
	Encoding for primary # 3 B (msw)		
	Encoding for primary # 3 B (lsw)		
	Minimum primary #1 value (msw)		

	Minimum primary #1 value (lsw)	
	Maximum primary #1 value (msw)	
	Maximum primary #1 value (lsw)	
	Minimum primary #2 value (msw)	
	Minimum primary #2 value (lsw)	
	Maximum primary #2 value (msw)	
	Maximum primary #2 value (lsw)	
	Minimum primary #3 value (msw)	
	Minimum primary #3 value (lsw)	
	Maximum primary #3 value (msw)	
	Maximum primary #3 value (lsw)	
	x Chromaticity for red primary (msw)	
	x Chromaticity for red primary (lsw)	
	y Chromaticity for red primary (msw)	
	y Chromaticity for red primary (lsw)	
	x Chromaticity for green primary (msw)	
	x Chromaticity for green primary (lsw)	
	y Chromaticity for green primary (msw)	
	y Chromaticity for green primary (lsw)	
	x Chromaticity for blue primary (msw)	
	x Chromaticity for blue primary (lsw)	
	y Chromaticity for blue primary (msw)	
	y Chromaticity for blue primary (lsw)	
	x Chromaticity for white point (msw)	
	x Chromaticity for white point (lsw)	
	y Chromaticity for white point (msw)	

	y Chromaticity for white point (lsw)	
	Gamma for red primary (msw)	
	Gamma for red primary (lsw)	
	Gain for red primary (msw)	
	Gain for red primary (lsw)	
	Gamma for green primary (msw)	
	Gamma for green primary (lsw)	
	Gain for green primary (msw)	
	Gain for green primary (lsw)	
	Gamma for blue primary (msw)	
	Gamma for blue primary (lsw)	
	Gain for blue primary (msw)	
	Gain for blue primary (lsw)	

** Floating point format

Examples of CID Usage

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

CID (data , data , ...)

Device RGB (dRGB) or Device CMY (dCMY)

Short Form

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

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

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

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

}
```

Long Form

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


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

CIE L*a*b*

Short Form

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

Long Form

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

Non-Linear SMPTE RGB, 2.2 Gamma, 1.0 Gain

Short Form

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

Long Form

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

Non-Linear Sony Trinitron

Short Form

<< not applicable >>

Long Form

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

YUV Chrominance-Luminance Space

Short Form

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

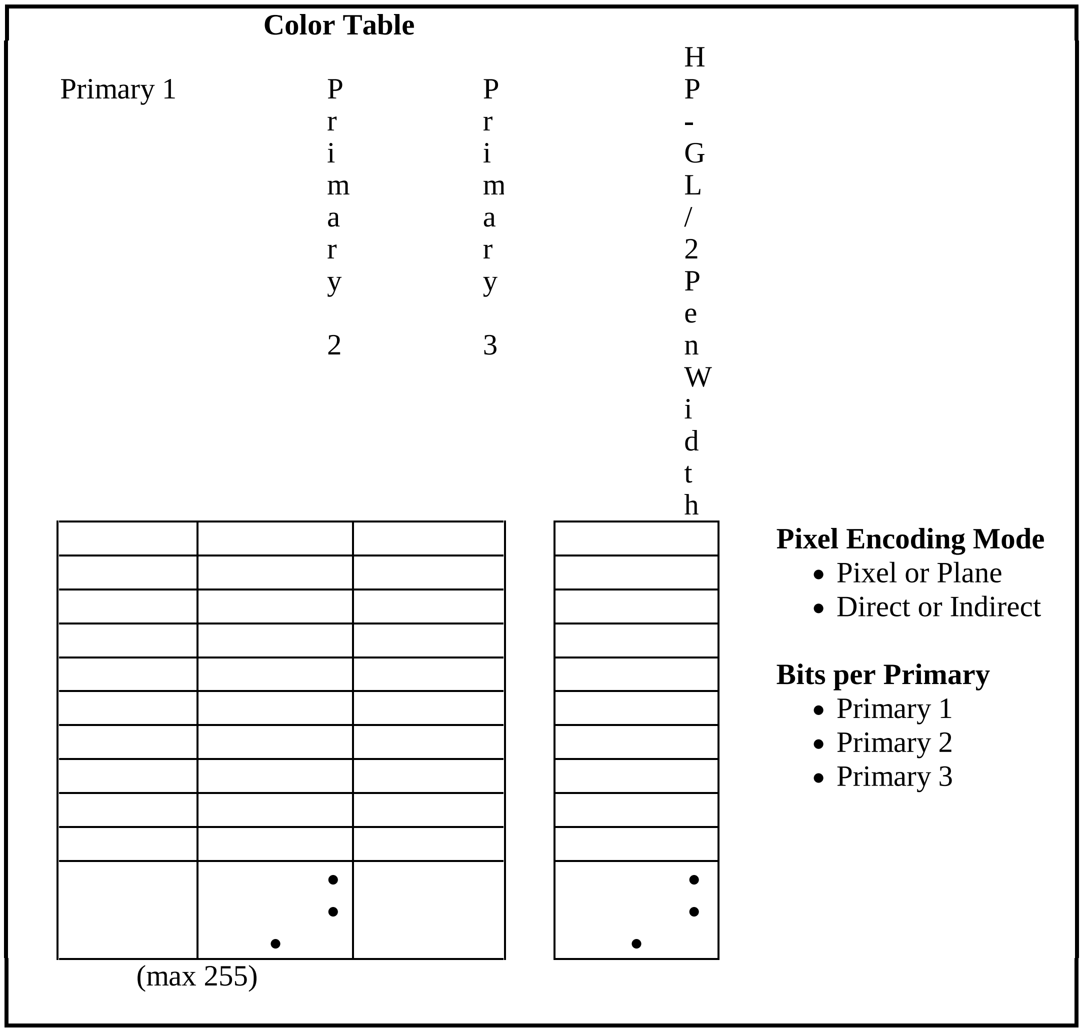
YUV Chrominance-Luminance with Sony Trinitron

Long Form

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

14.4 Palette Operations

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



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

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

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

Default palettes are created by all the PCL color modes (Black and White, Simple Color, PCL Imaging, and HP-GL/2 Imaging). The active palette may be modified in the PCL and HP-GL/2

Imaging modes, but not in the Simple Color or Black and White modes. The active palette is automatically transferred when switching between PCL and HP-GL/2 contexts.

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

- Palette ID
- Palette stack

Simple Color Palettes

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

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

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

Black and White (value = 1)

Index	**Color**
0	White
1	Black

RGB Palette (value = 3)

	Color
0	Black
1	Red
2	Green

CMY Palette (value = -3)

Index	**Color**
0	White
1	Cyan
2	Magenta

3	Yellow
4	Blue
5	Magenta
6	Cyan
7	White

3	Blue
4	Yellow
5	Green
6	Red
7	Black

4-Plane KCMY (value = -4)

	Black Pen	Color Pen
	White	White
	Black	White
	White	Cyan
	Black	Cyan
	White	Magenta
	Black	Magenta
	White	Blue
	Black	Blue
	White	Yellow
	Black	Yellow
	White	Green
	Black	Green

	White	Red
	Black	Red
	White	Black
	Black	Black

CID Color Palettes

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

Device RGB Palettes

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

Bits/Index = 1

Index	Color
0	White
1	Black

Bits/Index = 2

	Color
0	Black
1	Red
2	Green
3	White

Bits/Index = 3 through 8

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

Device CMY and Device-Independent Palettes

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

Bits/Index = 1

Index	Color
0	White
1	Black

Bits/Index = 2

0	
1	

Bits/Index = 3 through 8

Index	Color
0	White
1	Cyan

2	
3	

2	Magenta
3	Blue
4	Yellow
5	Green
6	Red
7	Black
n > 7	Black

HP-GL/2 Palettes

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

Two Pens

Pen Number	Color
0	White
1	Black

Four Pens

Pen Number	Color
0	White
1	Black
2	Red
3	Green

Eight Pens

Pen Number	
0	
1	
2	
3	
4	
5	
6	

7	
n >7	

Default HP-GL/2 Palettes

Palette Configuration

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

Palette Configuration ***Esc * d # w/W***

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

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

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

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

This command:

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

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

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

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

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

	Color
	White
	Black
	Red
	Green
	Yellow
	Blue
	Magenta
	Cyan
	Black

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

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

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

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

Byte 0: Color Space

Value = 1 Device Dependent RGB

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

Byte 1: Format

Value = 1 Unsigned 8-bit Integer Data

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

Byte 2: Last Palette Entry Index

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

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

DEVICE NOTE: DJ850C supports only 255.

Byte 3: Minimum Range Value for Red Component

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

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

DEVICE NOTE: DJ850C supports only 0.

Byte 4: Maximum Range Value for Red Component

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

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

DEVICE NOTE: DJ850C supports only 255.

Byte 5: Minimum Range Value for Green Component

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

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

DEVICE NOTE: DJ850C supports only 0.

Byte #6: Maximum Range Value for Green Component

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

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

DEVICE NOTE: DJ850C supports only 255.

Byte #7: Minimum Range Value for Blue Component

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

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

DEVICE NOTE: DJ850C supports only 0.

Byte #8: Maximum Range Value for Blue Component

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

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

DEVICE NOTE: DJ850C supports only 255.

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

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

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

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

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

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

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

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

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

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

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

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

Programming the Palette

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

Color Component One *Esc * v # A*

Specifies the first component of any new palette color entry.

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

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

Color Component Two *Esc * v # B*

Specifies the second component of any new palette color entry.

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

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

Color Component Three *Esc * v # C*

Specifies the third component of any new palette color entry.

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

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

Assign Color Index *Esc * v # I*

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

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

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

Saving Palettes

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

Push / Pop Palette *Esc * p # P*

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

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

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

Pushing a palette saves the following parameters:

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

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

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

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

DEVICE NOTE: PJ XL300 limits stack size to 256.

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

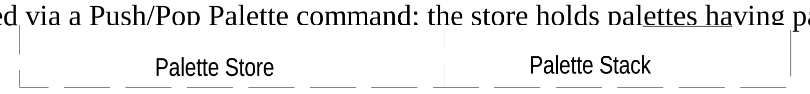
A reset or PJL entry empties the stack and overwrites the current palette with a non-programmable black and white palette. The *IN* and *DF* commands overwrite the current palette with the default HP-GL/2 palette, but have no effect on the palette stack.

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

Managing Palettes by ID

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

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



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

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

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

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

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

Select Palette *Esc & p # s/S*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Palette Control *Esc & p # c/C*

Provides a mechanism for copying and deleting palettes.

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

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

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

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

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

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

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

14.5 Foreground Color

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

Foreground Color *Esc * v # s/S*

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

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

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

Foreground color affects the following PCL page marking primitives.

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

The following are not affected:

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

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

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

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

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

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

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

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

14.6 Halftone Algorithms

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

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

Render Algorithm *Esc * t # j/J*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

14.7 User-Defined Dithers

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Format

This byte should be set to 0.

Number of Planes

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

Height and Width

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

Data Bytes

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

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

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

Multiple Dither Matrices

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

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

Example

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

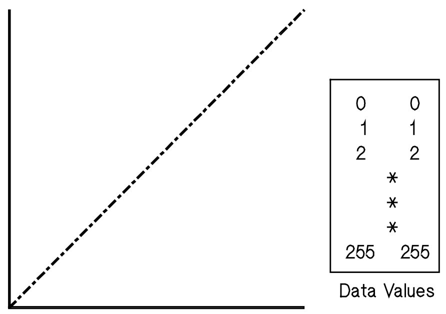
Byte	**15(msb) 8**	**7 (lsb) 0**	**Byte**
0	0	1	1
2	0	4	3
4	0	4	5
6	B0	B1	7
8	B2	B3	9
	*		
	*		
20	B14	B15	21

14.8 Color Lookup Tables

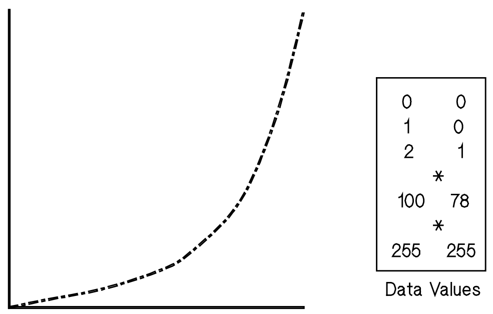
Color lookup tables, which map input data into a new output range based on point-by-point conversions, can modify input data for both device-dependent and device-independent color spaces. A lookup table is specified for each primary. Uses include:

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

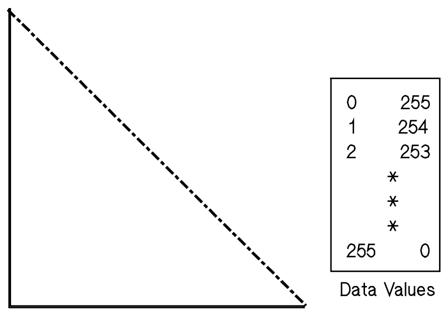
Like the CID command, the first byte of the data field identifies the color space to which the lookup tables will be applied. These tables specify on a point-per-point basis a transformation from an input space of 0...255 into an output space of 0...255. The following diagram illustrates the concept:



Unity Color Lookup Table



Gamma Color Lookup Table



Inversion Color Lookup Table

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

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

Enables and specifies color lookup tables.

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

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

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

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

	15 (msb) 8	**7 (lsb) 0**	
	Color Space	Reserved Data Field	
	Color Component 1, Index 0	Color Component 1, Index 1	
	Color Component 1, Index 2	Color Component 1, Index 3	
	Color Component 1, Index 254	Color Component 1, Index 255	
	Color Component 2, Index 0	Color Component 2, Index 1	
	Color Component 2, Index 2	Color Component 2, Index 3	
	Color Component 2, Index 254	Color Component 2, Index 255	
	Color Component 3, Index 0	Color Component 3, Index 1	
	Color Component 3, Index 2	Color Component 3, Index 3	
	Color Component 3, Index 254	Color Component 3, Index 255	

Byte # 0: Color Space

	Color Space

	Device RGB
	Device CMY
	Colorimetric RGB Spaces
	CIE L*a*b*
	Luminance-Chrominance Spaces

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

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

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

14.9 Gamma Correction

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

Gamma Correction *Esc * t # i/I*

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

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

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

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

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

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

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

14.10 Viewing Illuminant

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

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

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

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

The binary data field is formatted as follows:

	15 (msb) 8	**7 (lsb) 0**	
	x chromaticity white point (lsw)		
	x chromaticity white point (msw)		
	y chromaticity white point (lsw)		
	y chromaticity white point (msw)		

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

	30 23	22 0
	Exponent	Fraction

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

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

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

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

14.11 Page Media Color Commands

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

- Monochrome Print Mode
- Finish Mode

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

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

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

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

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

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

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

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

Finish Mode *Esc & b # f/F*

Determines the finish applied to printed pages.

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

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

Chapter 15: Configure Raster Data

Introduction

The Configure Raster Data (CRD) command performs many of the same functions as the Configure Image Data (CID) command described in Chapter 14. The two commands are not mutually exclusive, but no device has yet chosen to support both.

The CRD command currently has four different forms, determined by the format number in the first byte of binary data following the command. The formats are numbered from 2 to 5 (Format 1 is obsolete). Each of these forms configures the raster data differently.

Format Number	Type of Data
1	Complex Direct Planar (obsolete)
2	Complex Direct Planar
3	RGB Direct by Pixel
4	Indexed by Pixel
5	Mixed Monochrome and Indexed by Pixel

Format 2 configures for direct planar raster transfers. The data is sent by planes, with each plane supplying one bit for each pixel in row. Pixel color is specified directly (i.e., independent of a palette). Format 2 also specifies the horizontal resolution, vertical resolution, and the number of intensity levels for each component.

Format 3 configures for direct RGB pixel-by-pixel raster transfers. Each pixel is specified directly (i.e., independent of a palette). The horizontal and vertical resolution of the source pixels is also specified.

Format 4 configures for indexed pixel-by-pixel raster transfers. The bits for each pixel form an index number to a palette entry, which is the color of the pixel. All the bits for each pixel are sent before any bits are sent for the next pixel. The horizontal and vertical resolution of the source pixels is also specified.

Format 5 configures for an indexed pixel-encoded component and a direct monochrome component. The indexed data forms an index into the current palette for each pixel to be printed.

Configure Raster Data (CRD) *Esc * g # w/W*

The Configure Raster Data (CRD) command configures the device for complex raster data transfers.

Value(#)	=	Number of data bytes
Default	=	NA
Range	=	6 to 2^{32} - 1

DEVICE NOTE: DeskJet 520, 560C and 850C support a maximum range of 32767.

DEVICE NOTE: DeskJet 850C locks out Simple Color (*Esc*r#U*) and Raster Resolution (*Esc*t#R*) after a valid CRD until the receipt of an *EscE* or equivalent device reset, or a CRD command with a value field of 0 (*Esc*g0W*). A CRD command with a 0 value field unlocks Simple Raster (*Esc*r#U*) and Raster Resolution (*Esc*t#R*), resets the horizontal and vertical resolutions to 75 ppi, and resets the raster mode to one-bit direct (monochrome).

The data field must contain byte-aligned binary data, not ASCII. Unsupported or out-of-range values or other invalid configurations cause the entire command to be ignored and the data bytes to be discarded. Extra bytes are discarded. Value-field signs are ignored.

CRD provides raster configurations having one or more of the following characteristics:

- Horizontal resolution differs from vertical resolution.
- Color components may be specified at different resolutions.
- Each component may be specified at a multiple intensity level.
- Indexed raster data is separate from palette configuration.
- Direct raster data does not affect the palette configuration.
- Mixed indexed raster data and monochrome data for sleek top to bottom printing.

Reset (*EscE*), Simple Color (*Esc*r#U*), and CID (*Esc*v#W*) override a CRD configuration.

Raster resolution is determined by the most recent CRD even if it is of another format, or by the Raster Resolution (*Esc*t#R*) command. A Raster Resolution command issued after CRD:

- Sets horizontal and vertical resolutions for each component to the specified resolution.
- Matches the strip height for each component to the new resolution.
- Establishes a pixel unit size for source raster height and width.
- Zeros the seed rows.
- Sets the number of levels for each component to 2.
- Has no effect on the number of expected components (raster indexing mode is unchanged).

The lowest CRD horizontal resolution determines the horizontal pixel size for the Source Raster Width (*Esc*r#S*), to which all the planes of all the rows for all components are zero-filled or clipped. Similarly, the lowest CRD vertical resolution determines the vertical pixel size for Source Raster Height (*Esc*r#T*), which defines raster area vertical height.

Missing data is zero-filled if Raster Resolution is changed or a Y Offset (*Esc*b#Y*) command is received after only part of the data defining a strip is sent.

NOTE: Seed Row Source (*Esc*b#S*) should not be used for CRD raster data; the result is undefined.

As usual, the Transfer Raster commands (*Esc*b#W*, *Esc*b#V*) download raster data, and the Compression Method (*Esc*b#M*) command defines the compression rules.

As shown below, CRD has different forms based on the format byte. Each form of CRD shares only the first byte, the format number, with the other forms of CRD. Note that Format 1 is obsolete. Format 2 is identical to Format 1 except in the four-component KCMY case. Format 2 sends the data in component order: cyan, magenta, yellow, black; Format 1 sends black first. Format 2 should be used in all future implementations.

DEVICE NOTE: Format 1 was used only on DeskJet 520 and DeskJet 560C and is being obsoleted in favor of Format 2, which should be used on all future implementations.

Format Number	**Type of Data**
1	Complex Direct Planar (obsolete)
2	Complex Direct Planar
3	RGB Direct by Pixel
4	Indexed by Pixel
5	Mixed Monochrome and Indexed by Pixel

Format 2 Complex Direct Planar

Format 2 configures for raster transfers where:

- Horizontal resolution differs from vertical resolution.
- Color components are specified at different resolutions.
- Multiple intensity levels are specified for one or more components.

The data for each color component is organized into strips containing one or more rows. The lowest vertical-resolution component determines the vertical displacement of the other components; higher vertical-resolution components require more rows than lower-resolution components to cover the same vertical displacement. The horizontal resolution determins the number of bytes required to define a row (e.g., 600 dpi rows contain 8 times as many bytes as 75 dpi rows). Higher horizontal and vertical resolutions must be multiples of lower horizontal and vertical resolutions, respectively. The CRD command is ignored for unsupported resolutions.

Byte	**15 (MSB)**	**7 (LSB) 0**	**Byte**
0	Format=2 (UBYTE)	Number of components (UBYTE)	1
2	Horizontal Resolution for Component 1 (UINT16)		3
4	Vertical Resolution for Component 1 (UINT16)		5
6	Number of Levels for Component 1 (UINT16)		7
. . .	...		. . .

6(n-1)+2	Horizontal Resolution for Component n (UINT16)	6(n-1)+3
6(n-1)+4	Vertical Resolution for Component n (UINT16)	6(n-1)+5
6(n-1)+6	Number of Levels for Component n (UINT16)	6(n-1)+7

Byte 0 Format #

Value = 2

Format 2 configures for direct monochrome, CMY, or KCMY data in planar format.

DEVICE NOTE: Only DeskJet 540, DeskJet 660C and DeskJet 850C implement format 2.

Byte 1 Number of Components

Value =	1	Monochrome (K)
=	3	Three components (CMY)
=	4	Four components (KCMY)

Specifies the number of expected components. When the value is 3, the first component is cyan, the second magenta, and the third yellow. When the value is 4, the first component is cyan, the second magenta, the third yellow, and the fourth black. The raster transfer commands (*Esc*b#V*, *Esc*b#W*) must use the same component ordering to send data.

DEVICE NOTE: DeskJet 540 supports 1 or 3 components; DeskJet 850C only supports 1 or 4.

Bytes 2&3, 8&9, etc Horizontal Resolution

Value = 1 to 65535

Specifies the horizontal resolution of each component in pixels per inch of the source pixel. The "horizontal" axis is determined by the current Raster Presentation (*Esc*r#/F*), Orientation (*Esc&l#O*), and Print Direction (*Esc&a#P*). Bytes 2 & 3 specify the horizontal resolution for component one, bytes 8 & 9 specify the horizontal resolution for component two, etc. Higher horizontal resolutions must be multiples of lower horizontal resolutions.

DEVICE NOTE: DeskJet 540 supports only 300 ppi cyan, magenta, and yellow horizontal resolutions. Black horizontal resolution may be 300 or 600 ppi.

DEVICE NOTE: On DeskJet 660C, cyan, magenta, and yellow horizontal resolutions must be equal and must be either 300 or 600 ppi. When the cyan, magenta, or yellow horizontal resolution is 600 ppi, the vertical resolution must be 300 ppi. Black horizontal resolution may be 300 or 600 ppi.

DEVICE NOTE: On DeskJet 850C, horizontal and vertical resolutions must be equal. Cyan, magenta, and yellow resolutions must be equal and must be either 150 or 300 ppi. Black resolution may be different from the CMY resolutions and may be 150, 300, or 600.

Bytes 4&5, 10&11, etc   Vertical Resolution

Value =   1 to 65535

Specifies the vertical resolution of each component of the source pixel in pixels per inch. The “vertical” axis is determined by the current Raster Presentation (*Esc*r#/F*), Orientation (*Esc&l#O*), and Print Direction (*Esc&a#P*). Bytes 4 & 5 specify the vertical resolution for component one, bytes 10 & 11 specify the vertical resolution for component two, etc. Higher vertical resolutions must be multiples of lower vertical resolutions.

DEVICE NOTE: DeskJet 540 supports only a vertical resolution of 300 ppi.

DEVICE NOTE: DeskJet 660C supports only 300 ppi cyan, magenta, and yellow vertical resolutions. Black vertical resolution may be 300 or 600 ppi. When the black vertical resolution is 300 ppi, the black horizontal resolution must also be 300 ppi.

DEVICE NOTE: On DeskJet 850C, horizontal and vertical resolutions must be equal. Cyan, magenta, and yellow resolutions must be equal and must be either 150 or 300 ppi. Black resolution may be different from the CMY resolutions and may be 150, 300, or 600.

Bytes 6&7, 12&13, etc   Number of Intensity Levels

Value =   2 to 255

Specifies the number of intensity or grayscale levels for each component. A level of 2 allows only two intensities, since a pixel is either ON or OFF. A level of 4 allows four intensities, etc. The lowest intensity is 0; the highest intensity is Number of Levels - 1.

Bytes 6 and 7 specify the number of levels for component one, bytes 12 and 13 specify the number of levels for component two, etc.

The number of planes expected for each row is the ceiling function of log base 2 of the number of levels. For example, three levels require two planes. The lowest order plane is transmitted first, and the highest order plane last for a given row.

DEVICE NOTE: DeskJet 540 and DeskJet 660C support only a level of 2.

DEVICE NOTE: DeskJet 850C supports levels of 2, 3, and 4 for 300 ppi components, and a level of 2 for 150 or 600 ppi components. CMY components must have equal levels. A black component may have either two or four levels. If Black is 600 ppi, the CMY components in a four-component case must be either 2 or 4 levels.

Acceptable Combinations of Parameters for Format 2

A = Number of Components
B = Horizontal Resolution of Component 1
C = Vertical Resolution of Component 1
D = Number of Intensity Levels of Component 1
E = Horizontal Resolution of Component 2
F = Vertical Resolution of Component 2
G = Number of Intensity Levels of Component 2
H = Horizontal Resolution of Component 3
I = Vertical Resolution of Component 3
J = Number of Intensity Levels of Component 3
K = Horizontal Resolution of Component 4
L = Vertical Resolution of Component 4
M = Number of Intensity Levels of Component 4

DEVICE NOTE: Acceptable combinations of parameters for DeskJet 540:

DEVICE NOTE: Acceptable combinations of parameters for DeskJet 660C:

DEVICE NOTE: Acceptable combinations of parameters for DeskJet 850C:

Example 1 of Format 2

Assume the following CRD command is sent:

Format	=	2
Number of components	=	4
Horizontal resolution component 1	=	300
Vertical resolution component 1	=	300
Levels for component 1	=	2
Horizontal resolution component 2	=	300
Vertical resolution component 2	=	300
Levels for component 2	=	2
Horizontal resolution component 3	=	300
Vertical resolution component 3	=	300
Levels for component 3	=	2
Horizontal resolution component 4	=	300
Vertical resolution component 4	=	300
Levels for component 4	=	2

The vertical displacement of all four strips is 1/300 of an inch, since this is the displacement of the component with the lowest vertical resolution (300 ppi).

Component 1 consists of one row containing one plane of black. Only one row is needed at 300 ppi vertical resolution to cover 1/300 of an inch of vertical displacement. And only one plane is needed to encode two intensity levels ($\log_2 2=1$).

Component 2 consists of one row containing one plane of cyan; component 3 consists of one row with one plane of magenta; and component 4 consists of one row with one plane of yellow.

Transfer Raster by Plane (*Esc*b#V*) is used to send all the planes of all the rows except the last plane of the last row within a strip, which uses Transfer Raster by Row/Block (*Esc*b#W*). As shown below, unlike Format 1, components are sent in order from 1 to 4.

Component 1	*Esc*b#V*	300 ppi Black data
Component 2	*Esc*b#V*	300 ppi Cyan data
Component 3	*Esc*b#V*	300 ppi Magenta data
Component 4	*Esc*b#W*	300 ppi Yellow data

Example 2 of Format 2

This example illustrate Format 2 in which one component is sent at a different resolution. This shows how the vertical displacement of each component of a strips is determined by the minimum vertical resolution of all components.

Assume the following CRD command is sent:

Format	=	2
Number of components	=	4
Horizontal resolution component 1	=	600
Vertical resolution component 1	=	600
Levels for component 1	=	2
Horizontal resolution component 2	=	300
Vertical resolution component 2	=	300
Levels for component 2	=	2
Horizontal resolution component 3	=	300
Vertical resolution component 3	=	300
Levels for component 3	=	2
Horizontal resolution component 4	=	300
Vertical resolution component 4	=	300
Levels for component 4	=	2

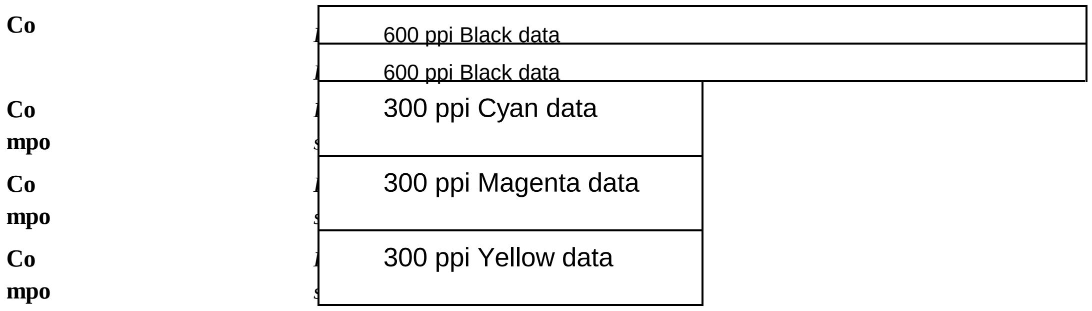
In this example, the vertical displacement of all four strips is 1/300 of an inch, which is the displacement of the component with the lowest vertical resolution (300 ppi).

Component 1 consists of two rows which each contain one plane of black. Two 600 ppi rows of data are required to fill a 1/300 of an inch strip. Only one plane is needed to encode two intensity levels ($\log_2 2=1$).

Component 2 consists of one row containing one plane of cyan. Only one row is needed at 300 ppi vertical resolution to cover 1/300 of an inch of vertical displacement. Only one plane is needed to encode two intensity levels ($\log_2 2=1$).

Component 3 consists of one row containing one plane of magenta; and component 4 consists of one row containing one plane of yellow.

Transfer Raster by Plane (*Esc*b#V*) is used to send all the planes of all the rows except the last plane of the last row within a strip, which uses Transfer Raster by Row/Block (*Esc*b#W*). The least significant plane of a row is sent first. Components are sent in order, from 1 to 4. Note that, as shown below, the 600 ppi planes contain twice the data of the 300 ppi planes.



Example 3 of Format 2

In this example one component has a different resolution, and the lower resolution components have more than two intensity levels. Assume the following CRD command is sent:

Format	=	2
Number of components	=	4
Horizontal resolution component 1	=	600
Vertical resolution component 1	=	600
Levels for component 1	=	2
Horizontal resolution component 2	=	300
Vertical resolution component 2	=	300
Levels for component 2	=	4
Horizontal resolution component 3	=	300
Vertical resolution component 3	=	300
Levels for component 3	=	4
Horizontal resolution component 4	=	300
Vertical resolution component 4	=	300
Levels for component 4	=	4

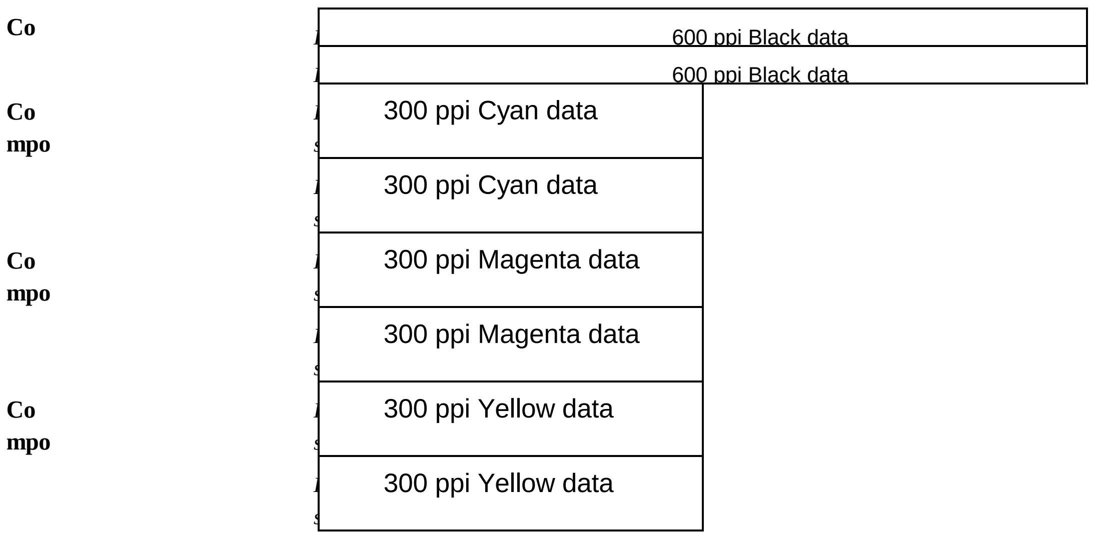
The vertical displacement of all four strips is 1/300 of an inch, since that is the displacement of the component with the lowest vertical resolution (300 ppi).

Component 1 consists of two rows, each containing one black plane. Two rows of 600 ppi data are required to fill a 1/300" vertical strip. One plane is needed for two intensity levels ($\log_2 2=1$).

Component 2 consists of one row containing two cyan planes. Only one row of 300 ppi data is required to fill a 1/300" vertical strip. Two planes are needed for four intensity levels ($\log_2 4=2$).

Components 3 and 4 have the same encoding as component 2. Component 3 consists of one row containing two magenta planes; component 4 consists of one row containing two yellow planes.

All the planes of all the rows use Transfer Raster by Plane (*Esc*b#V*), except the last plane of the last row within a strip, which uses Transfer Raster by Row/Block (*Esc*b#W*).



Example 4 of Format 2

This example illustrates the flexibility of Format 2: all components have different resolutions and a different number of intensity levels. There is no printer that implements this complex combination of parameters, but this example shows how the sequence could meet a wide variety of future needs. Assume the following CRD command is sent:

Format	=	2
Number of components	=	4
Horizontal resolution component 1	=	300
Vertical resolution component 1	=	600
Levels for component 1	=	2
Horizontal resolution component 2	=	150
Vertical resolution component 2	=	75
Levels for component 2	=	16
Horizontal resolution component 3	=	75
Vertical resolution component 3	=	150
Levels for component 3	=	8
Horizontal resolution component 4	=	100
Vertical resolution component 4	=	300
Levels for component 4	=	4

In this example, the vertical displacement of all four strips is 1/75 of an inch, since that is the displacement of the component with the lowest vertical resolution (75 ppi).

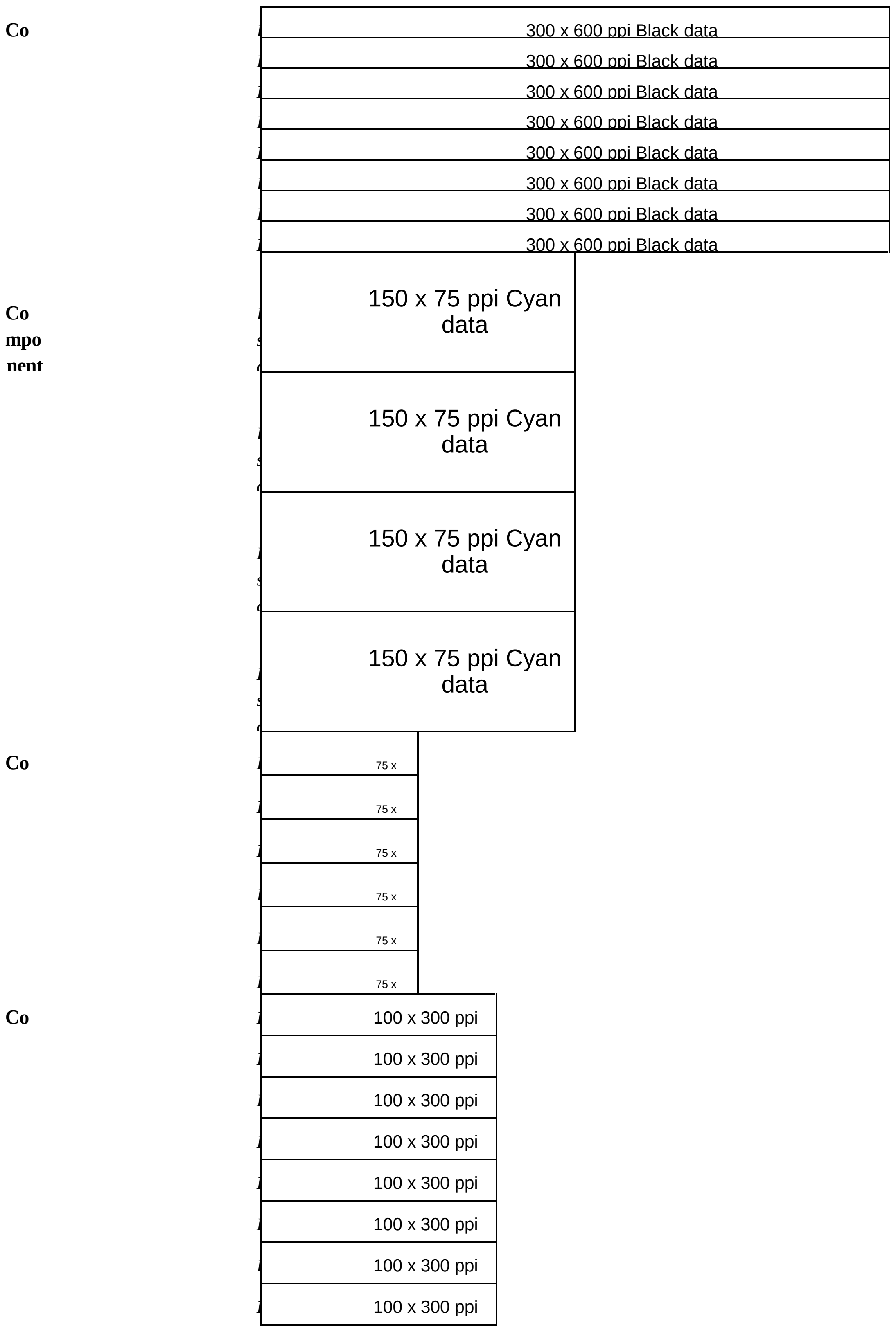
Component 1 consists of eight single-plane rows of black. Eight rows of 600 ppi vertical-resolution data are required to fill a 1/75" strip. Only one plane is needed to encode two intensity levels ($\log_2 2=1$).

Component 2 consists of one row containing four planes of cyan. Only one row is needed at 75 ppi vertical resolution to fill 1/75" vertical displacement. Four planes are needed to encode 16 intensity levels ($\log_2 16=4$).

Component 3 consists of two rows, each containing three planes of magenta. Two rows are needed at 150 ppi vertical resolution to fill a 1/75" strip. Three planes are needed to encode eight intensity levels ($\log_2 8=3$).

Component 4 consists of four rows, each containing two planes of yellow. Four rows are needed at 300 ppi vertical resolution to fill a 1/75" strip. Two planes are needed to encode four intensity levels ($\log_2 4=2$).

All the planes of all the rows use Transfer Raster by Plane (*Esc*b#V*) except the last plane of the last row within a strip, which uses Transfer Raster by Row/Block (*Esc*b#W*). As shown below, components are sent in order from 1 to 4. The least significant plane is sent first and the most significant plane last of the same row. Note that the 600 ppi planes contain eight times the data of the 75 ppi planes.


Co
300 x 600 ppi Black data
300 x 600 ppi Black data
300 x 600 ppi Black data
300 x 600 ppi Black data
300 x 600 ppi Black data
300 x 600 ppi Black data
300 x 600 ppi Black data
300 x 600 ppi Black data
Co
mpo
nent
150 x 75 ppi Cyan data
150 x 75 ppi Cyan data
150 x 75 ppi Cyan data
150 x 75 ppi Cyan data
Co
75 x
75 x
75 x
75 x
75 x
75 x
Co
100 x 300 ppi
100 x 300 ppi
100 x 300 ppi
100 x 300 ppi
100 x 300 ppi
100 x 300 ppi
100 x 300 ppi
100 x 300 ppi

Format 3 RGB Direct by Pixel

Configures for direct RGB pixel-by-pixel raster transfers. Raster transfer is independent of the palette.

	15 (MSB) **8**	**7 (LSB)** **0**	
0	Format=3 (UBYTE)	Number of Bits for Red (UBYTE)	1
2	Number of Bits for Green (UBYTE)	Number of Bits for Blue (UBYTE)	3
4	Horizontal Resolution (UINT16)		5
6	Vertical Resolution (UINT16)		7

Byte 0 Format

Value = 3

Format 3 configures for direct-by-pixel device-dependent RGB data.

DEVICE NOTE: Only DeskJet 850C implements format 3.

Byte 1 Number of Bits for Red

Value = 1 to 255

Specifies the number of bits necessary to completely specify the red component of a pixel when in direct device-dependent RGB raster mode. This in turn also specifies $2^{\text{Number of Bits for Red}}-1$ to be the maximum intensity level for a red component. The minimum intensity level is 0.

DEVICE NOTE: DeskJet 850C will support only a value of 8.

Byte 2 Number of Bits for Green

Value = 1 to 255

Specifies the number of bits necessary to completely specify the green component of a pixel when in direct device-dependent RGB raster mode. This in turn also specifies $2^{\text{Number of Bits for Green}}-1$ to be the maximum intensity level for a green component. The minimum intensity level is 0.

DEVICE NOTE: DeskJet 850C will support only a value of 8.

Byte 3 Number of Bits for Blue

Value = 1 to 255

Specifies the number of bits necessary to completely specify the blue component of a pixel when in direct device-dependent RGB raster mode. This in turn also specifies $2^{\text{Number of Bits for Blue}}-1$ to be the maximum intensity level for a blue component. The minimum intensity level is 0.

DEVICE NOTE: DeskJet 850C will support only a value of 8.

Bytes 4&5 Horizontal Resolution

Value = 1 to 65535

Specifies horizontal resolution in pixels per inch of the source pixel. "Horizontal" is determined by the current Raster Presentation (*Esc*r#/F*), Orientation (*Esc&l#O*), and Print Direction (*Esc&a#P*).

DEVICE NOTE: DeskJet 850C only supports only a value of 300.

Bytes 6&7 Vertical Resolution

Value = 1 to 65535

Specifies the vertical resolution in pixels per inch of the source pixel. The "vertical" axis is determined by the current Raster Presentation (*Esc*r#F*), Orientation (*Esc&l#O*), and Print Direction (*Esc&a#P*).

DEVICE NOTE: DeskJet 850C only supports a value of 300.

Example 1 of Format 3

Assume the following CRD command is sent:

Format	=	3
Number of Bits for Red	=	8
Number of Bits for Green	=	8
Number of Bits for Blue	=	8
Horizontal Resolution	=	300
Vertical Resolution	=	300

It takes eight bits (one byte) to form an R, G, or B component, i.e, a total of 24 bits (three bytes) to directly specify each pixel. Since this is pixel encoding rather than planar, each pixel in a row is fully specified before any bits are sent for the next pixel.

The data is sent in the following way for a 20-pixel wide by 10-pixel high block (total of 200 pixels):

pixel 1, pixel 2, pixel 3, pixel 4, ...pixel 200

If pixel 1 components are red = 255, green = 0, blue = 3 and each component is eight bits, the data for the first pixel is formatted as 111111110000000000000011. If pixel 2 components are red = 16, green = 128, blue = 6, the data stream for pixels 1 and 2 would appear as:

```
11111111 00000000 00000011 00001000 10000000 00000110
  (255)      (0)      (3)      (16)     (128)     (6)
```

Format 4 Indexed by Pixel

Configures for indexed pixel-by-pixel raster transfers. The bits for each pixel form an index number to a palette entry, which is the color of the pixel. All the bits for each pixel are sent before any bits are sent for the next pixel. Palette configuration, performed by Configure Image Data (*Esc*v#W*) or Palette Configuration (*Esc*d#W*), is independent of raster transfer.

Byte	**15 (MSB) 8**	**7 (LSB) 0**	**Byte**
0	Format=4 (UBYTE)	Number of Index Bits (UBYTE)	1
2	Horizontal Resolution (UINT16)		3
4	Vertical Resolution (UINT16)		5

Byte 0 Format

Value = 4

Format 4 configures for indexed data, where the data is indexed by pixel.

DEVICE NOTE: Only DeskJet 850C implements format 4.

Byte 1 Number of Index bits

Value = 1 to 255

Specifies the number of bits required to form an index into the palette. The *Number of Index Bits* field may limit the number of possible selectable palette entries to be less than the palette size. Palette size is set by either Configure Image Data (*Esc*v#W*) or Palette Configuration (*Esc*d#W*). If a specified index exceeds the palette size, the modulo of the palette size is used to achieve an index within the palette.

DEVICE NOTE: DeskJet 850C only supports a value of 8.

Bytes 2&3 Horizontal Resolution

Value = 1 to 65535

Specifies the source pixel's horizontal resolution in pixels per inch. The “horizontal” axis is determined by the current Raster Presentation (*Esc*r#F*), Orientation (*Esc&l#O*), and Print Direction (*Esc&a#P*).

DEVICE NOTE: DeskJet 850C only supports a value of 300.

Bytes 4&5 Vertical Resolution

Value = 1 to 65535

Specifies the vertical resolution of the source pixel in pixels per inch. The “vertical” axis is determined by the current Raster Presentation (*Esc*r#F*), Orientation (*Esc&l#O*), and Print Direction (*Esc&a#P*).

DEVICE NOTE: DeskJet 850C only supports a value of 300.

Example 1 of Format 4

Assume the following CRD command is sent:

Format	=	4
Number of Index bits	=	8
Horizontal Resolution	=	300
Vertical Resolution	=	300

In this example, eight bits are required to form a palette index. If the palette was configured for 256 entries, each byte in the row describes a pixel by forming an palette entry index. Each pixel will have a resolution of 300 x 300 ppi.

The palette index for each pixel in a row is fully specified before any bits are sent for the next pixel's palette index. The *Number of Index Bits* field determines index data boundaries. The data would be sent in the following way for a block width of 20 pixels and height of 10 pixels (total of 200 pixels):

index for pixel 1, index for pixel 2, index for pixel 3, index 4 ...index for pixel 200

A *Nmber of Index Bits* value of 8 implies that the data stream is parsed in 8 bit segments. The following data would specify that the first three palette entries have values 16, 0 and 7.

```
00010000 00000000 00000111
  (16)      (0)       (7)
```

Example 2 of Format 4

Assume the following CRD command is sent:

Format	=	4
Number of Index bits	=	4
Horizontal Resolution	=	150
Vertical Resolution	=	150

In this example, it takes 4 bits or one nibble to form an index into the palette. If the palette was configured for 256 entries, each byte of the row describes a pixel by forming an index to a palette entry. Each pixel will have a resolution of 150 x 150 pixels per inch.

The data is sent in the following way for a block width of 20 pixels and height of 10 pixels (total of 200 pixels):

index for pixel 1, index for pixel 2, index for pixel 3, index 4, ...index for pixel 200

A *Number of Index Bits* value of 4 implies that the data stream is parsed in 4-bit segments. The following data specifies the first 6 pixels, with pixel values of 1, 0, 0, 0, 0, and 7.

```
0001 0000 0000 0000 0000 0111
 (1)  (0)  (0)  (0)  (0)  (7)
```

If the palette had only 12 entries, a value of 1111 (15) would specify (index value modulo palette size - 15MOD12), or palette entry 3. This recalculation occurs whenever the palette size is smaller than the maximum number that can be specified with the number of index bits.

Format 5 Mixed Monochrome and Indexed by Pixel

Configures for an indexed pixel-encoded component and a direct monochrome component. The indexed data forms an index into the current palette for each pixel to be printed. The bits for each pixel define that pixel by forming a palette index number to a palette entry. Each pixel in a row is fully specified before bits are sent for the next pixel. Either Configure Image Data (*Esc*v#W*) or Palette Configuration (*Esc*d#W*) may configure the palette. Format 5 may be used for sleek top-to-bottom printing.

	15 (MSB) **8**	**7 (LSB)** **0**	
	Format=5 (UBYTE)	Number of Index bits (UBYTE)	
	Indexed Horizontal Resolution (UINT16)		
	Indexed Vertical Resolution (UINT16)		
	Monochrome Horizontal Resolution (UINT16)		
	Monochrome Vertical Resolution (UINT16)		
	Data Organization (UBYTE)		

Byte 0 Format

Value = 5

Format 5 configures for combined by pixel-encoded indexed device-dependent RGB and by pixel direct monochrome data.

DEVICE NOTE: Only DeskJet 850C implements format 5.

Byte 1 Number of Index bits

Value = 1 to 255

Specifies the number of bits required to form an index into the palette. The number of bits may limit the number of selectable palette entries to less than the palette size. If an index is sent that exceeds the size of the palette, the modulo of the palette size is used to achieve an index within the palette.Palette size is set by either Configure Image Data (*Esc*v#W*) or Palette Configuration (*Esc*d#W*).

DEVICE NOTE: DeskJet 850C only supports a value of 8.

Bytes 2&3 Indexed Horizontal Resolution

Value = 1 to 65535

Specifies the horizontal resolution in pixels per inch of the source pixel for the indexed component. The “horizontal” axis is determined by the current Raster Presentation (*Esc*r#f/F*), Orientation (*Esc&l#o/O*), and Print Direction (*Esc&a#p/P*).

DEVICE NOTE: DeskJet 850C only supports a value of 300.

Bytes 4&5 Indexed Vertical Resolution

Value = 1 to 65535

Specifies the vertical resolution in pixels per inch of the source pixel for the indexed component. The “vertical” axis is determined by the current Raster Presentation (*Esc*r#f/F*), Orientation (*Esc&l#o/O*), and Print Direction (*Esc&a#p/P*).

DEVICE NOTE: DeskJet 850C only supports a value of 300.

Bytes 6&7 Monochrome Horizontal Resolution

Value = 1 to 65535

Specifies the horizontal resolution in pixels per inch of the monochrome source pixel. The “horizontal” axis is determined by the current Raster Presentation (*Esc*r#f/F*), Orientation (*Esc&l#o/O*), and Print Direction (*Esc&a#p/P*).

DEVICE NOTE: DeskJet 850C only supports a value of 600.

Bytes 8&9 Monochrome Vertical Resolution

Value = 1 to 65535

Specifies the vertical resolution in pixels per inch of the monochrome source pixel. The “vertical” axis is determined by the current Raster Presentation (*Esc*r#f/F*), Orientation (*Esc&l#o/O*), and Print Direction (*Esc&a#p/P*).

DEVICE NOTE: DeskJet 850C only supports a value of 600.

Byte 10 Data Organization

Value	=	1	Indexed first, then monochrome data, which is placed on top of indexed data
	=	2	Monochrome first, then indexed data, which is placed on top of monochrome data
	=	3	Monochrome first, then indexed. Monochrome is placed on top of indexed data.
	=	4	Indexed data first, then monochrome. Indexed is placed on top of monochrome.

Specifies how the raster scan lines are organized in the image data. A value of one means that the indexed color portion of the vertical strip is sent and processed first, followed by the monochrome portion. A value of two means the reverse. This setting is important because the temporal ordering of data causes the data sent last to be placed on top of the first data.

DEVICE NOTE: DeskJet 850C only supports a value of 1. DeskJet 860C will support 3.

Example 1 of Format 5

Assume the following CRD command is sent:

Format	=	5
Number of Index bits	=	8
Index Horizontal Resolution	=	300
Index Vertical Resolution	=	300
Monochrome Horizontal Resolution	=	600
Monochrome Vertical Resolution	=	600
Data Organization	=	1

The vertical displacement of both strips is 1/300", since that is the displacement of the component with the lowest vertical resolution (300 ppi).

A monochrome pixel has a 600 x 600 ppi resolution. The monochrome component consists of two rows, since two rows are needed at 600 ppi vertical resolution to fill 1/300" vertical displacement.

Each pixel of the indexed component has a resolution of 300 x 300 ppi.The indexed component consists of one row, since only one row is needed at 300 ppi vertical resolution to fill 1/300" of vertical displacement.

The *Number of Index Bits* field determines index data boundaries. In this example, eight bits form an index into the palette. Each byte in a row describes a pixel by forming an index to one of 256 palette entries. The index for each pixel in a row is fully specified before any bits are sent for the next pixel's index.

The data is sent in the following way for a 4-pixels wide by 2-pixels high block of 300 ppi indexed pixels (total of 8 pixels), and an 8-pixels wide by 4-pixels high block of 600 ppi monochrome pixels (total of 32 pixels):

```
index for pixel 1, index for pixel 2, index for pixel 3, index for pixel 4
monochrome data for pixels 1 to 8
monochrome data for pixels 9 to 16
index for pixel 5, index for pixel 6, index for pixel 7, index for pixel 8
monochrome data for pixels 17 to 24
monochrome data for pixels 25 to 32
```

Chapter 16: The Color Print Model

Contents of this Chapter

This chapter discusses the following PCL commands:

16.1 Introduction

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

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

The print model process consists of the following steps:

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

Definitions

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

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

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

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

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

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

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

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

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

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

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

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

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

16.2 Logical Operations

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

Operators

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

Operands

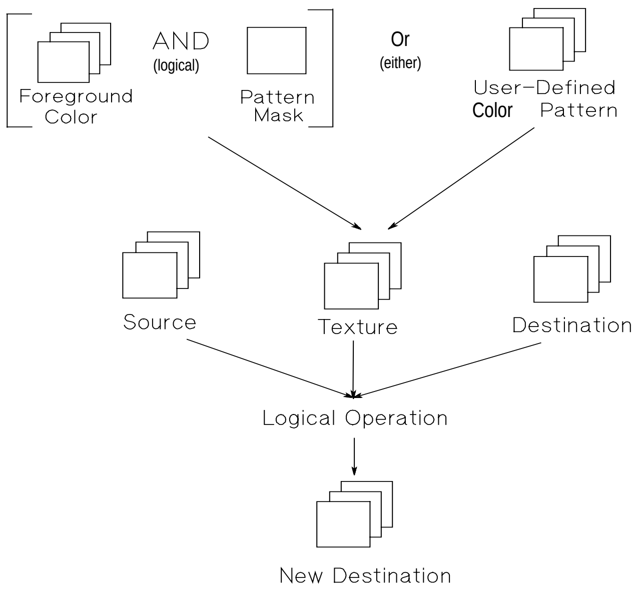
- Source objects — character cell, raster image, rule, HP-GL/2 vectors and polygons.
- Texture — foreground color + pattern mask, or color pattern (format 1).
- Destination — current page definition.

Operation

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

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

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



Logical Operation *Esc * l # o/O*

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

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

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

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

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

Transparency Interactions

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

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

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

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

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

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

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

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

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

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

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

NOT (texture XOR (source OR destination))

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

(texture | source)

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

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

Table of Logical Operations (ROPs)

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

S	=	Source	a	=	AND
T	=	Texture	o	=	OR

D	=	Destination	n	=	NOT
			x	=	EXCLUSIVE OR

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

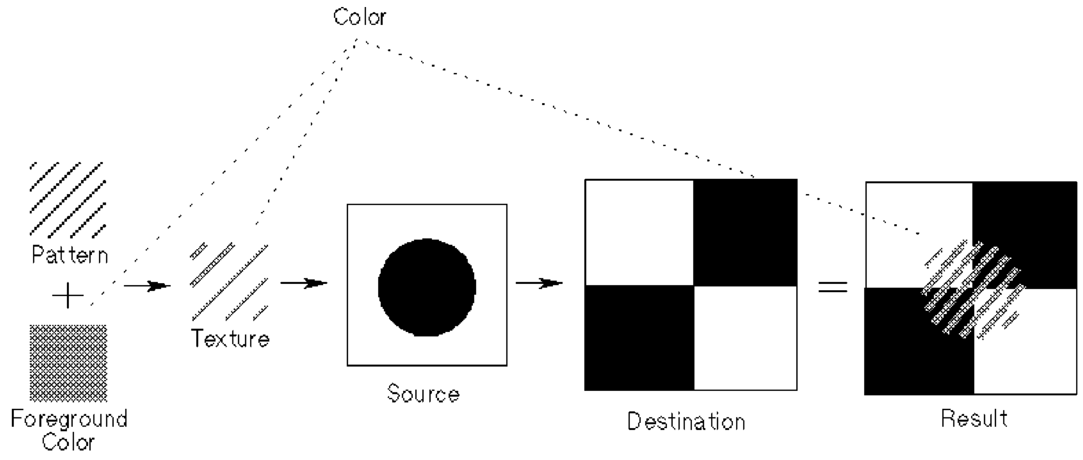
Input Value	Boolean Function	Input Value	Boolean Function
128	DTSaa	192	TSa
129	STxDSxon	193	STDSnaoxn
130	DTSxna	194	STDSonoxn
131	STDSnoaxn	195	TSxn
132	SDTxna	196	STDnoa
133	TDSTnoaxn	197	STDSxoxn
134	DSTDSoaxx	198	SDTnax
135	TDSaxn	199	TSDToaxn
136	DSa	200	SDToa
137	SDTSnaoxn	201	STDoxn
138	DSTnoa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

16.3 The Default Print Model

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

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

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



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

16.4 Transparency Modes

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

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

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

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

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

Sets source transparency mode.

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

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

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

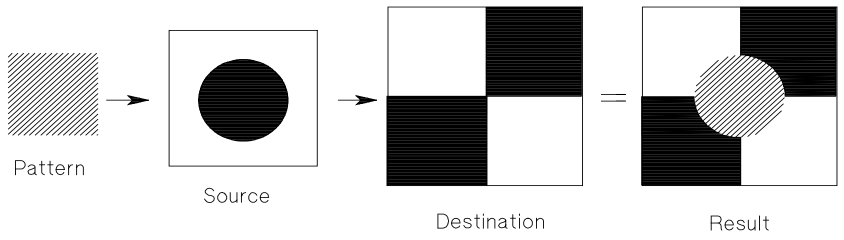
Sets pattern transparency mode.

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

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

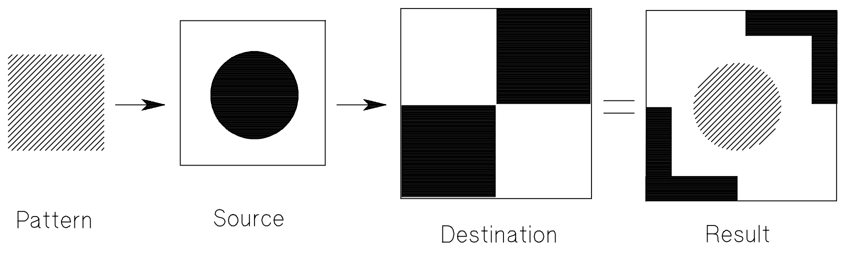
NOTE: For white rectangular area fills (rules), pattern transparency mode is always opaque.

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

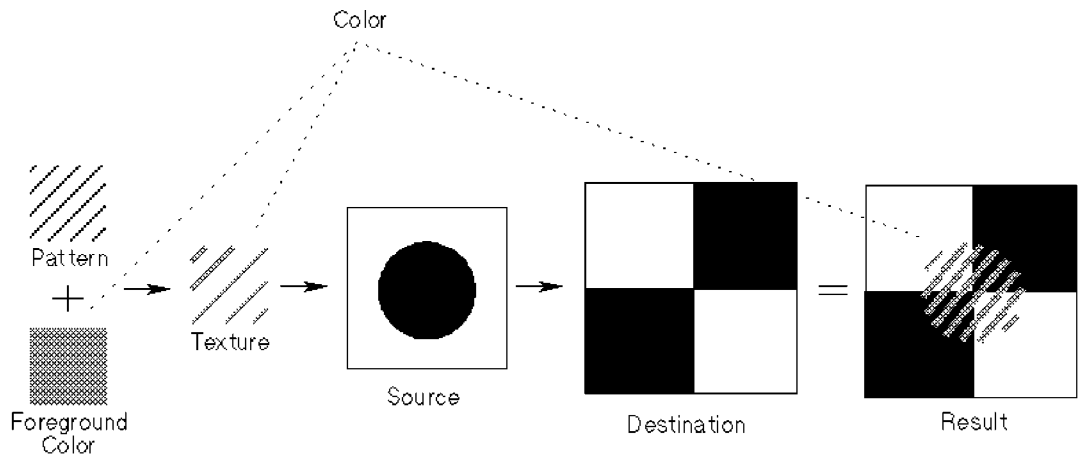


Source Transparency = 0 (Transparent)
Pattern Transparency = 1 (Opaque)

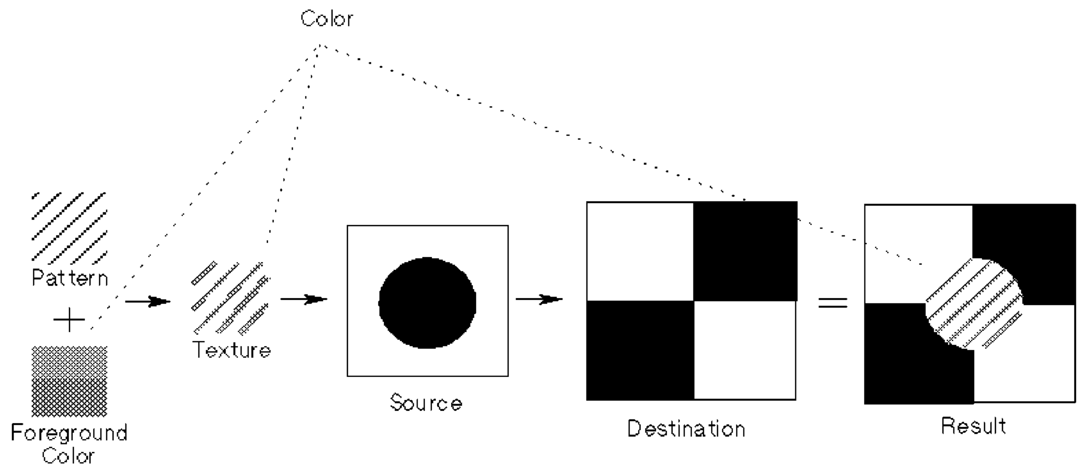
Source Transparency = 1 (Opaque)
Pattern Transparency = 0 (Transparent)



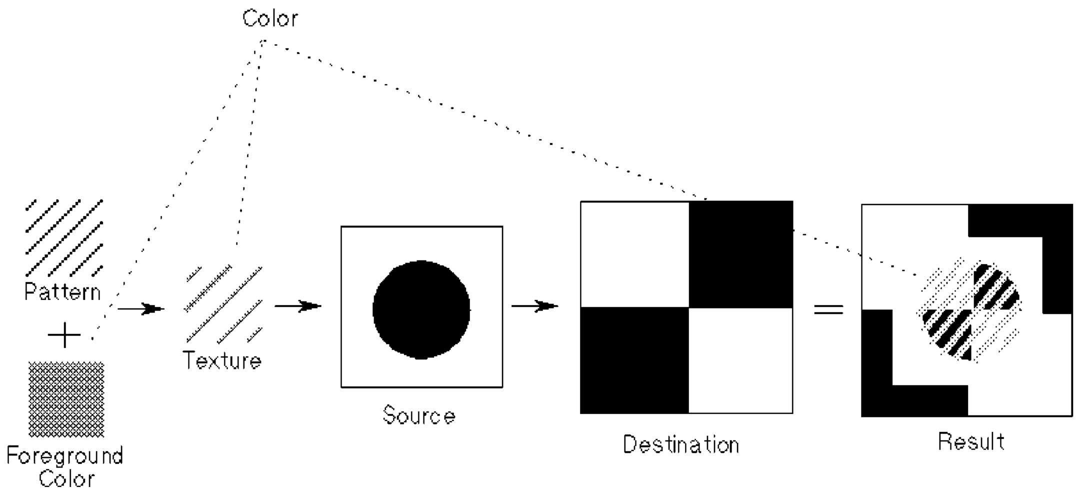
Source Transparency = 1 (Opaque)
Pattern Transparency = 1 (Opaque)



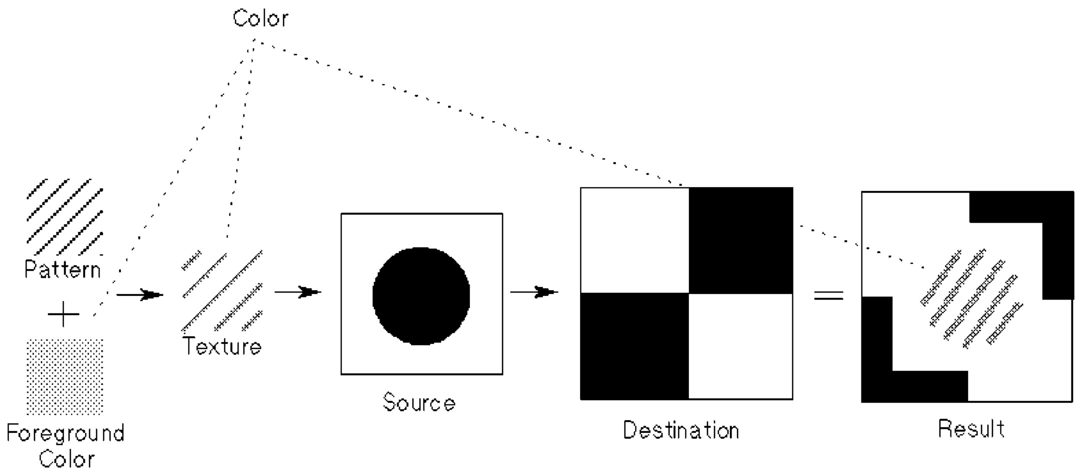
Foreground Color Applied
Source Transparency = 0, Pattern Transparency = 0



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



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



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

16.5 Pixel Placement

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

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

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

Grid Intersection Grid Centered Actual Grid Centered Implementation

Pixel Placement *Esc * l # r/R*

Determines how pixels are rendered in images.

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

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

16.6 Patterns

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

Patterns for Text and Raster

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

1. Specify the Pattern ID (*Esc*c#G*).

2. Download the pattern (*Esc*c#W*). This step is for user-defined patterns only. The downloaded pattern adopts the most recently-specified pattern ID.

3. Apply the pattern to subsequent text and raster. Send the Current Pattern command (*Esc*v#T*).

Patterns for Rules (Rectangular Areas)

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

1. Specify the Pattern ID (*Esc*c#G*). For HP-defined patterns, select an ID that matches an HP-defined pattern.

2. Download the pattern (*Esc*c#W*). This step is for user-defined patterns only. The downloaded pattern adopts the most recently-specified pattern ID.

3. Define the rule. Position CAP and specify rule size (*Esc*c#A*, *Esc*c#H*) or (*Esc*c#B*, *Esc*c#V*).

4. Fill the rule with the pattern. Send the Fill Rectangular Area command (*Esc*c#P*).

HP-GL/2 Patterns

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

Current Pattern *Esc * v # t/T*

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

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

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

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

Pattern ID *Esc * c # g/G*

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

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

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

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

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

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

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

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

HP-defined Hatch Patterns

16.7 User-Defined Patterns

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

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

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

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

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

1. Define a binary raster image as the pattern.
2. Assign a pattern ID (*Esc*c#G*)
3. Download the pattern to the printer (*Esc*c#W[data]*)
4. To fill all subsequent text and raster with the pattern, send Current Pattern (*Esc*v#T*).

or

5. To fill a rule, position CAP, define the size of the rule, and send position Fill Rectangular Area (*Esc*c#P*).

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

Downloads user-defined pattern data.

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

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

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

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

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

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

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

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

	15 (MSB)8	**7 0 (LSB)**	
	Format	Reserved (0)	
	Pixel Encoding	Reserved (0)	
	Height in pixels		
	Width in pixels		
	X resolution*		
	Y resolution*		
	Pattern image . . .		

*Format 20 only

Format Byte

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

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

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

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

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

Pixel Encoding Byte

7 5	4	3 0
0 0 0	Unused	Bits/Index

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

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

Height in Pixels

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

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

Width in Pixels

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

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

X Resolution

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

Y Resolution

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

Pattern Image

The pattern image is the raster data describing the pattern.

Pattern Control   *Esc * c # q/Q*

Manipulates user-defined patterns.

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

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

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

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

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

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

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

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

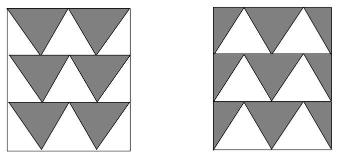
When all the tiles use the same pattern reference point, the pattern in adjoining or overlapping fragments is aligned. This command sets the reference point to CAP, allowing the pattern to be adjusted for different fill areas. The reference point may be shifted for as many fill areas as there are on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

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

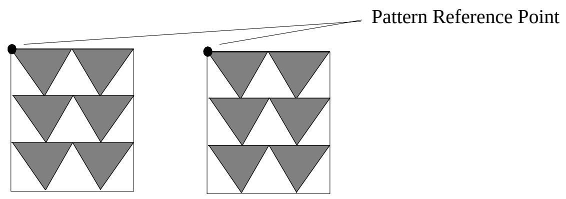
NOTE: Patterns, including user-defined patterns, are applied to images only when a fill is performed by the Fill Rectangular Area (*Esc*c#P*) or Current Pattern (*Esc*v#T*) commands, which can occur any number of times per page.

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

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



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



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

16.8 Rectangular Area Fills (Rules)

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

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

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

Except for the absence of undefined pixels in a rule, pattern transparency acts the same for rules as for other sources. Pattern pixels, defined and undefined, interact with the entire rectangular area.

Horizontal Rectangle Size (PCL Units) ***Esc * c # a/A***

Specifies horizontal rectangle size.

Value(#)	=	Horizontal rectangle size in PCL Units (formerly dots)
Default	=	0
Range	=	0 to 2^{32} -1(clipped to logical page)

Power-up and reset default this value to 0.

Vertical Rectangle Size (PCL Units) ***Esc * c # b/B***

Specifies vertical rectangle size.

Value(#)	=	Vertical rectangle size in PCL Units (formerly dots)
Default	=	0
Range	=	0 to 2^{32} -1(clipped to logical page)

Power-up and reset default this value to 0.

Horizontal Rectangle Size (Decipoints) *Esc * c # h/H*

Specifies horizontal rectangle size.

Value(#)	=	Horizontal rectangle size in decipoints (valid to 4 decimal places)
Default	=	0
Range	=	0 to 2^{32} -1 (fractional values allowed; valid to 4 decimal places)

Power-up and reset default this value to 0.

Vertical Rectangle Size (Decipoints) *Esc * c # v/V*

Specifies vertical rectangle size.

Value(#)	=	Vertical rectangle size in decipoints (valid to 4 decimal places)
Default	=	0
Range	=	0 to 2^{32} -1 (fractional values allowed; valid to 4 decimal places)

Power-up and reset default this value to 0.

Fill Rectangular Area *Esc * c # p/P*

Fills a rectangular area with the specified shade or pattern.

Value(#)	=	0	Solid Black or Foreground Color
	=	1	Solid White
	=	2	HP-defined Shading pattern
	=	3	HP-defined Hatched pattern
	=	4	User-defined pattern
	=	5	Current Pattern
Default	=	0	
Range	=	0 to 5	

When the value is 2, 3, or 4, the Pattern ID (*Esc*c#G*) is an index to the selected fill. A value of 5 uses the Current Pattern (*Esc*v#T*) for the fill.

When the value is 0, 2, 3, 4, or 5, the pattern transparency mode determines the effect of the pattern's undefined pixels on the destination. For a value of 1, pattern transparency is opaque.

NOTE: Pattern transparency mode is treated as if it were opaque when printing white rules.

Example: Filling Rules or Text and Raster

This example shows how to download a pattern and and fill pattern for text and raster data, then load and access another pattern for a rectangular area fill.

FILLING TEXT AND RASTER

1. Download the pattern to be used with ID #1:

*Esc*c1G*	Select a pattern ID.
*Esc*c#W[data]*	Download the pattern to be associated with the ID.

2. Activate the current pattern so it can be printed:

*Esc*v4T*	Use the pattern to fill all subsequent text and raster.

FILLING A DEFINED RULE

1. Download a pattern with ID #2:

*Esc*c2G*	Select a pattern ID.
*Esc*c#W[data]*	Download the pattern to be associated with the new ID.

2. Define a 5 x 7 dot rule.

*Esc*c5A*	Make the rule 5 dots wide.
*Esc*c7B*	Make the rule 7 dots high.

3. Fill the rule with pattern #2.

*Esc*c4P*	Use the pattern to fill the defined rule.

Note that pattern #2 is used to fill the rule; but text or raster fills will still be rendered using pattern #1.

16.9 Arbitrary Masking

Applications that benefit the most from arbitrary clipping or masking in the PCL language are those that perform gradient shading on polygons or clip images to polygons.

To fill a primitive such as a polygon with a gradient, a graphics application can theoretically use one of the following three methods:

Method	**Procedure**
Arbitrary Clip Path	1. Create a clipping window in the shape of the polygon. 2. Draw the gradient with simple rectangles (or circles in the case of a radial gradient) that fully encompass the polygon.
ROPs	1. Set ROP 90 (Destination XOR Texture) and draw the gradient with simple rectangles that fully encompass the polygon. The colors are inverted because of the XOR. 2. Set ROP 240 (Destination = Pattern) and place the solid black polygon on top of the gradient. 3. Set ROP 90 and draw the gradient again (see step 1). Gradient colors within the polygon are set to their final values and gradient colors outside the polygon are removed.
Clipping Windows	1. Create rectangular clipping windows that are the size of scanlines within the polygon to be filled. (One pixel high, and as wide as the polygon at that scanline). Notes: For radial gradients, the clipping windows may be only one pixel wide and one pixel high; This method is resolution dependent. 2. Draw a rectangle, filled with the appropriate color, that encompasses the clipping window.

The most efficient method of the three is Arbitrary Clip Path.

Clip Mask *Esc * l # p/P*

Allows the driver to create a mask that will be used on subsequent drawing primitives.

Value(#)	=	0	Turn off arbitrary masking and delete active mask from memory.
	=	1	Start mask definition.
	=	2	End definition and mask objects with the new mask. (inclusive clipping).
	=	3	End definition and mask objects with the compliment of the new mask. (exclusive clipping).
Default	=	0	
Range	=	0 to 3	

In this context, a “mask” is a bitmap created for the purpose of logical ANDing operations; and “masking” is the application of the bitmap to objects that have been rasterized into bitmap form. The mask is defined by the same objects and commands that a driver would use to draw on the page; however, none of the objects that define the mask are drawn on the page itself. Masking is performed at the printer’s bit-level resolution.

The procedure is as follows:

1. The driver sends this command with a value of 1 to start the mask definition.
2. The driver defines the mask by drawing objects that are rendered into the mask instead of on the page. All PCL and HP-GL/2 primitives are supported except color or multi-plane raster, patterns and color fills. The intent is to be able to use HP-GL/2 polygons, PCL text, and simple (*Esc*r1U*) single-plane monochrome (black & white) raster (bitmaps) to create a mask.
3. When the mask definition is complete, the driver initiates the masking operation by sending the command with a value of 2 or 3.
4. All subsequent objects on the page are masked until masking is turned off by a value of 0 or the printer receives a formfeed or reset. A mask may only apply to the current page.

For example, to clip a raster image to a polygon:

1. Start the definition. (Send the command with a value of 1).
2. Define the mask. (i.e., create and fill a polygon in HP-GL/2).
3. End the definition and start masking. (Send the command with a value of 2).
4. Send the image (which appears only where the polygon was drawn).
5. Terminate masking. (Send the command with a value of 0).

An existing mask is ANDed with a new mask (or its complement from operation 3). That is, primitives defining a mask will not be altered by the active mask. This feature is required to be compatible with PostScript and GDI.

If the current state of mask is off, values of 2 or 3 cause the command to be ignored. A value of 1 is ignored when already in the definition state.

Resulting Mask　　　　Resulting Mask

Example 1　　　　**Example 2**

In **Example 1**, a rectangular mask is already active when a new circular mask is defined. When the definition is completed with a value of 2 in the sequence, a new mask is created by ANDing together the first two masks. In **Example 2**, the same mask is active when the second mask is defined. This time, a value of 3 is sent causing the new mask to be complemented then ANDed to the first mask.

NOTE: For illustration purposes, black represents the areas where objects can mark the page.

Modifications to Current PCL Commands

When the printer is reset (via *EscE*), the arbitrary mask is disabled and deleted. While macros may use this feature, it is not part of the Modified Print Environment. State changes such as foreground color, ROP or patterns which are ignored for mask definition will be retained as appropriate for objects rendered subsequent to mask definition. Non-monochrome (single plane) raster sent during mask definition will be discarded.

Chapter 17: Vector Graphics

Contents of this Chapter

This chapter discusses the following commands:

17.1 Introduction

PCL5 includes most of the HP-GL/2 command set (see *The HP-GL/2 Implemtor's Guide* for the entire HP-GL/2 command set). HP-GL/2 gives PCL a vector graphics capability that is often more efficient than raster graphics. This chapter describes commands that enter and exit the HP-GL/2 context and manage the scaling of HP-GL/2 graphics on the PCL logical page.

Whenever 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.

PCL ***picture presentation*** commands can specify the picture frame size and position. Other picture presentation commands can scale the HP-GL/2 graphic with respect to the picture frame.

However, it may not be necessary to define the picture frame or scale the HP-GL/2 graphic. The picture frame defaults to the size of the factory default PCL text area; and if SC is invoked, any HP-GL/2 graphic is automatically scaled to fit within the picture frame, whatever its size.

17.2 Entering HP-GL/2

Esc%#B enters HP-GL/2 mode and specifies whether CAP and the PCL coordinate system are inherited by HP-GL/2. The current palette is always inherited.

Enter HP-GL/2 Mode *Esc % # B*

Causes a device to interpret data as HP-GL/2 commands instead of PCL commands.

Value(#)	=	-1	Stand-alone plotter mode (single context)
	=	0	Use previous or default HP-GL/2 pen position
	=	1	Use current PCL CAP
	=	2	Use PCL dot coordinate system and previous or default HP-GL/2 pen position
	=	3	Use PCL dot coordinate system and the current PCL CAP
Default	=	0	
Range	=	-1 to 3 (unsupported negative values default to -1; command is ignored for other unsupported values)	

A value field of -1 is the stand-alone plotter mode: HP-GL/2 and PCL data cannot be merged on the same page, and picture presentation commands are ignored. All other values initiate a dual context mode allowing PCL and HP-GL/2 objects to be merged on the same page. A value of 1 or 3 sets the HP-GL/2 pen position and the label carriage return point to the current PCL CAP. A value of 2 or 3 transfers the current PCL dot coordinate system, including the PCL origin and axes, into the HP-GL/2 environment; the coordinate system thus established is independent of the positions of P1 and P2.

HP-GL/2 mode remains in effect until an *Esc%#A*, *EscE*, *Esc%-12345X*, or power-on.

HP-GL/2 ignores this command.

The capital "B" terminator must always be used.

Dual Context

When the value field is non-negative, HP-GL/2 and PCL contexts can both be used on the same page

- HP-GL/2 and PCL objects can be combined on the same page.
- PCL picture presentation commands can specify the size and position of the picture frame on the logical page, as well as scaling the HP-GL/2 graphic.
- The palette and color configuration are transferred between contexts.
- The current logical operation and pixel placement are transferred between contexts.
- CAP can be transferred between contexts.
- PCL orientation determines HP-GL/2 orientation.

DEVICE NOTE: LaserJets do not support values of -1, 2, and 3.

DEVICE NOTE: LaserJets in the past have sometimes set the HP-GL/2 hardclip limts one dot larger than the PCL printable area

Effects of PCL States

The following PCL state changes can affect the HP-GL/2 state:

Reset (*EscE*) or control-panel reset:

- Executes BP, or IN if BP is not supported.
- Creates a 2-pen PCL default palette.
- Defaults picture frame size to the factory default PCL text area.

- Defaults the picture frame anchor point to the upper left corner of the text area.
- Defaults HP-GL/2 plot size to the picture frame size.
- Defaults HP-GL/2 orientation to the current PCL logical page orientation.
- Does not change the SP pen number, but may change the color of that pen by defaulting the palette.

Orientation *(Esc&l#O*), **Paper Size** (*Esc&l#A*), **Page Length** (*Esc&l#P*)

- Defaults picture frame size to the factory default PCL text area.
- Defaults the picture frame anchor point to the upper left corner (relative to the current orientation) of the PCL text area.
- Defaults P1 and P2 to the lower left and upper right corners (relative to the current orientation) of the picture frame, respectively ("IP;").
- Defaults the soft-clip window to the picture frame ("IW;").
- Clears the polygon buffer ("PM0; PM2;").
- Moves the HP-GL/2 pen position to the lower-left corner of the picture frame (P1).
- Recomputes the picture frame scale factor (ratio of the new picture frame size to previous plot size).

Picture Frame Size (*Esc*c#X*, *Esc*c#Y*), **Anchor Point** (*Esc*c#T*)

- Defaults P1 and P2 to the lower left and upper right corners of the picture frame, respectively ("IP;").
- Defaults soft-clip window to the picture frame ("IW;").
- Clears the polygon buffer ("PM0; PM2;").
- Moves the HP-GL/2 pen position to the lower-left corner of the picture frame (P1).
- Recomputes the picture frame scale factor (ratio of the new picture frame size to previous plot size).

HP-GL/2 Plot Size (*Esc*c#K*, *Esc*c#L*)

- Changes the picture frame scale factor

Palette Redefinition (*Esc*r#U*, *Esc*v#W*, *Esc*v#A*, *Esc*v#B*, *Esc*v#C*, *Esc*v#I*)

- Changes the colors selected by SP and used in patterns defined by RF and/or used by FT.

Logical Operation (*Esc*l#O*)

- ROP values are transferred between contexts.

Pixel Placement (*Esc*l#R*)

- Pixel Placement is transferred between contexts.

Render Algorithm (*Esc*j#T*)

- Changes the halftoning method employed to render non-primary colors.

DEVICE NOTE: LaserJets prior to LJ4, DeskJets below DJ1200C, PaintJet XL300, and the Color LaserJet do not allow *Esc%#A*, *Esc%#B*, or HP-GL/2 commands in PCL macro definitions.

Transferring the Coordinate System

Esc%2B and *Esc%3B* have the following additional effects:

- HP-GL/2 adopts the current PCL dot coordinate system, origin, axes, and print direction. All HP-GL/2 commands using current units will use PCL dot units.
- An implicit RO0 is issued. Any RO will move the HP-GL/2 origin to where the PCL origin would be if the PCL coordinate system were rotated by the Print Direction command (*Esc&a#P*).

- Picture frame scaling using the plot size commands (*Esc*c#K*, *Esc*c#L*) is disabled; images are clipped to the picture frame.
- An SC with parameters re-adopts the HP-GL/2 coordinate system and applies picture frame scaling. A subsequent *IN* or "*SC;*" will again restore the PCL dot coordinate system.

Shared State Variables

In dual context mode, the following state variables are shared:

- Current palette (whether created by *EscE*, *Esc*r#U*, *Esc*v#W*, *Esc*d#W*, *IN*, *BP*, or reprogrammed by *Esc*v#A*, *Esc*v#B*, *Esc*v#C*, *Esc*v#I*, *CR*, *NP*, *PC*)
- Logical operation (*Esc*l#O*, *MC*)
- Pixel placement (*Esc*l#R*, *PP*)

Single Context

When the value field is -1, the device behaves like a "stand-alone plotter".

- The current page is closed and printed. HP-GL/2 is initialized (BP/IN) and HP-GL/2 output begins on a new page. Exiting HP-GL/2 via *Esc%#A* closes and prints the current page and performs *EscE* before entering PCL. HP-GL/2 and PCL cannot be combined on the same page.
- The only PCL commands recognized are *EscE* and *Esc%#A*, or the Universal Exit command.
- PCL picture presentation directives are ignored.
- The default HP-GL/2 orientation is reverse landscape.
- HP-GL/2 hardclip limits are equal to PCL printable area.
- *EscE* retains its usual functionality.
- The command is ignored in HP-GL/2 mode.

17.3 Exiting HP-GL/2

Esc%#A exits HP-GL/2 and specifies whether the new CAP is derived from the HP-GL/2 pen position or the previous CAP.

Enter PCL Mode *Esc % # A*

Causes a device to interpret data as PCL commands instead of HP-GL/2 commands.

Value(#)	=	0	Use previous PCL CAP
	=	1	Use current HP-GL/2 pen position for CAP
Default		=	0
Range		=	0,1

A missing or even value maps to 0; an odd value maps to 1.

The capital "A" terminator must always be used with this command.

If the current HP-GL/2 pen position is outside the PCL logical page and the value field is 1, CAP moves to the nearest point on the logical page boundary.

This command is ignored in PCL mode.

In dual context mode, the following state variables are shared:

- Current palette (whether created by *EscE*, *Esc*r#U*, *Esc*v#W*, *Esc*d#W, IN*, *BP*, or reprogrammed by *Esc*v#A*, *Esc*v#B*, *Esc*v#C*, *Esc*v#I*, *CR*, *NP*, *PC*)
- Logical operation (*Esc*l#O*, *MC*)
- Pixel placement (*Esc*l#R*, *PP*)

NOTE: This command is also used to transition from non-HP emulation modes to PCL mode. When making the transition from a non-HP emulation mode (e.g., Epson), an absent or 0 value field resets the PCL state environment.

DEVICE NOTE: LaserJets prior to LJ4, DeskJets below DJ1200C, PaintJet XL300, and the Color LaserJet do not allow *Esc%#A*, *Esc%#B*, or HP-GL/2 commands in PCL macro definitions.

17.4 Resetting to Defaults

Reset *Esc E*

Both HP-GL/2 and PCL recognize *EscE*. An *EscE* executed within HP-GL/2 has the following effects:

- Peforms a PCL reset (with the same effects on PCL variables as if it were executed within PCL).
- Executes an IN / BP.
- Defaults all programmable features.
- Configures a two-pen, non-modifiable black and white default PCL palette.
- Prints any partial pages of data that have been received.
- Returns to PCL parsing mode.

After *EscE*, the device should remain on-line, no data should be lost, and there should be no effect on I/O or host-to-peripheral communication.

EscE is a valid HP-GL/2 terminator and has all the functionality of IN, as well as defaulting:

- Color palette
- Logical page orientation
- Picture frame
- Picture frame anchor point
- HP-GL/2 plot size

Within an HP-GL/2 label when Transparent Data is off (TD = 0), *EscE* resets a device and switches to PCL mode. When TD = 1, *EscE* is interpreted as data and does not cause a reset.

EscE, page length, page size, and orientation commands do not eject pages containing no print data (characters, raster graphics, or HP-GL/2 graphics).

At power-on or *EscE*, all programmable features are defaulted:

Picture frame:	Defaults to the PCL text area.
Picture frame anchor point:	Defaults to the left edge of the logical page at default top margin.
Plot size:	Defaults to the picture frame size.
Scale factor:	Defaults to 1. No scaling is performed.
P1 and P2:	Default to the lower-left and upper-right picture frame corners, respectively. The lower-left corner is at HP-GL/2 position (0,0) with user scaling off.
Soft-clip window:	Defaults to the picture frame.

HP-GL/2 Factory Environment Defaults

Character Group

- Character set = device dependent
- Font spacing = device dependent
- Pitch = device dependent
- Height = device dependent
- Posture = device dependent
- Stroke weight = device dependent
- Typeface = device dependent
- Character direction = horizontal
- Character direction mode = absolute
- Character size mode = absolute
- Character width = device dependent
- Character height = device dependent
- Character slant = 0
- Extra horizontal space = 0
- Extra vertical space = 0
- Character fill mode = solidly fill characters with current pen without edging
- Label origin = 1
- Label terminator = non-printing
- Transparent data mode = off

Vector Group

- Plotting mode = absolute
- Pen state = up

Polygon Group

- Polygon buffer = cleared
- Polygon mode = off

Line and Fill Attribute Group

- Line type = solid
- Line type repeat length = 4% of the diagonal distance from P1 to P2
- Line cap = butt
- Line join = mitered
- Miter limit = 5
- Pen color = default HP-GL/2 palette
- Pen width selection mode = metric, pen widths = 0.35 mm
- Selected pen = 1 (black)
- Symbol mode = off
- Fill type = solid (bidirectional), odd-even fill algorithm
- User-defined line type = eight standard line types as defined by LT
- Area fill anchor corner = (0,0) plotter units
- User-defined fill types = solid fill

Configuration and Status Group

- Scale mode = off (units are plotter units)
- Window = PCL default picture frame (PCL default logical page less 1/2" at the top and the bottom)
- Coordinate system orientation = orientation of PCL default logical page coordinate system
- P1, P2 = lower left, upper right corners, respectively, of picture frame

HP-GL/2 Modified Print Environment

The modified print environment consists of the current settings for the following:

- Current font
- Primary font
- Secondary font
- Primary font attributes
- Secondary font attributes
- Label terminator
- Label direction
- Label direction mode (abs/rel)
- Extra horizontal space
- Extra vertical space
- Character size mode (abs/rel)
- Character width
- Character height
- Character slant
- Label origin
- Character fill mode
- Area fill anchor corner
- Fill type
- Line type
- Line type mode (abs/rel)
- Line type repeat length
- Plotting mode (abs/rel)
- Selected pen
- Pen state (up/down)
- Symbol mode (on/off)
- Soft clip window
- Current scaling information
- P1x
- P1y
- P2x
- P2y
- Pen width selection mode (rel/metric)
- Pen width units
- Pen width
- Transparent data mode
- Line cap
- Line join
- Miter limit
- Coordinate system orientation
- Current active position

The following items in the HP-GL/2 context are not part of the modified print environment:

- User-defined raster fill patterns
- Polygon buffer
- Parser state

The macro overlay environment does not include any of the HP-GL/2 settings.

17.5 Picture Presentation Commands

The default picture frame (PCL text area) normally defines the plot's destination; but picture presentation directives 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.

The General Process

The following command sequence is suggested. Some steps are unnecessary or are automatically:

1. Position the picture frame on the logical page (anchor point).
2. Size the picture frame, if necessary.
3. Position the plot in the picture frame, if necessary.
4. Scale the plot to the picture frame, if necessary.
5. Enter HP-GL/2.
6. Send the HP-GL/2 commands describing the graphic.
7. Exit HP-GL/2.

Positioning the Picture Frame

The picture frame position on the logical page is determined by the Picture Frame Anchor Point command (*Esc*c#T*), which places the upper-left corner of the picture frame at CAP.

Picture Frame Anchor Point *Esc * c # t/T*

Sets the location of the picture frame anchor point to the PCL CAP

Value(#)	=	0 - Picture frame anchor point is set to CAP
Default	=	Left edge of the logical page at the default top margin
Range	=	0 (all other values are ignored)

The picture frame anchor point defines the the upper-left corner of the picture frame. *Upper-left* refers to the corner whose PCL X and Y coordinates are minimized when print direction is 0.

Reset (*EscE*), Page Length (*Esc&l#P*), Page Size (*Esc&l#A*), Orientation (*Esc&l#O*), or the absence of a command defaults the anchor point to the left edge of the logical page at the default top margin. Print direction (*Esc&a#P*) has no effect on the physical location of the anchor point or the picture frame.

This command has the following additional effects:

- Defaults P1 and P2 to the lower-left and upper-right corners of the picture frame, as viewed from the current orientation.
- Resets the soft-clip window to the PCL picture frame boundaries.
- Clears the polygon buffer.
- Sets CAP to the lower-left corner (P1) of the picture frame (viewed in the current orientation).
- Sets the Label Carriage Return point to the lower-left (P1) corner of the picture frame (viewed in the current orientation).

Sizing the Picture Frame

Default picture frame width is the width of the logical page; default length is the default text length (which extends the length of the physical page except for 0.5" at the top and bottom edges). The picture frame size commands (*Esc*c#X*, *Esc*c#Y*) described below can modify these default dimensions.

Horizontal Picture Frame Size (decipoints) *Esc * c # x/X*

Specifies the horizontal size of the window on the logical page allocated for an HP-GL/2 plot.

Value(#)	=	Horizontal size in decipoints
Default	=	Width of the current logical page
Range	=	0 to 2^{32} -1 (fractional values allowed; valid to 4 decimal places)

The picture frame’s horizontal aspect is parallel to the current PCL X-axis. Once specified, subsequent orientation and print direction changes do not affect picture frame position and dimensions.

Horizontal picture frame size defaults to the logical page width if no parameter or a parameter of 0 is sent, or following a reset, page length, page size, orientation or UEL command.

This command has the following additional effects:

- Defaults P1 and P2 to the lower-left and upper-right corners of the picture frame.
- Resets the soft-clip window to the PCL picture frame boundaries.
- Clears the polygon buffer.
- Updates CAP and the Label carriage-return point to the lower-left corner of the picture frame (P1), as viewed from the current orientation.

Vertical Picture Frame Size (decipoints) *Esc * c # y/Y*

Specifies the vertical size of the window in the logical page allocated for an HP-GL/2 plot.

Value(#)	=	Vertical size in decipoints
Default	=	Distance between the default top and bottom margins (default text length)
Range	=	0 to 2^{32} -1 (fractional values allowed; valid to 4 decimal places)

The picture frame’s vertical aspect is parallel to the current PCL Y-axis. Once specified, subsequent orientation and print direction changes do not affect picture frame position and dimensions.

Vertical picture frame size defaults to the default PCL text length if no parameter or a parameter of 0 is sent, or following a reset, page length, page size, orientation command, or UEL command.

This command has the following additional effects:

- Defaults P1 and P2 to the lower-left and upper-right corners of the picture frame.
- Resets the soft-clip window to the PCL picture frame boundaries.
- Clears the polygon buffer.
- Updates CAP and the Label carriage-return point to the lower-left corner of the picture frame (P1), as viewed from the current orientation.

Positioning the Plot

The orientation of the HP-GL/2 plot within the picture frame depends upon:

- Logical page orientation
- RO, the HP-GL/2 rotate command
- The value field of *Esc%#B*
- Print direction if HP-GL/2 is entered with *Esc%2B.*

PCL print direction does not affect the physical position and dimensions of the picture frame, the picture frame anchor point, or HP-GL/2 orientation (except when using *Esc%2B* or *Esc%3B*).

Effect of *Esc%2B* or *Esc%3B*

Esc%2B and *Esc%3B* transfer the PCL coordinate system, origin, and unit of measure. This allows HP-GL/2 to operate in the PCL coordinate system. *Esc%2B* and *Esc%3B* also ignore the picture-frame scaling of the Plot Size commands *Esc*c#K* and *Esc*c#L*.

Logical Page Tracking

Default HP-GL/2 orientation tracks PCL logical page orientation. In the default HP-GL/2 orientation (RO0), the origin is at the lower-left corner of the picture frame. Both the PCL and HP-GL/2 X-axes increase in the same direction; but the Y-axes increase in opposite directions.

How HP-GL/2 Orientation Tracks PCL Logical Page Orientation

Modifying the Orientation with RO

RO can modify the default HP-GL/2 orientation. RO0 moves the default HP-GL/2 coordinate system origin to the lower-left corner of the PCL picture frame. (The lower-left corner is that in which the PCL X-coordinate is minimized and the Y-coordinate maximized when print direction is defaulted.)

NOTE: Entering HP-GL/2 with *Esc%2B* or *Esc%3B* implicitly executes RO0.

How RO Modifies the Default HP-GL/2 Orientation

Fitting the Plot to the Picture Frame

When an HP-GL/2 plot is to be placed into the picture frame on the PCL logical page, there are three possible outcomes:

- A *page-size independent* plot (specified entirely in user-units by SC) is automatically scaled to fit into the picture frame — user-scaling.
- A *page-size dependent* plot (specified in absolute units) is scaled to fit into the picture frame by specifying horizontal and vertical plot size (*Esc*c#K*, *Esc*c#L*) — picture frame scaling.
- Unless the plot size commands (*Esc*c#K*, *Esc*c#L*) are used, a *page-size dependent* plot using absolute units is not scaled. The plot is clipped if it extends beyond the effective window.

Page-Size Independent Plots

The parameters of many HP-GL/2 commands are interpreted according to the *current units* in effect:

- If user scaling (SC) is in effect, current units are in ***user units***. User units are defined by the two scaling points P1 and P2, which specify opposite corners of user-coordinate space.
- If user scaling (SC) is not in effect, current units are in absolute ***plotter units***. One plotter unit equals 0.025 mm or 0.00098 inches.

A plot specified entirely in user units (SC) is automatically sized to fit the picture frame. Such a plot is page-size independent if it satisfies the following requirements:

- No positioning or drawing command parameter is in plotter units.
- Scaled mode (SC, types 0 or 1, not type 2) is used exclusively.
- Scalable fonts (SB) are used exclusively.
- Either the default P1 and P2 locations are used, or their positions are specified by IR.
- Either the default window is used, or the window is specified in user units.
- Only the SR mode is used for labels; SI is not used.
- Only the DR mode is used for label direction.
- The pen width selection mode (WU) is relative, not metric.
- The pattern length for line types (LT) is relative, not metric.
- Stick fonts should use relative or no pen width.

Scaling with Plot Size Commands

Page-size dependent plots (those originally specified in absolute units) are rendered in their original native size and clipped to the picture frame if necessary. However, the Plot Size commands (*Esc*c#K*, *Esc*c#L*) can scale the plot to the picture frame — picture frame scaling. The printer uses the sizes specified by these commands to compute a horizontal and a vertical *scale factor* (the ratio of a picture frame dimension to the corresponding plot dimension). The plot is then scaled to fit the picture frame in each dimension. The plot's aspect ratio is maintained only if the specified plot sizes have the same aspect ratio as the picture frame.

For example, if the original HP-GL/2 drawing is 8.5" wide by 7" high, sending *Esc*c8.5k7L* scales the plot as necessary to fit into the picture frame. If the horizontal plot size is specified as 12" when the horizontal picture frame size is 6", the horizontal scale factor is 1:2, and the plot's horizontal dimension is reduced to 1/2 its original size. If the ratio is 2:1, the plot's horizontal dimension is enlarged to twice its original size.

If user scaling (SC) is not in effect and the picture frame and plot are not the same size, current units are in *picture frame units*, which are plotter units multiplied by the picture frame scale factor.

Some commands operate only on absolute units. For example, SI parameters are always in centimeters. The picture frame scale factor is applied to the parameters of these commands.

Scaling also affect labels printed with a bitmap font (SB1). As with SI and SR, the bitmap font most closely matching the desired character size will be used.

The scale factor modifies sizes specified as parameters to SI, SD, and AD. For example, if a 14-point font is requested and the scale factor is 1/2, a 7-point font is selected if available.

The Effective Window

When drawing an HP-GL/2 plot, the intersection of four boundaries delimits the area on the physical page where a vector graphics image can be printed:

- Hard clip limits: the HP-GL/2 physical printing limits.
- Soft clip limits: temporary HP-GL/2 boundaries defined by IW or IR.
- Logical page: the current PCL logical page boundaries.
- Picture frame: the current PCL picture frame.

Horizontal Plot Size *Esc * c # k/K*

Specifies the horizontal size of the imported HP-GL/2 plot.

Value(#)	=	horizontal size in inches
Default	=	Picture frame width
Range	=	0 to 2^{32} -1 (fractional values allowed; valid to 4 decimal places)

The horizontal plot size determines the horizontal scale factor used to fit the drawing into the PCL picture frame. For example, if horizontal plot size is 12" and picture frame width is 4", the horizontal scale factor is 1:3. The horizontal component of the image would be reduced to one-third its original size to fit into the picture frame.

Plot width defaults to picture frame width, with no scaling (i.e., the plot is rendered in its original size and clipped to the picture frame if necessary), if the value field is zero, or following a reset, page length, page size, orientation, or UEL command.

Vertical Plot Size *Esc * c # l/L*

Specifies the vertical size of the imported HP-GL/2 plot

Value(#)	=	Vertical size in inches
Default	=	Picture frame length
Range	=	0 to 2^{32} -1 (fractional values allowed; valid to 4 decimal places)

The vertical plot size value determines the vertical scale factor used to fit the drawing into the PCL picture frame. For example, if vertical plot size is 7" and picture frame height is 14", the vertical scale factor is 2:1. The vertical component of the image would be enlarged to twice its original size to fit into the picture frame.

Plot height defaults to picture frame height, with no scaling (i.e., the plot is rendered in its original size and clipped to the picture frame if necessary), if the value field is zero, or following a reset, page length, page size, orientation, or UEL command.

17.6 HP-GL/2 Modifications in the PCL Context

The functionality of the following HP-GL/2 commands is modified in the PCL context.

CR Color Range

Can be used to change the black and white references for palettes created by *IN*, *BP*, or *Esc*v#W*. The new references are remembered when the context is changed back to PCL. This command is ignored if the current palette was created by *Esc*r#U* or *EscE*.

FT Fill Type

The following additional *fill_types* are imported from PCL when the HP-GL/2 context is entered with a zero or positive parameter.

21:	PCL predefined pattern
22:	PCL user-defined pattern

Fill_type 21 selects PCL predefined patterned fill. *Option 1* specifies the pattern type. A fill type is specified using a value between 1 and 6. *Option 2* is ignored if present.

Fill_type 22 selects a PCL user-defined fill previously specified by *Esc*c#W*. *Option 1* specifies the PCL ID of the user-defined fill pattern. *Option 2* is ignored if present. If option 1 is invalid for any reason (e.g., deleted pattern), a solid fill in the current pen color is selected.

NP Number of Pens

Changes the number of palette entries after *IN*, *BP*, or *Esc*v#W*. This command is ignored if the current palette was created by *Esc*r#U* or *EscE*.

The size and contents of the current palette depend upon where it was created (*Esc*v#W*, *Esc*d#W*, *IN*, or *BP*) and whether it has been reprogrammed (*Esc*v#A*, *Esc*v#B*, *Esc*v#C*, *Esc*v#I*, or *PC*). *NP* can increase or decrease the size of the current palette. If *NP* increases the current palette size, additional entries will contain black. If *NP* reduces the current palette size, the excess indices are eliminated. In either case, the bits-per-index value which indicates the size of the palette is adjusted accordingly.

PC Pen Color Assignment

Changes the color of pens in a palette created by *IN*, *BP*, or *Esc*v#W*. This command is ignored if the current palette was created by *Esc*r#U* or *EscE*.

SV Screened Vectors

The following additional screen types are imported from PCL when the HP-GL/2 context is entered with a zero or positive parameter.

21:	PCL predefined pattern
22:	PCL user-defined pattern

Chapter 18: Macros

Contents of this Chapter

This chapter discusses the following PCL commands:

Macro ID	*Esc&f#Y*	18-7
Macro Control	*Esc&f#X*	18-7

18.1 Introduction

A macro is a group of PCL commands and data created by the user that may reside on cartridge or be downloaded and stored in a device's user memory. Once stored, a macro can be executed repeatedly by a single command, using an assigned macro ID.

Macros eliminate the need to repeatedly download the same information, thus saving transmission time. However, the trade-off is the consumption of user memory. An example of a macro might be the printing of a company letterhead. The letterhead is created as a macro and stored in the printer. Whenever the letter is printed, a macro command prints the letterhead.

The number of macros that can be stored in user memory is limited to either 32767, the number of macro IDs that can be assigned, or by the available memory.

DEVICE NOTE: LaserJets prior to LJ4, DeskJets below DJ1200C, PaintJet XL300, and the Color LaserJet do not support HP-GL/2 commands or picture frame directives within macros.

18.2 Creating Macros

Macro creation entails the following steps:

1. **Designate a macro identification number** with the Macro ID command (*Esc&f#Y*). Any macro already assigned this number is deleted.

2. **Start the macro definition** with the Macro Control command (*Esc&f0X*). Subsequent commands and data are to be stored as a macro.

3. **Send the commands in the macro**. The macro commands and data are sent to the printer in the intended order of execution.

4. **Stop the macro definition** with the Macro Control command (*Esc&f1X*), which identifies the end of the macro definition.

18.3 Invoking Macros

Macros are invoked with the Macro Control command (*Esc&f#X*). There are three ways to invoke a macro:

- Call
- Execute
- Overlay

Calling a Macro

A **called** macro restores the original modified print environment (except for CAP) upon completion.

Executing a Macro

An **executed** macro does not restore the original modified print environment upon completion. Changes made by the macro are retained.

Overlaying

A macro **overlay** is the final operation each time a page is printed. A macro overlay restores the original modified print environment upon completion. During execution, a macro overlay uses the overlay environment, which is a combination of user defaults and the modified print environment.

18.4 Managing Macros

Macros are managed like fonts. They are automatically designated **temporary** at definition and are deleted during a printer reset unless designated **permanent**. The Macro Control command (*Esc&f#X*) designates macros as temporary or permanent. Temporary and permanent macros are removed from memory at power off.

Several PCL mechanisms can explicitly delete macros from user memory. Macros may be deleted explicitly by the Macro Control command (*Esc&f#X*), or implicitly by *EscE*, control panel reset, macro definition, and self test.

18.5 The Modified Print Environment

The modified print environment consists of all current feature settings. Settings altered by escape sequences are recorded in the modified print environment. A macro **call** or **overlay** saves the current modified print environment and restores it following completion. The modified print environment consists of the following state variables:

JOB CONTROL
- Left and top registration
- Copy count
- Job separation
- Media destination

PAGE CONTROL
- Size, Length
- Orientation
- Print direction
- Media source

MARGINS
- Left, Right, Top, Bottom
- Perforation skip

FONTS
- Primary selected font
- Secondary selected font
- Primary font attributes
- Secondary font attributes
- Font ID
- Character code

TEXT
- EOL wrap
- Line termination
- HMI, VMI/Line spacing
- Underline mode
- Text Parsing
- Text Path Direction

COLOR
- Foreground color
- Number of raster planes
- Gamma value
- Render algorithm
- Downloaded dither
- Current palette
- Palette select ID
- Palette control ID
- Monochrome print mode
- Viewing illuminant
- Finish mode
- Lightness setting
- Saturation setting
- Color treatment
- Color lookup tables
- Primary chroma values
- White point
- Primary gamma values
- Primary gain values
- Primary encoding values

RASTER
- Raster mode
- Presentation
- Compression method
- Source width and height
- Resolution
- Left graphics margin
- Scale algorithm
- Destination width and height

AREA FILLS
- Current Pattern
- Pattern ID
- Horizontal and vertical rectangle size
- Pattern reference point

MACROS
- Macro ID

PRINT MODEL
- Logical operation
- Source and pattern transparency

The following items are not part of the modified print environment.

Overlay
CAP, and the CAP stack
Raster seed row
Downloaded fonts/macros

NOTE: CAP is not part of the modified print environment: CAP is not saved when a macro is called, nor is it restored upon completion. The Push/Pop CAP command (*Esc&f#S*) can be used to save and recall CAP.

The HP-GL/2 Modified Print Environment

On printers allowing HP-GL/2 commands within macros, the HP-GL/2 modified print environment must be saved when a macro is called or enabled for overlay. The HP-GL/2 modified print environment consists of the following:

Alternate font definition
Anchor corner mode
Carriage return position
Chacter width
Character fill type
Character size
Character slant
Clip tolerance
Clip window
Coordinate system orientation
Curently selected pen
Current font selection
Current pen position
Default hardclip limits
Extra horizontal space
Extra vertical space
Fill type
Input window
Input window mode
Label direction
Label direction mode
Label origin
Label terminator
Last pen down

Last symbol position
Last vector
Last zero-length vector
Line attributes
Line type mode
Logical page boundaries
Orientation
Palette
Pen hardclip limits
Pen width
Pen width mode
Pen width unit mode
Physical size
Picture frame
Plotting mode (abs/rel)
Scaling
Screen
Skip vector flag
Standard font definition
Symbol mode
Thin vector flag
Transparency
Transparent data mode
Zero length vector flag

The following items in the HP-GL/2 context are not part of the modified print environment:

User-defined raster fill patterns
Polygon buffer
Parser state

DEVICE NOTE: LaserJets prior to the LaserJet 4, PaintJet XL300, DeskJets prior to DeskJet 1200C, and the Color LaserJet do not support HP-GL/2 commands within macros. This includes Enter HP-GL/2 (*Esc%#B*) as well as the picture frame directives (*Esc*c#T*, *Esc*c#X*, *Esc*c#Y*, *Esc*c#K*, and *Esc*c#L*).

18.6 The Overlay Print Environment

The overlay print environment is the same as the modified print environment except that CAP and CAP stack are also saved. After the overlay is completed, the saved feature settings are restored.

18.7 Rules Governing Macros

1. A macro may span multiple pages, since commands causing page ejects are allowed in a macro (e.g., Form Feed, Page Length).

2. A macro may call or execute another macro, which in turn may call or execute another macro; two levels of nesting (i.e., three total levels) are allowed. Attempting another level is ignored.

3. Other than call and execute, no macro operations may occur within a macro.

4. Reset (*EscE*) is not allowed in a macro.

5. A macro enabled for overlay is executed on each page until the macro is disabled or deleted, reset occurs, or page length, page size, or orientation is changed. The macro overlay is executed last on each page; i.e., it is the final operation before each page is printed.

6. If page length, size, or orientation is changed, or the primary font is deleted during a call or overlay, the following actions are necessary to restore the original modified print environment:

   - If the original page length or size was different than the current length or size, the current page is closed and all pages are printed. Page length and size are changed to the original value, and CAP is set to (0,0).

   - If the original orientation was different than the current orientation, the page is closed and printed, the original orientation is set, and CAP is set to (0,0).

   - If the primary or secondary font was deleted, the font select commands are used to select the closed font.

DEVICE NOTE: LaserJets prior to LJIII except LJ2000 ignore any font download or font deletion commands within a macro.

7. A multiple-page macro may be enabled for overlay. However, when the macro causes a page eject, it must not cause the overlay to re-invoke the overlay (i.e., once overlay is in progress, it cannot be invoked again until the current overlay is finished).

18.8 Macro Commands

Macro ID *Esc & f # y/Y*

Specifies an ID number for use in subsequent macro commands.

Value(#)	=	Macro ID number
Default	=	0
Range	=	0 to 2^{32} -1

A unique identification (ID) number should be designated prior to the definition of a macro; this number is then associated with the macro, and subsequent macro operations are accomplished using the macro ID number. If a macro is already associated with this ID number, the existing macro is deleted from user memory during the definition of the new macro.

Macro Control *Esc & f # x/X*

Provides mechanisms for definition, invocation, and deletion of macros. This command is ignored if the macro does not exist.

Value(#)	**Function**
0	Start macro definition (last ID specified)
1	Stop macro definition
2	Execute macro (last ID specified)
3	Call macro (last ID specified)
4	Enable macro for overlay (last ID specified)
5	Disable overlay
6	Delete all macros
7	Delete all temporary macros
8	Delete macro (last ID specified)
9	Make macro temporary (last ID specified)
10	Make macro permanent (last ID specified)
11	Create bitmap of macro

0, 1 Start Macro Definition or Stop Macro Definition cause the following action:

- A macro with the same ID as the current Macro ID state variable is deleted (even if it is permanent). A new macro is created, marked as temporary, and assigned the current Macro ID as its identifier.
- All data (including binary data) is parsed, but not executed, and stored as the definition of Macro ID until a Stop Macro Definition or reset (control panel or EscE) is received. The entire macro is discarded if out-of-memory occurs during the macro definition.

2 ***Execute Macro*** causes the following action:

- Processing from the I/O buffer halts, the current parser state is saved, and the parser is reset to the "top level".
- The macro specified by Macro ID is executed using the current user's environment. This is done by parsing and executing the raw macro data as input (as if it had come from the I/O).
- Upon completion of the macro, the parser state is restored and processing resumes with the next character in the I/O buffer.

3 ***Call Macro*** causes the following action:

- Processing from the I/O buffer is halted, current parser state and current user's environment are saved, and the parser is reset to the "top level".
- The macro specified by Macro ID is executed using the current user's environment. This is done by parsing and executing the raw macro data as input (as if it had come from the I/O).
- Upon completion of the macro, the parser state and the user's environment are restored, and processing resumes with the next I/O buffer character.

4 ***Enable Macro for Overlay*** marks the macro specified by *Macro ID* as the auto macro overlay for each user page and performs the following action whenever a user page is closed (before being sent out for printing):

- Processing from the I/O buffer is halted, current parser state and current user's environment are saved, and the parser is reset to the "top level".
- The overlay environment is created using the default values for most of the state variables and some variables from the current user's environment.

NOTE: The overlay environment is newly created on each page, just before the macro is executed.

- The auto macro overlay macro is executed using the overlay environment.
- Upon macro completion, the current page is closed and printed, the parser state and the user's environment are restored, and processing resumes with the next I/O buffer character.

NOTE: If the auto macro overlay is disabled before the current page is ejected, the overlay will not appear on that page.

NOTE: If the specified macro does not exist, the overlay is disabled and no macro is enabled.

NOTE: Any macro previously enabled for auto macro overlay is disabled.

5 ***Disable Overlay*** disables the auto macro overlay, starting with the current page.

NOTE: Commands that change page length, page size, or orientation disable the auto macro overlay after the command has been executed.

6 **Delete All Macros** deletes all temporary and permanent macros. This command disables the auto macro overlay.

7 **Delete All Temporary Macros** deletes all temporary macros. If this command deletes the auto macro overlay, it disables the auto macro overlay.

8 **Delete Macro** deletes the macro specified by the Macro ID state variable. If this command deletes the auto macro overlay, it disables the auto macro overlay.

9 ***Make Macro Temporary*** sets the state of the macro specified by *Macro ID* to temporary.

10 ***Make Macro Permanent*** sets the state of the macro specified by *Macro ID* to permanent.

11 ***Create Bitmap of Macro*** creates the static overlay bitmap, which can then be managed by using the other macro control values**.**

Static overlay is a hardware/software performance enhancement that rasterizes a macro into a full-page bitmap that is then used for subsequent printing. Static overlay can improve the printing performance of large static images like electronic forms. It is best used for large complex images that are used repeatedly, such as billing invoices to the HP5000.

Prior to rasterizing the macro, the current modified print environment is saved and replaced with the overlay environment; after rasterization, the previous modified print environment is restored. The resulting bitmap is stored in memory, designated temporary (by default), and assigned the same ID as the macro from which it was generated.

NOTE: The bitmap version of a macro can be printed only if the macro is enabled and invoked as an automatic overlay. Otherwise, if the macro is called or executed, the native version of the macro (text and PCL commands) will be used.

Since the bitmap version of a macro has the same ID as the native version, operations such as making a macro permanent or temporary, or deleting a macro affects both versions. For example, deleting a macro by ID deletes both the native and bitmap versions of that macro.

A value field of 11 modifies the other macro operations as follows:

1. Unchanged.
2. Unchanged.
3. Execute macro using the native version only (last ID specified).
4. Call macro using the native version only (last ID specified).
5. Enable macro for automatic overlay. If the bitmap version exists (i.e., the macro has been rasterized with *Esc&f11X*), it will be printed when the macro is invoked implicitly through a page eject such as <FF>.
6. Disable auto overlay.
7. Delete native and bitmap versions of all macros.
8. Delete native and bitmap versions of temporary macros.
9. Delete native and bitmap versions of macro (last ID specified).
10. Make native and bitmap versions of macro temporary (last ID specified).

The generation of a static overlay bitmap closes and prints all open pages. Re-rasterizing a macro with an existing static overlay bitmap deletes the existing bitmap and creates a new one from the native version with current environment settings.

The bitmap version can be printed only when the macro is invoked through auto overlay. The native version is used when the macro is called or executed.

Multiple static overlays can be used within the same print job. To switch bitmap images, enable the auto overlay of the macro with the desired bitmap; the previously enabled macro will be disabled.

A printer not supporting static overlay ignores an *Esc&f11X* and uses the native version for printing.

A bitmap's size and orientation correspond to the page size and orientation settings at the time the macro is rasterized.

Macro bitmaps are static and non-positionable: once created, orientation and size cannot be changed. Each bitmap spans the entire current page size and cannot be cropped or repositioned. In addition, a macro generated into a bitmap can span a maximum of one page; that is, no escape sequences or control characters are allowed in the macro that will cause a page to be closed (page eject, FF, etc)

Assigning an existing ID to a new macro deletes the native and bitmap versions of the existing macro.

HP5000 F1XX DEVICE NOTES:

If the page size/orientation of the bitmap to be printed does not correspond to the current page size/orientation settings, the printer stops with the error message "STATIC OVERLAY WRONG PAGE SIZE", and the printer state goes to "NOT READY". The operator can either press CANCEL or READY. If the operator presses READY, printing will continue, but only the data is printed. The bitmap will not be printed until the page size/ orientation settings match that of the bitmap.

The macro that is generated into a bitmap can span a maximum of one page. If a macro being generated into a bitmap spans more than one page, the error message "STATIC OVERLAY PAGE OVERFLOW" is displayed, and the printer state goes to "NOT READY". If the operator presses READY, the printer will resume printing, but the native version will be used if the macro is enabled and invoked for auto overaly.

If the printer does not have enough memory to store the bitmap while the macro is being generated, the portion of the bitmap that has already been created is deleted. The message "GSP MEMORY FULL, STATIC OVERLAY CANNOT BE RASTERIZED" is displayed on the V24 terminal. The printer does not stop. However, subsequent printing if the macro is enabled and invoked for auto overlay will be done with the native version. Note that exceeding the available memory may cause the printer to hang or slow down

Examples

To define a macro with an ID of 7, send:

Esc&f7y0X

...

Escape sequences, control codes, and data

...

Esc&f1X

To make the macro with an ID of 7 permanent, send:

Esc&f7y10X

To enable the macro with an ID of 7 for automatic overlay, send:

Esc&f7y4X

To delete the macro with an ID of 7, send:

Esc&f7y8X

Macro Overlay Example

The following illustrates the definition of a letterhead macro:

Esc&f1Y	Specify the Macro ID as 1.
Esc&f0X	Start the Macro Definition.
Esc&a540h360V	Position logo at (540,360) decipoints in the coordinate system.
*Esc*t150R*	Set graphics resolution to 150 dots-per-inch.
*Esc*r1A*	Start raster image of logo.
*Esc*b60W [Raster data]*	Send the first raster line.
...	
...	
*Esc*b60W [Raster data]*	Send the last raster line.
*Esc*rC*	Stop raster graphics.
Esc&a540h780V	Position for lettering at (540,780) decipoints.
Esc(1X	Select font with ID of 1.
ABC Corp.	Text
Post Office Box 15	Text
Fred, Texas 83707	Text
Esc&a540h960V	Position first rule at (540,960) decipoints.

*Esc*c10v4680H*	Set rule height and width.
*Esc*c0P*	Print the first rule.
Esc&a540h980V	Position the second rule at (540,980) decipoints.
*Esc*c0P*	Print the second rule.
Esc&a1200v540H	Position for first line of text at (540,1200) decipoints.
Esc&f1X	Stop Macro Definition.

This macro can now be **executed, called,** or enabled for **overlay**.

Chapter 19: Status Readback

Contents of this Chapter

This chapter discusses the following PCL commands:

19.1 Introduction

Status readback lets the user obtain status information from the printer, such as available printer memory, a list of fonts and symbol sets, and the ID numbers of macros and user-defined patterns.

Entity Status

Entity status readback provides information about the printer's entities — fonts, symbol sets, macros, or user-defined patterns. For example, the user can ask for a list current fonts and symbol sets, or the ID numbers of macros and user-defined patterns.

Requesting entity status involves the following three steps:

1. Identify **Location Type**
2. Identify **Location Unit**
3. Request status

Terminology

Entity: A font, symbol set, macro, or user-defined pattern.

Entity Status: Provides status for the printer's entities. Each individual entity request is limited to a specific entity and a specific location. To request an entity status, the user must send commands that identify a *location type* and a *location unit*, and then request the status.

Location Type: The memory location that stores an entity — internal ROM, RAM (for downloaded entities), cartridges, user-installable ROM (SIMMs), and one additional location called *currently selected*, which is the active entity, such as the last-selected font or user-defined pattern (currently selected does not apply to macros or symbol sets).

Location Unit: A specific location (or device) within a location type. For example, location unit 1 for location type "cartridge" might refer to the left cartridge; or location unit 1 for location type "downloaded" might refer to temporary, rather than permanent, fonts.

Process

Entity status readback follows the procedure below:

1. Specify Location Type (*Esc*s#T*)
2. Specify Location Unit (*Esc*s#U*)
3. Inquire Entity — identify the entity and request status (*Esc*s#I*)
4. The printer returns a status or error response.

Syntax

Entity status readback responses have the same syntax in both the PCL and PJL contexts:

- Responses start with `PCL<CR><LF>`.
- Each line of the response terminates with `<CR><LF>`.
- Status responses end with `<FF>`.
- PCL status readback responses should be in English.

The basic syntax is:

```
PCL<CR><LF>
INFO TITLE<CR><LF>
KEYWORD1=data1<CR><LF>
KEYWORD2=data2<CR><LF>
...
<FF>
```

TITLE, *KEYWORD*, and *data* are strings that vary depending on the status readback command. Each response is associated with one or more keyword lines. Since, future keywords may be added, applications should ignore lines with unrecognized keywords.

Sample responses:

```
PCL<CR><LF>
INFO MEMORY<CR><LF>
TOTAL=100000<CR><LF>
LARGEST=25000<CR><LF>
<FF>

PCL<CR><LF>
INFO FONTS<CR><LF>
SELECT="<Esc>(8U<Esc>(s0p10.00h12.00v0s0b3T"<CR><LF>
SELECT="<Esc>(0N<Esc>(s0p16.67h8.5v0s0b0T"<CR>LF>
SELECT="<Esc>(s1p_ _v1s0b4101T<Esc>(78X"<CR><LF>
SYMBOLSETS="0N,0U,1U,8U,10U,11U,12U"<CR><LF>
SELECT="<Esc>(8U<Esc>(s1p_ _v0s3b4148T"<CR><LF>
SYMBOLSETS="0N,0U,1U,8U,10U,11U,12U"<CR><LF>
<FF>
```

A status request for non-existent information generates an error response so the host will not wait indefinitely. An error response is returned only for recognized escape sequences. For example, an internal memory error, such as out-of-memory, generates the following error response (An appropriate string replaces TITLE, depending on the requested status):

```
PCL<CR><LF>
INFO TITLE<CR><LF>
ERROR=INTERNAL ERROR<CR><LF>
<FF>
```

NOTE: Examples in the following sections do not show the <CR><LF>, or <FF> characters.

Memory Status

Memory Status requests identify the amount of user-memory available in the printer. This is useful when downloading entities, so printer errors can be avoided. The Free Space command (*Esc*s#M*) is used to request the current number of free bytes in memory.

Synchronization

Status requests and their associated responses are processed in the order received. On some printers, status responses may be stored in a buffer which is not cleared until the responses are read or the printer is turned off. If multiple users are requesting status, it can be difficult to distinguish one user's status response from another. The Echo command (*Esc*s#X*) may be used to synchonize an application's status request to the printer's response.

19.2 Entity Status Commands

Three commands are used for entity status. Inquire Entity (*Esc*s#I*) makes the status request. Location Type (*Esc*s#T*) and Location Unit (*Esc*s#U*) specify the queried entities.

Location Type *Esc * s # t/T*

Specifies the memory location that stores an entity.

Value(#)	=	0	Invalid location
	=	1	Currently selected
	=	2	All locations
	=	3	Internal
	=	4	Downloaded
	=	5	Cartridge
	=	7	User-installable ROM device
Default		=	0
Range		=	0 to 5, 7 (out-of-range values are set to 0, indicating an invalid location)

Location type 0 (default) causes an error response. Location type 1 is the current value of the entity specified by Inquire Entity (current font, current user-defined pattern, etc).

Location Unit *Esc * s # u/U*

Specifies a location or device within a location type.

Value field interpretation depends on location type:

Value(#)	Interpretation	Location Type
*	Invalid location	0
*	Currently selected	1
*	All locations	2
0	All internal	3
0	All downloaded	4
1	Temporary downloaded	4
2	Permanent downloaded	4
0	All cartridges	5
1-n	Cartridges 1 to n (lowest priority)	5
0	All SIMMs	7
1-n	SIMMs 1 to n (lowest priority)	7

Default = 0
Range = 0 to 2^{32} -1

* The location unit is ignored for location type values 0, 1 and 2.

The interpretation of location unit occurs only after Inquire Entity is received. Location unit then determines which set of entities to give status on.

The printer retains the location unit setting, which may be changed by this command.

Location type and unit may be set in any order. Invalid combinations are determined when Inquire Entity is received. Even if the value is out of range, the unit is set to that value so that an appropriate error response will be sent.

Inquire Entity *Esc * s # i/I*

Returns status for all entities of the specified type in the current location.

Value(#)	=	0	Font
	=	1	Macro
	=	2	User-defined pattern
	=	3	Symbol set (for unbound fonts)
	=	4	Font Extended
Default		=	NA
Range		=	0 to 4 (unspecified values produce an error response)

This command does not affect the current values of location type or location unit.

The response to this command depends on the type of entity, as described below.

Font

A font status request may provide different types of information, depending on the request, the font type, and the location. The possible font status keywords are listed below:

SELECT=
SYMBOLSETS=
LOCTYPE=
LOCUNIT=
DEFID=
NAME=

An example font status (value=0) is shown below. The five fonts returned are listed in order:

Internal bitmap (Courier) font
Internal bitmap (Line Printer) font
Internal unbound scalable (CG Times) font
Downloaded bound scalable (CG Palacio) font
Downloaded bound scalable (Dom Casual) font

The response syntax for all location types except the currently selected font is:

```
PCL
INFO FONTS
SELECT="<Esc>(8U<Esc>(s0p10.00h12.0v0s0b3T"
SELECT="<Esc>(0N<Esc>(s0p16.67h8.5v0s0b0T"
SELECT="<Esc>(s1p_ _v1s0b4101T"
SYMBOLSETS="0N,0U,1U,8U,10U,11U,12U"
SELECT="<Esc>(1U<Esc>(s1p_ _v0s0b4111T<Esc>(21X"
SELECT="<Esc>(1U<Esc>(s1p_ _v0s3b4148T<Esc>(22X"
...
```

In the above example, the "SELECT=" keyword identifies five fonts. As described under "SELECT=" below, this keyword varies according to whether the font is bitmap, bound, scalable, unbound scalable, currently selected, or downloaded. Additional keywords are added for unbound fonts ("SYMBOLSETS="), currently selected fonts ("LOCTYPE=" and "LOCUNIT="), and font extended requests ("DEFID=" and "NAME=").

"SELECT=" is returned for **all fonts**. It identifies the font by specifying the font selection characteristics, which are listed in the font selection order of priority (highest first).

For **downloaded** (soft) fonts, the PCL ID selection number is also included at the end. For example, an ID number of 27 is presented as ...<Esc>(27X".

For **bitmap** fonts, all font selection characteristics are listed. In addition, if the bitmap font is a soft font, its PCL ID number is also included at the end of the "SELECT=".

For **scalable** fonts, only relevant characteristics are listed. Symbol set is included for bound scalable fonts. For unbound fonts the "SYMBOLSETS=" keyword is added. Either height or pitch is listed, depending on whether the font is proportional or fixed spaced. Two underscores indicate that the font is scalable.

"SYMBOLSET=" is added for **unbound scalable** fonts. This keyword lists all the symbol sets that can be bound to the font. Duplicate symbol set IDs are removed so that a given symbol set ID will appear only once in the list.

If **currently selected** status is requested, the two keywords below are included. Also, if the currently selected font is scalable, the "SELECT=" keyword shows the current symbol set and current size (point size or pitch). Also, if the currently selected font is a secondary font, the "(" character is replaced by ")".

"LOCTYPE=" is returned only for a currently selected font request. It identifies the location type of the currently selected font.

"LOCUNIT=" is returned only for a currently selected font. It identifies the location unit of a currently selected font request only.

An example of a response for the currently selected font is:

```
PCL
INFO FONTS
SELECT="<Esc>(8U<Esc>(s0p18.00h20.0v0s0b4101T"
LOCTYPE=3
LOCUNIT=1
```

Font Extended Response

The font extended response (value field = 4) is the same as a font response (value field = 1) with two additional keywords, "DEFID=" and "NAME=".

```
PCL
INFO FONTS EXTENDED
SELECT="<Esc>(8U<Esc>(s0p10.00h12.0v0s0b3T"
DEFID="I 21"
NAME="Courier"
SELECT="<Esc>(s1p_ _v1s3b4101T"
SYMBOLSETS="0N,0U,1U,8U,10U,11U,12U"
DEFID="M2 8"
NAME="CG Times     BdIt"
...
```

"DEFID=" identifies the printer font number, which is used by PJL or the control panel to select a default font, consists of two parts, a Location and an ID number, such as "I 21". The possible locations are:

I	-	Internal
C	-	Cartridge
S	-	Permanent soft fonts
Mn	-	SIMMS (where n is the number of the SIMM slot)
NONE	-	Temporary soft fonts

NOTE: The printer font ID number — which is different than the Font ID (*Esc*c#D*) assigned to downloaded fonts — is assigned by the printer. Temporary fonts, which cannot be selected as the default from the control panel or PJL, do not have a printer font ID number.

"NAME=" is returned only for font extended requests. It identifies the name of the font (Courier) and its treatment (bold italic). The name returned is implementation-dependent. If the location type is 1 (currently selected), LOCTYPE and LOCUNIT appear after NAME.

An example of a font extended response is:

```
PCL
INFO FONTS EXTENDED
SELECT="<Esc>(8U<Esc>(s0p10.00h12.0v0s0b3T<Esc>"
DEFID="I 21"
NAME="Courier"
SELECT="<Esc>(s1p_ _v1s3b4101T"
SYMBOLSETS="0N,0U,1U,8U,10U,11U,12U"
DEFID="I 45"
NAME="CG Times  BdIt"
SELECT="...
...
```

An example for a font extended response for the currently selected font is:

```
PCL
INFO FONTS EXTENDED
SELECT="<Esc>(8U<Esc>(s0p18.00hv0s3b4099T"
DEFID="I 21"
NAME="Courier   Bd"
LOCTYPE=3
LOCUNIT=1
```

Macro Response

The macro status response lists all of the macro IDs ("IDLIST=") for the macros in the specified location. ID order is arbitrary and implementation-dependent.

NOTE: Location Type 1 (currently selected) is an invalid location for macros.

An example of a macro status response (value field = 1) is shown below:

```
PCL
INFO MACROS
IDLIST="1,8,3,29,32"
```

User-Defined Pattern Response

The response for user-defined patterns lists all of the user-defined pattern IDs ("IDLIST=") for the patterns in the specified location. ID order is arbitrary and implementation-dependent.

An example of a user-defined pattern response (value field = 2) is shown below:

```
PCL
INFO PATTERNS
IDLIST="9,27,2,456,13,1"
```

If Location Type is 1 (currently selected), "LOCTYPE=" and "LOCUNIT=" lines are added to identify the location (type and unit) of the pattern. An example of a user-defined pattern response for the currently selected pattern is shown below:

```
PCL
INFO PATTERNS
IDLIST="88"
LOCTYPE=4
LOCUNIT=2
```

If the current pattern is set to one of the internal HP-defined patterns (no pattern ID number is assigned), then no number is available and the error response "ERROR=NONE" is returned.

```
PCL
INFO PATTERNS
ERROR=NONE
```

Downloadable Symbol Set Response

The response for symbol sets (value field = 3) lists all of the symbol set IDs ("IDLIST=") for all the symbol sets for unbound fonts in the specified location (type and unit). ID order in the list is arbitrary and implementation-dependent.

Location type 1 (currently selected) is an invalid location for symbol sets and will return "ERROR=NONE".

An example of a symbol set response is:

```
PCL
INFO SYMBOLSETS
IDLIST="8U,0U,2K,11U,8M"
```

Error Responses

If the entity type specified is out of range or unsupported, an invalid entity error is returned. For example, if the Inquire Entity command contains an out-of-range value of 8 (*Esc*s8I*), the following error response is generated:

```
PCL
INFO ENTITY
ERROR=INVALID ENTITY
```

If the entity type is valid, but the location (either type or unit or both) is invalid, or if the specified device is not installed, an invalid location error is returned. For example, if the status for a cartridge type is requested at an out-of-range location unit of 9 (*Esc*s4t9U*), the following error response is generated:

```
PCL
INFO FONTS
ERROR=INVALID LOCATION
```

If the entity type and location are valid, but there are no entities of the specified type in that location, or if the type is inappropriate for the specified entity (e.g. internal user-defined pattern or currently selected macro), an error response is generated. For example if the user requests a downloaded symbol set and there are no downloaded symbol sets, the following error response is generated:

```
PCL
INFO SYMBOLSETS
ERROR=NONE
```

The status response for some requests can be fairly large (e.g., fonts). If the printer runs out of memory processing status responses, it generates an internal error, as shown below:

```
ERROR=INTERNAL ERROR
```

NOTE: The error conditions described above are the only conditions that cause an error response. If the user makes a syntax error in the escape sequence or sends a command that the printer cannot interpret, the printer ignores the command issues no error response.

19.3 Memory Status Commands

The Free Space command (*Esc*s#M*) requests the amount of free user-memory. The Flush All Pages command (*Esc&r#F*) suspends input data until all current pages are printed.

These two commands may be used together or separately. Together, they tell the user how much memory is available without page data. If the printer does not contain sufficient memory to accept a downloaded entity, the user can clear some memory by deleting unneeded entities (fonts, macros, user-defined patterns, etc).

Free Space *Esc * s # m/M*

Requests the current number of bytes of free internal user-memory in the device.

Value(#)	=	Internal memory unit
Default	=	1
Range	=	1 (other values produce an error response)

The response returns two values: **TOTAL=** identifies the total available user-memory in bytes; **LARGEST=** identifies the largest continuous block of available memory in bytes.

The following example shows that the printer has 100,000 bytes of available memory, with the largest continuous block being 25,000 bytes.

```
PCL
INFO MEMORY
TOTAL=100000
LARGEST=25000
```

An out-of-range value field that does not specify a valid internal memory unit causes an "Invalid Unit" error. For example, *Esc*s2M* would produce the following error response:

```
PCL
INFO MEMORY
ERROR=INVALID UNIT
```

Flush All Pages *Esc & r # f/F*

Suspends input stream processing until all pages currently in the device have been printed.

Value(#)	=	0	-	Flush all complete pages
	=	1	-	Flush all pages
Default		=	0	
Range		=	0, 1 (other values are ignored)	

A value of 0 processes only complete pages. A value of 1 performs a conditional formfeed to process partial pages.

For example, if a printer containing two complete pages (A and B) and one partial page (C) receives this command with a value field of 0, it will eject A and B and retain C. If the value field is 1, the printer will process and eject pages A, B, and C.

The printer resumes receiving data when the last page is processed and ejected.

19.4 Response Synchronization

The Echo command allows the user to determine which status response is associated with a given application or user.

Echo *Esc * s # x/X*

The echo command echoes its value field (in ASCII format) back to the host.

Value(#)	=	Echo value (ASCII)
Default	=	0
Range	=	-2^{31} to 2^{31} -1

Multiple users make it difficult to distinguish one user's response from another. This command provides the means to label status responses. Since the user-selected value for the value field is returned, this command can be used as a user identification mark or place holder. Sending this command with a specific value at the beginning and/or the end of a status request delineates status response data.

Since status readback responses are sent in the order requests are received, this command permits the host to know that all PCL status up to this command has already been sent.

As an example, *Esc*s-999X* produces the following status response:

```
PCL
ECHO -999
```

Chapter 20: Obsolete Codes

Contents of this Chapter

This chapter discusses the following PCL commands:

20.1 Procedure for Obsoleting Functionality

The procedure for obsoleting a functionality is:

1. Move the obsolete functionality to the obsolete section of the the Implementor's Guide.

2. Products that contain the obsolete functionality must stop documenting the functionality.

3. After two product generations, when most software supporting the product has stopped utilizing the functionality, it can be removed from the device.

20.2 Movement Segment

Clear All Horizontal Tabs *Esc 3*

Clears all horizontal tab stops.

Clear Horizontal Tab at Current Column *Esc 2*

Clears the horizontal tab stop at CAP, if one is set.

Horizontal Tabulation *Esc I*

Prforms the same function as the Horizontal Tab (HT) character.

Set Horizontal Tab *Esc & a # t/T*

Sets a horizontal tab stop at the specified column.

Value(#)	=	Column number
Range	=	0 to printer's physical limits (unspecified values ignored)

Set Horizontal Tab at Current Column *Esc 1*

Sets a horizontal tab stop at CAP.

Clear Horizontal Tab *Esc & a # u/U*

Clears the horizontal tab stop at the specified column, if one is set.

Value(#)	=	Column number
Range	=	0 to printer's physical limits (unspecified values ignored)

Vertical Tab *VT*

Moves CAP to the same horizontal position on the next vertical tab stop. If no such line exists, VT moves CAP to the first line of the next logical page (i.e., to the default vertical tab stop at top of form).

Half Line Feed *Esc =*

Moves the current active position to the same horizontal position 1/2 line down.

Set Vertical Tab (Relative) *Esc & l 1 m/M*

Sets a vertical tab stop at the current line. It should be possible to set a vertical tab stop in every line within the logical page.

Clear Vertical Tab (Relative) *Esc & l 2 m/M*

Clears the vertical tab stop at the current line, if one is set. The tab stop at top of form cannot be cleared.

Clear All Vertical Tabs *Esc & l 3 m/M*

Clears all vertical tab stops, except the tab at top of form.

Clear Vertical Tab (Absolute) *Esc & l # r/R*

Clears the vertical tab stop in the specified line, if one is set.

Value(#)	=	Line number
Default	=	One vertical tab at top of form
Range	=	0 to logical page bound (unspecified values ignored)

The vertical tab stop at top of form cannot be cleared.

Set Vertical Tab (Absolute) *Esc & l # y/Y*

Sets a vertical tab in the specified line specified.

Value(#)	=	Line number
Default	=	One vertical tab at top of form
Range	=	0 to printer's physical limits (unspecified values ignored)

The first line within a page is line 0 at the top of form.

Print with Paper Motion via VFC   *Esc & l # v/V*

Moves CAP to the first character position of the next line referenced by the specified VFC channel.

Value(#)	=	Channel number
Default	=	NA
Range	=	0 to 16

If no line is referenced, CAP moves to the left margin at the top of form of the next page.

A printer should support a 16 channel VFC. If a nonexistent channel (e.g., 17) is referenced, this command causes a CR-LF. A value field of 0 is a special case, causing the printer to move paper to the left margin and top of form of the next page, unless already there.

Define VFC Table   *Esc & l # W[VFC data]*

Programs the VFC with the data bytes immediately following the terminating character.

Value(#)	=	Number of data bytes following the terminating character
Default	=	NA
Range	=	0 to 256

The data field consists of binary-packed two-byte words, with the most significant byte sent first. Each two-byte word represents one line. The most significant bit of the most significant byte describes channel 16; the least significant bit of the least significant byte describes channel 1. The maximum byte count allowed is twice the maximum supported physical page length. If (byte count / 2) is less than page length, the rest of the VFC table is filled with zeros. If (byte count / 2) is greater than page length, or if the line spacing is 0, or when the byte count is odd, the downloaded VFC is discarded. A 1 in a bit position references that line.

The first occurrence of channel 2 determines text length: if channel 2 is clear, or the first stop is outside the end of the page, text length is equal to the end of logical page. If an orientation, page length, top margin, text length, or line spacing command is received, the programmed VFC is removed and a default VFC calculated from the current line spacing, page length, top margin and text length settings. If the requested VFC table has more lines than the maximum supported, the VFC table is truncated at its maximum limit. If a value field of 0 is received, a default VFC is calculated from the default top margin and text length.

If an illegal command is received, the associated data is stripped so it is not misinterpreted (i.e. neither printed nor acted upon).

NOTE: It is recommended that the VFC when defaulted should not be recalculated until needed (i.e., receipt of *Esc & l # V*); calculating a default VFC table is time consuming.

Default VFC

The default definition for each of the 16 VFC channels is as follows:

Channel	Definition
1	Top of form
2	Bottom of form
3	Single spacing
4	Double spacing
5	Triple spacing
6	Half form
7	Quarter form
8	Tenth line
9	Bottom of form
10	Bottom of form - 1
11	Top of form - 1
12	Top of form
13	Seventh line
14	Sixth line
15	Fifth line
16	Fourth line

NOTE: VFCs, though not required, may be needed for some EDP system level printers.

Definitions

The number of a line is LINE_NUMBER. The first line of a form (i.e. specified by top margin) is line zero. The last line of a page is END_OF_PAGE. The last line of a form is END_OF_FORM. The operator *div* indicates integer division. The operator *leq* indicates "less than or equal".

Channel 1 is active at line 0 only.

Channel 2 is active at END_OF_FORM only.

Channel 3 is active for all lines from 0 through END_OF_FORM.

Channel 4 is active for all lines where LINE_NUMBER is evenly divisible by 2 and 0 *leq* LINE_NUMBER *leq* END_OF_FORM.

Channel 5 is active for all lines where LINE_NUMBER is evenly divisible by 3 and 0 *leq* LINE_NUMBER *leq* END_OF_FORM.

Channel 6 is active for lines 0 and (END_OF_FORM + 2) *div 2*.

Channel 7 is active for lines 0 , (END_OF_FORM + 4) *div 4*, (END_OF_FORM + 2) *div 2*, and (3 * END_OF_FORM + 6) *div 4*.

Channel 8 is active for all lines where LINE_NUMBER is evenly divisible by 10 and 0 *leq* LINE_NUMBER *leq* END_OF_FORM.

Channel 9 is active at END_OF_FORM only.

Channel 10 is active at END_OF_FORM **-** 1 only.

Channel 11 is active at END_OF_PAGE only.

Channel 12 is active at line 0 only.

Channel 13 is active for all lines where LINE_NUMBER is evenly divisible by 7 and 0 *leq* LINE_NUMBER *leq* END_OF_FORM.

Channel 14 is active for all lines where LINE_NUMBER is evenly divisible by 6 and 0 *leq* LINE_NUMBER *leq* END_OF_FORM.

Channel 15 is active for all lines where LINE_NUMBER is evenly divisible by 5 and 0 *leq* LINE_NUMBER *leq* END_OF_FORM.

Channel 16 is active for all lines where LINE_NUMBER is evenly divisible by 4 and 0 *leq* LINE_NUMBER *leq* END_OF_FORM.

Example

The following is an example default VFC for an 11-inch page at 6 lpi (line zero corresponds to the top margin setting). Page length is 66 lines (END_OF_PAGE = 65) and the text length is 60 lines (END_OF_FORM = 59). VFC lines are relative to the top margin, which defaults to 1/2 inch from the top of the physical page (depending on the perforation skip).

Channel	**Lines Active**
1	0
2	59
3	0, 1, 2, ...59
4	0, 2, 4, ...58
5	0, 3, 6, ...57
6	0, 30
7	0, 15, 30, 45
8	0, 10, 20, ...50
9	59
10	58
11	65
12	0
13	0, 7, 14, ...56
14	0, 6, 12, ...54
15	0, 5, 10, ...55
16	0, 4, 8, ...56

20.3 Rendering Segment

Horizontal Raster Resolution *Esc * r # l/L*
Vertical Raster Resolution *Esc * r # v/V*

These commands allow horizontal and vertical raster graphics resolutions to be set independently.

Value(#)	=	Resolution in dots per inch (dpi)
Default	=	Device dependent
Range	=	0 to 32767

GPIS Data Binding (GPIS Vector Graphics) *Esc * t # g/G*

Specifies the vector graphics binding of the data in the data transfer escape sequence (i.e., *Esc*t#W*). The default binding is "none"; a binding must be specified on at least the first transfer of vector graphics data. In the event that vector graphics data is received with an unsupported or no binding specified, the printer will strip and discard the data.

Binding #	Binding Type
0	Exit all vector graphics modes (Default)
1	16-bit binary GPIS binding
2	8-bit binary GPIS binding
3	Character-encoded GPIS binding

Horizontal Picture Dimension *Esc * t # h/H*

NOTE: This sequence has been reassigned to Destination Raster Width (Chapter 13).

The value field specifies the width of the vector graphics picture in decipoints. This value is only meaningful if the device is in Printer Mode. The picture dimension has precedence over the scaling information embedded in subsequent vector graphics metafiles, effectively making all metafile data of the picture library type. The default horizontal size in Printer Mode is the width of the logical page. If either the horizontal or vertical dimensions are 0, the printer does not override the scaling information included in the subsequent vector graphics metafile.

Vector Graphics Operating Mode *Esc * t # m/M*

This escape sequence specifies the operating mode of the printer. A 0 in the value field, the default mode, specifies Plotter Mode. In Plotter Mode all vector graphics data is interpreted literally (i.e., the presentation directives and the print control comes from commands within the vector graphics data stream). A 1 in the value field specifies Printer Mode. In Printer Mode escape sequences control the presentation and printing of the image, overriding the information embedded in the vector graphics data stream. This mode allows the user to control the presentation and printing of the picture without altering the vector graphics data.

Vector Graphics Mapping Mode *Esc * t # n/N*

Mapping mode specifies whether distortion should be permitted when mapping the picture image to the user-specified height and width. This value is only meaningful if the device is in Printer Mode. The default Mapping Mode in Printer Mode is 1 (undistorted). A value field of 2 specifies that distortion is allowed, and a value of 0 defaults to the vector graphics file mapping mode.

Print Control *Esc * t # p/P*

This command controls the printing of vector graphics images. A value field of 0 specifies that the image should be printed and retained. A value field of 1 specifies that the vector graphics picture should be printed and cleared. The picture will be printed with the upper left hand corer of the image at the current active position.

Print Mode (Graphics) *Esc * p # n/N*

Defines the print mode or printing-motion algorithm for graphics.

Value(#)	=	0	Graphics default (no-break algorithm)
	=	1	Print graphics bidirectionally
	=	2	Print graphics left to right
	=	3	Print graphics right to left
	=	4	Conditionally print bidirectional using no-break or smart bidirectional algorithm *
Default	=	0	
Range	=	0 to 4 (command is ignored for unsupported values)	

* DEVICE NOTE: DJ550C prints bidirectional if there are four or more blank raster rows between print passes. DJ500 prints bidirectional if there is only a single blank raster row between passes.

Print Mode (Text) *Esc & k # w/W*

Defines the print mode or printing-motion algorithm for text. See the individual product guides for the meaning of specific values.

Value(#)	=	0	Unidirectional (left-to-right)
	=	1	Bidirectional mode
	=	2	Unidirectional (right-to-left)
	=	3	High intensity mode
	=	5	Text scale OFF
	=	6	Text scale ON
	=	7	Fast high intensity
Default	=		Device dependent (recommend bidirectional)
Range	=		Device dependent

"High intensity" is used for special printer technology media combinations (e.g., thermal inkjet on transparency).

"Text scale ON" allows 66 lines at 6 lpi, or 88 lines at 8 lpi to print on an 11 inch page, even if the effective paper length is shorter than 11 inches. Thus, on an 11 inch page with an 1/2 inch unprintable region, 66 lines (at 6 lpi) would still be printed.

DEVICE NOTE: PJ uses values of 0, 1, 3.

DEVICE NOTE: DJ uses values of 0, 1, 2, 5, 6.

NOTE: This escape sequence terminated with a *W* is an exception in that it **does not** include binary data.

Raster Graphics Quality *Esc * r # q/Q*

Determines the quality with which graphics is printed.

Value(#)	=	0	User-selected
	=	1	Draft
	=	2	High quality
Default	=	0	
Range	=	0 to 2	

Selecting a value of 1 causes raster graphics to be printed at nearly double the normal speed. Print quality is traded for ink savings and increased throughput. The graphics quality is not reset by the End Raster Graphics command (*Esc*rC*).

Vertical Picture Dimension *Esc * t # v/V*

NOTE: This sequence has been reassigned to Destination Raster Height (Chapter 13).

The value field specifies the length of the vector graphics picture in decipoints. This value is only meaningful if the device is in Printer Mode. The picture dimension has precedence over the scaling information embedded in subsequent vector graphics metafiles. The default vertical size in Printer Mode is the current text length of the logical page. If either the horizontal or vertical dimensions ar 0, the printer does not override the scaling information included in the subsequent vector graphics metafile.

GPIS Data Transfer *Esc * t # W [GPIS Data]*

There are two methods of sending data to a vector graphics device: by putting it into a mode where it interprets all data as vector graphics commands (referred to a "stream" mode), or by "encapsulating" the data in an escape sequence. Some devices will implement both modes, the choice of which depends on the operating environment.

Bar Code Label Placement *Esc * z # c/C*

Moves CAP to the specified column position on the pending line.

Value(#)	=	Absolute column position (+ or - is ignored)
Default	=	NA
Range	=	NA (values less than the current active position (CAP) move the CAP one column to the right)

The value is based on the pitch of the current character set, or HMI if it is implemented.

This command provides device-independent bar code positioning. Like the Horizontal Positioning (*Esc&a#C*) command, starting location is specified by a column number. Horizontal spacing is based on the current font pitch, or on HMI if it is implemented.

NOTE: Because a bar code has no standard width, the width can be device specific.

Bar Code Label Height *Esc * z # h/H*

Defines the height of the label in tenths of an inch.

Value(#)	=	Label height (tenths of an inch)
Default	=	6 (6/10 of an inch)
Range	=	0 to 32767

A value of 0 sets bar code height to the current line spacing (e.g., 6 or 8 lines per inch).

DEVICE NOTE: The HP 2934A product supports a range of 0 to 99.

Bar Code Header Control *Esc * z # q/Q*

Controls the printing of the bar code header string.

Value(#)	=	0	Disables the printing of the header
	=	1	Prints the header above the bar code
	=	2	Prints the header below the bar code

Default	=	1
Range	=	0 to 2

DEVICE NOTE: The HP 2934A supports a range of 0-1. Printing the header below the bar code is not supported. The HP 2934A prints all valid characters in a bar code label string and ignores illegal characters. The HP 2934A truncates at the printer's physical right limit, not the right margin.

Bar Code Label   *Esc * z <Bar code> z/Z*

Sends the bar code, specified by an alphanumeric string in angle brackets, to the printer.

When execution this command, the device prints all buffered bar code data, moves to the left margin, and spaces down the height of the bar code. Start/Stop and Check characters are added as necessary, as defined by the current bar code selected

DEVICE NOTE: If an illegal character in a bar code is received, the 256x products make the bar code unreadable. The 256x products truncate the bar code at the right margin.

Bar Code Label X Offset   *Esc * z # x/X*

Defines the relative horizontal offset in dots for the next label.

Value(#)	=	Relative horizontal offset (+ or - is ignored)
Default	=	NA
Range	=	0 to 32767

The value field specifies the number of dots the label is offset to the right.

DEVICE NOTE: For the HP 2934A product, the dot spacing is 1/90 inch.

Bar Code Selection *Esc * z # v/V*

Selects the bar code type.

Value	Bar Code Type
0	Code 3 of 9
1	Industrial 2 of 5
2	Matrix 2 of 5
3	User-defined
4	2 of 5 interleaved
5	Codabar
6	MSI/Plessey
8	UPC A
9	UPC E
10	EAN 8
11	EAN 13
12	Code 128
13	Code 93
14	Extended code 3 of 9

Default = 0
Range = 0 to 14 (unsupported types are ignored: the previous type remains in effect)

Illegal or unsupported bar code types are ignored; the previously selected bar code type remains in effect.

DEVICE NOTE: The HP 256X products support 0,1,4,8,9,10, and 11. The 256x products cannot merge bar codes of different types on the same line.

DEVICE NOTE: The HP 2934A supports values 0 to 4.

Bar Code Print Density:

Esc * z # r/R	Bar Code Wide Bar Width
Esc * z # s/S	Bar Code Narrow Bar Width
Esc * z # t/T	Bar Code Wide Space Width
Esc * z # u/U	Bar Code Narrow Space Width

Specify the dot width of the wide bar, narrow bar, wide space, and narrow space elements.

Value(#) = Width in dots (half dot increments; 0.5)
Default = Device dependent
Range = 1 to 9.5

The width of these elements determine the density at which a bar code symbol is printed.

DEVICE NOTE: For the HP 2934A, the dot spacing is 1/90 inch.

DEVICE NOTE: The HP 256X products do not support this sequence.

End Raster Graphics *Esc * r B*

This command ends Raster Mode. It signifies the end of the transfer of a raster graphics image and ends the current raster row. Commands locked out by Start Raster Graphics (*Esc*r#A*) are enabled. If a value field is received, it is ignored and the command still executed.

NOTE: It is recommended that *Esc*rC* be used rather than *Esc*rB*.

X Offset *Esc * b # x/X*

Defines a temporary horizontal offset, which is to be in effect only for the next raster line received.

Value(#)	=	Number of pixels in the offset (should be a multiple of 8) *
Default	=	NA
Range	=	0 to 32767

* Values that are not a multiple of eight are rounded down to the nearest multiple.

The X Offset plus the left graphics margin determines the horizontal location of the next raster line.

Text Color *Esc & v # s/S*

Selects the predefined color for text.

Value(#)	=	0	Black
	=	1	Red
	=	2	Green
	=	3	Yellow
	=	4	Blue
	=	5	Magenta
	=	6	Cyan
	=	7	White
Default	=	0	
Range	=	0 to 7 (command is ignored for other values)	

This command uses the above fixed color table, which is different from the color palette used for raster graphics.

Configure Raster Data (format 1) *Esc * g # W*

The Configure Raster Data (CRD) command configures the device for complex raster data transfers.

Value(#)	=	Number of data bytes
Default	=	NA
Range	=	6 to 4,294,967,295

DEVICE NOTE: DeskJet 520, and DeskJet 560C support a maximum range of 32767.

DEVICE NOTE: DeskJet 850C locks out Simple Color (*Esc*r#U*) and Raster Resolution (*Esc*t#R*) after a valid CRD until the receipt of an *EscE* or equivalent device reset, or a CRD command with a value field of 0 (*Esc*g0W*). A CRD command with a 0 value field unlocks Simple Raster (*Esc*r#U*) and Raster Resolution (*Esc*t#R*), resets the horizontal and vertical resolutions to 75 ppi, and resets the raster mode to one-bit direct (monochrome).

The data field must contain byte-aligned binary data, not ASCII. Unsupported or out-of-range values or other invalid configurations cause the entire command to be ignored and the data bytes to be discarded. Extra bytes are discarded. Value-field signs are ignored.

CRD provides raster configurations having one or more of the following characteristics:

- Horizontal resolution differs from vertical resolution.
- Color components may be specified at different resolutions.
- Each component may be specified at a multiple intensity level.
- Indexed raster data is separate from palette configuration.
- Direct raster data does not affect the palette configuration.
- Mixed indexed raster data and monochrome data for sleek top to bottom printing.

Reset (*EscE*), Simple Color (*Esc*r#U*), and Configure Image Data (*Esc*v#W*) override a CRD configuration.

Raster resolution is determined by the most recent CRD even if it is of another format, or by the Raster Resolution (*Esc*t#R*) command. A Raster Resolution command issued after CRD:

- Sets horizontal and vertical resolutions for each component to the specified resolution.
- Matches the strip height for each component to the new resolution.
- Establishes a pixel unit size for source raster height and width.
- Zeros the seed rows.
- Sets the number of levels for each component to 2.
- Has no effect on the number of expected components (raster indexing mode is unchanged).

The lowest CRD horizontal resolution determines the horizontal pixel size for the Source Raster Width (*Esc*r#S*), to which all the planes of all the rows for all components are zero-filled or clipped. Similarly, the lowest CRD vertical resolution determines the vertical pixel size for Source Raster Height (*Esc*r#T*), which defines raster area vertical height.

Missing data is zero-filled if Raster Resolution is changed or a Y Offset (*Esc*b#Y*) command is received after only part of the data defining a strip is sent.

NOTE: The Seed Row Source (*Esc*b#S*) command should not be used for CRD-configured raster data because the result is undefined.

As usual, the Transfer Raster commands (*Esc*b#W*, *Esc*b#V*) download raster data, and the Compression Method (*Esc*b#M*) command defines the compression rules.

CRD has different forms based on the format byte. Each form of CRD shares only the first byte, the format number, with the other forms of CRD.

NOTE: Format 1 was used only on DeskJet 520 and DeskJet 560C and is being obsoleted in favor of Format 2, which should be used on all future implementations.

Format 1 Complex Direct Planar Alternative

Format 1 provides configuration for raster transfers where:

- Color components may have different resolutions.
- Horizontal and vertical resolutions may be different.
- Color components may have different intensity levels.

NOTE: Format 1 was used only on DeskJet 520 and DeskJet 560C and is being obsoleted in favor of Format 2, which should be used on all future implementations.

The data for each color component is organized into strips containing one or more rows. The lowest vertical-resolution determines the vertical displacement of all the components in a strip; higher vertical-resolution components require more rows than lower vertical-resolution components to cover the same vertical displacement. The horizontal resolution of a component determines the number of bytes needed to define a row (e.g., 600 dpi rows contain 8 times as many bytes as 75 dpi rows). Higher horizontal and vertical resolutions must be multiples of lower horizontal and vertical resolutions, respectively. The CRD command is ignored for unsupported resolutions.

As shown below, the information specified for each color component consists of:

- horizontal resolution.
- vertical resolution.
- the number of levels of grayscale (continous tone) data.

Byte	**15 (MSB)**	**7 (LSB) 0**	**Byte**
0	Format=1 (UBYTE)	Number of components (UBYTE)	1
2	Horizontal Resolution for Component 1 (UINT16)		3
4	Vertical Resolution for Component 1 (UINT16)		5
6	Number of Levels for Component 1 (UINT16)		7
. . .	...		...
6(n-1)+2	Horizontal Resolution for Component n (UINT16)		6(n-1)+3
6(n-1)+4	Vertical Resolution for Component n (UINT16)		6(n-1)+5
6(n-1)+6	Number of Levels for Component n (UINT16)		6(n-1)+7

Byte 0: Format #

Value	=	1
Default	=	1
Range	=	1 to 5 (command is ignored for unsupported values)

Format 1 configures for direct monochrome, CMY, or KCMY data in planar format.

NOTE: Format 1, implemented only on DeskJet 520 and DeskJet 560C, should not be documented externally.

Byte 1 Number of Components

Value =	1	Monochrome (K)
=	3	Three components (CMY)
=	4	Four components (KCMY) (command is ignored for unsupported values)

Specifies the number of expected components. When the value is 3, the first component is cyan, the second magenta, and the third yellow. When the value is 4, the first component is cyan, the second magenta, the third yellow, and the fourth black. The raster transfer commands (*Esc*b#V*, *Esc*b#W*) must use the same component ordering, except in the four-components case, where the data is transferred black first, cyan, magenta, and finally yellow.

DEVICE NOTE: DeskJet 520 supports only 1 component.

Bytes 2&3, 8&9, etc Horizontal Resolution

Value = 1 to 65535 (command is ignored for unsupported values)

Specifies the horizontal resolution of each component in pixels per inch (ppi) of the source pixel. The “horizontal” axis is determined by the current Raster Presentation (*Esc*r#/F*), Orientation (*Esc&l#O*), and Print Direction (*Esc&a#P*). Bytes 2 & 3 specify the horizontal resolution for component one, bytes 8 & 9 specify the horizontal resolution for component two, etc. The highest horizontal resolution must be a multiple of lower horizontal resolutions.

DEVICE NOTE: On DeskJet 520 and DeskJet 560C, a CRD command with a horizontal resolution of 600 and a vertical resolution of 300 will evoke Letter Quality and override Draft if in effect.

DEVICE NOTE: On DeskJet 520 and DeskJet 560C, all non-black CRD horizontal resolutions must match the resolution specified by the Raster Resolution command (*Esc*t#R*). Black CRD horizontal resolution must also match the Raster Resolution command (*Esc*t#R*) unless the black horizontal resolution is 600 and the resolution specified by the Raster Resolution command (*Esc*t#R*) is 300.

Bytes 4&5, 10&11, etc Vertical Resolution

Value = 1 to 65535 (command is ignored for unsupported values)

Specifies the vertical resolution of each component of the source pixel in pixels per inch. The “vertical” axis is determined by the current Raster Presentation (*Esc*r#/F*), Orientation (*Esc&l#O*), and Print Direction (*Esc&a#P*). Bytes 4 & 5 specify the vertical resolution for component one, bytes 10 & 11 specify the vertical resolution for component two, etc. The highest vertical resolution must be a multiple of lower vertical resolutions.

DEVICE NOTE: On DeskJet 520 and DeskJet 560C, a CRD command with a horizontal resolution of 600 and a vertical resolution of 300 will evoke Letter Quality and override Draft if in effect.

DEVICE NOTE: On DeskJet 520 and DeskJet 560C, CRD vertical resolution must match the resolution specified by the Raster Resolution command (*Esc*t#R*).

Bytes 6&7, 12&13, etc Number of Intensity Levels

Value = 2 to 255 (command is ignored for unsupported values)

Specifies the number of intensity or grayscale levels for each component. A level of 2 allows only two intensities (a pixel is either on or off). A level of 4 allows four intensities, etc. The lowest intensity is 0; the highest intensity is Number of Levels - 1.

Bytes 6 and 7 specify the number of levels for component one, bytes 12 and 13 specify the number of levels for component two, etc.

The number of planes expected for each row is the ceiling function of log base 2 of the number of levels. For example, three levels require two planes. The lowest order plane is transmitted first, and the highest order plane last for a given row.

DEVICE NOTE: DeskJet 520 and DeskJet 560C support only a level of 2.

Acceptable Combinations of Parameters for Format 1

A = Number of Components
B = Horizontal Resolution of Component 1
C = Vertical Resolution of Component 1
D = Number of Intensity Levels of Component 1
E = Horizontal Resolution of Component 2
F = Vertical Resolution of Component 2
G = Number of Intensity Levels of Component 2
H = Horizontal Resolution of Component 3
I = Vertical Resolution of Component 3
J = Number of Intensity Levels of Component 3
K = Horizontal Resolution of Component 4
L = Vertical Resolution of Component 4
M = Number of Intensity Levels of Component 4

DEVICE NOTE: Acceptable combinations of parameters for DeskJet 520

DEVICE NOTE: Acceptable combinations of parameters for DeskJet 560C

Example 1 of Format 1

Assume the following CRD command is sent:

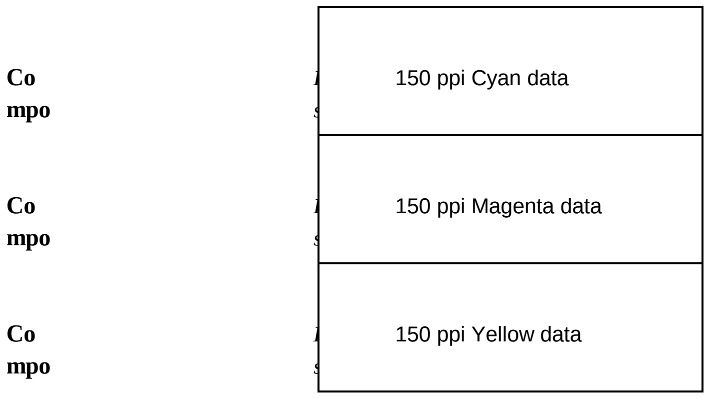
Format	=	1
Number of components	=	3
Horizontal resolution component 1	=	150
Vertical resolution component 1	=	150
Levels for component 1	=	2
Horizontal resolution component 2	=	150
Vertical resolution component 2	=	150
Levels for component 2	=	2
Horizontal resolution component 3	=	150
Vertical resolution component 3	=	150
Levels for component 3	=	2

In this example, the vertical displacement of all four strips is 1/150", since that is the displacement of the component with the lowest vertical resolution (150 ppi).

Component 1 consists of one row containing one plane of cyan. Only one row is needed at 150 ppi vertical resolution to cover 1/150" of vertical displacement. And only one plane is needed to encode two intensity levels ($\log_2 2=1$).

Components 2 and 3 are encoded similarly. Component 2 consists of one row containing one plane of magenta; component 3 consists of one row containing one plane of yellow.

As shown below, components are sent in order from 1 to 3.



Example 2 of Format 1

Assume the following CRD command is sent:

Format	=	1
Number of components	=	4
Horizontal resolution component 1	=	300
Vertical resolution component 1	=	300
Levels for component 1	=	2
Horizontal resolution component 2	=	300
Vertical resolution component 2	=	300
Levels for component 2	=	2
Horizontal resolution component 3	=	300
Vertical resolution component 3	=	300
Levels for component 3	=	2
Horizontal resolution component 4	=	600
Vertical resolution component 4	=	300
Levels for component 4	=	2

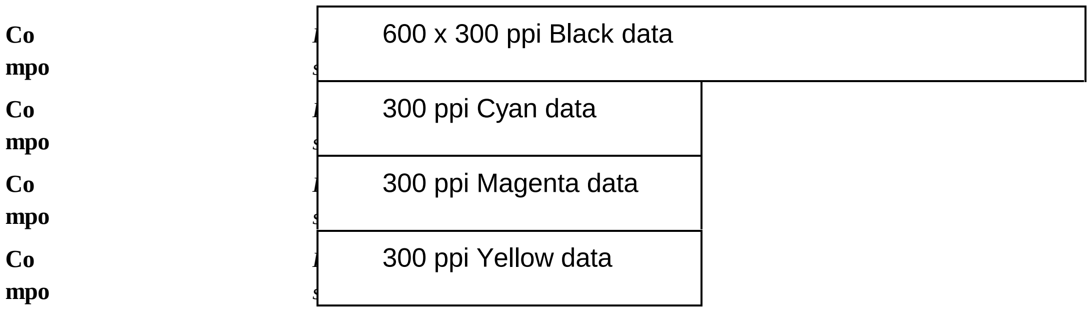
In this example, the vertical displacement of all four strips is 1/300", since that is the displacement of the component with the lowest vertical resolution (300 ppi).

Component 1 consists of one row containing one plane of cyan. Only one row is needed at 300 ppi vertical resolution to cover 1/300" of vertical displacement. And only one plane is needed to encode two intensity levels ($\log_2 2=1$).

Components 2 and 3 are encoded similarly. Component 2 consists of one row containing one plane of magenta; component 3 consists of one row containing one plane of yellow.

Component 4 consists of one row containing one plane of 600 ppi horizontal by 300 ppi vertical black. As shown below, a 600 x 300 ppi plane contains twice the data of a 300 x 300 ppi plane.

Transfer Raster by Plane (*Esc*b#V*) is used to send all the planes of all the rows except the last plane of the last row within a strip, which uses Transfer Raster by Row/Block (*Esc*b#W*). The least significant plane of a row is sent first. Components are sent in the order: 4, 1, 2, 3.



20.4 Font Segment

Large Character Placement *Esc * c # c/C*

Moves CAP to the specified column position on the current line

Value(#)	=	Absolute column position (+ or - is ignored)
Default	=	NA
Range	=	NA (values less than CAP move CAP one column to the right)

The value is based on the pitch of the current character set, or HMI if implemented.

The command provides device-independent large character positioning. Like the Horizontal Positioning command (*Esc&a#C*), starting location is specified by a column number. Horizontal spacing is based on the current font pitch, or on HMI if implemented.

Large Character Size *Esc * c # m/M*

Defines the horizontal and vertical magnification of the standard character size.

Value(#)	=	Character size in # times the standard cell size)
Default	=	1
Range	=	1 to 28 (values less 1 are set to 1; values greater than 28 are set to 28)

Large Character Horizontal Size *Esc * c # s/S*

NOTE: This sequence has been reassigned to Symbol Set Code (Chapter 12).

Defines the horizontal magnification of the standard character size.

Value(#)	=	Horizontal character size in # times the standard cell size
Default	=	1
Range	=	1 to 28 (values less 1 are set to 1; values greater than 28 are set to 28)

The sequence provides for independent control of the horizontal and vertical magnification when used in conjunction with the Large Character Size command (*Esc*c#M*).

Large Character Tab *Esc * c # n/N*

Moves CAP to the next horizontal tab stop on the current line. The location of the tab stop is determined by the current value of HMI, not by the horizontal magnification of the large characters. A value parameter is ignored.

Large Character X Offset *Esc * c # x/X*

NOTE: This sequence has been reassigned to Picture Frame Horizontal Size (Decipoints) (Chapter 16).

Defines the relative horizontal offset in dots for the next large character.

Value(#)	=	Relative horizontal offset in dots (+ or - is ignored)
Default	=	NA
Range	=	0 to 32767

The value field specifies the number of dots to the right to place the next large character symbol.

DEVICE NOTE: The dot spacing is 1/90 inch for the HP 2934A.

Large Character Print Data *Esc * c <Text> z/Z*

Sends the large characters to the printer. Large Characters are sent to the device as an alphanumeric string enclosed in angled brackets. To specify an angled bracket in the data, a left angled bracket must precede it.

Upon termination of this escape sequence, the device prints all the buffered large character data at the selected magnification and position.

Control codes are treated as invalid characters.

DEVICE NOTE: In the HP 2934A, invalid characters in the alphanumeric string are ignored and replaced with a space.

Character Set Selection Control *Esc & k # i/I*

Controls whether the current alternate character set is to be accessed via SI/SO control or via the eighth bit. If a 0 is received in the value field, SI/SO should be used. If a 1 is received, the eighth bit should be used. The requirement for this command does not exist, since the method the user desires is implied by the data he sends. If the eighth bit is a 1 or if a SO is received, the secondary character set should be used. If the eighth bit is a 0 and a SO has not been received (or a SI has), the primary character set should be used. The default access to the secondary character set should be via SI/SO. Note: This escape sequence is in conflict with the 8-bit character set standard.

Primary and Secondary Pitch Mode   *Esc & k # s/S*

Defines the pitch mode for both primary and secondary fonts.

Value	Pitch (Recommended Nominal)
0	Normal (10.0)
1	Double Wide (5.0)
2	Compressed (16.5 - 16.7)
3	Double Wide Compressed (8.25 - 8.35)
4	Elite (12.0)
8	Double Size (Double wide/high)

Default	=	Defined by printer default set (generally 10 cpi)
Range	=	0 to 8 (unavailable values are ignored)

This command defaults the HMI value to the width of the newly invoked font's SPACE.

Auto Centering and Justification   *Esc & l # j/J*

Centers or justifies print.

Value(#)	=	0	No centering or justification
	=	1	Right flush (optional)
	=	2	Centering
	=	3	Justification
Default		=	0
Range		=	0 to 3

Right flush moves the entire line to the right margin. If the line is wider than the margins, no change is made to the line. The right flush parameter is optional; centering and justification may be implemented without implementing right flush.

Centering positions the line between the left and right margins. If the line is wider than the margins, no centering is performed.

Justification takes a line of data and expands it from the position of the first printable character to the right margin. Leading spaces, tabs, or pen moves are not used in justification. The printer may stretch only the spaces between the words, or it may stretch the spacing between individual characters.

Character Set Default Control   *Esc & s # i/I*

Controls the action that should be taken when a request for an unavailable character set is received. If a 0 is received in the value field, a "blank" character set should be the default for an unavailable character set. If a 1 is received, the default for an unavailable character set should be to ignore the escape sequence. This command exists to provide compatibility with early terminal devices. The default should be to ignore a request for a character set which is not available.

Primary Font Placement (superior/inferior)   *Esc (s # u/U*
Secondary Font Placement (superior/inferior)   *Esc) s # u/U*

Specifies the font superior/inferior attribute.

Value(#)	=	0	Normal placement
	=	1	Inferior
	=	-1	Superior
Default	=	0	
Range	=	-1 to 1 (unspecified values are ignored)	

This command does not define character placement implementation: one printer may raise or lower a normal-height character by half a line; another may reduce the character size.

Print Enhancement (underline) *Esc & k # e/E*

Determines whether underline works on a mode basis or a line-by-line basis

Value(#)	=	0	Line-by-line
	=	1	Mode
Default	=	1	
Range	=	0,1 (command is ignored for other values)	

Mode basis means that a state is active until turned off; line-by-line means the state turns off at the end of the print line.

Shift-In/Shift-Out Control *Esc & k # f/F*

Determines whether the shift-in and shift-out control codes work on a mode basis or a line-by-line basis

Value(#)	=	0	Line-by-line (the state turns off at the end of the print line)
	=	1	Mode (the state is active until turned off)
Default	=	1	
Range	=	0,1 (command is ignored for other values)	

Font Quality (Primary) *Esc (s # q/Q*
Font Quality (Secondary) *Esc) s # q/Q*

Specifies font quality.

Value(#)	=	0	Data processing
	=	1	Near letter quality
	=	2	Letter quality
Default	=	Device dependent	
Range	=	0 to 2.	

This attribute determines printing quality, not the actual font representation. If the requested quality is unavailable, the closest fit is chosen.

DEVICE NOTE: The 256X series implements -1 as less than data processing quality.

20.5 Page Presentation Segment

Set Left Margin at Current Horizontal Position *Esc 4*

Sets the left margin to the current active horizontal position.

Set Right Margin to Current Horizontal Position   *Esc 5*

Sets the right margin to the current horizontal position.

20.6 Device Control and Diagnostics Segment

I/O Status Request   *Esc ?*

This request is executed immediately when received (i.e., when the I/O checks for *Esc?* in the input data stream). *Esc?* is always acted upon, even in binary data; it must stripped from the input data stream if it appears in non-binary data. With an RS-232-C interface, *Esc?* must be followed immediately by a *DC1* to cause the printer to transmit the one-byte response, whose bits are defined as follows:

Bit	Definition
0 (LSB)	1 Paper out, manual feed, print error, operator and service calls.
1	1 Temporary printer busy (buffer full, warming up).
2	1 Printer off-line.
3	1 I/0 data error received since last status check. 1) Parity error 2) Framing error 3) Overrun (input buffer or UART RAM)
4	Unused. Set to one.
5	Unused. Set to one.
6	Unused. Set to one.
7 (MSB)	Parity if seven-bit mode. Zero if eight-bit mode.

Primary Status Request *Esc ^*

Requests the printer to return information on symbol set status, print modes, line density, power-on configuration modes, and hardware status. The information is in 4-bit groups encoded as characters from column 3 of the ASCII table. The number of characters varies, but the information is preceded by *Esc * and is terminated by CR-LF.

Primary Status:	Request	*Esc^* (RS-232-C and RS-422 require a DC1 trigger)
	Response	*Esc\<six status bytes>CR-LF*

Return Model Number *Esc * r # K*

Requests that the printer return its Hewlett-Packard model number followed by CR-LF. Using serial interface the printer must receive a DC1 trigger immediately after the terminator before sending the model number. The return model number escape sequence does not meet the general theory of zi (parameter character) and Zn (terminator character); only Zn is effective.

Return Model Number *Esc * s # ^*

Requests that the printer return its Hewlett-Packard model number followed by CR-LF. Using serial interface the printer must receive a DC1 trigger immediately after the terminator before sending the model number. The return model number escape sequence does not meet the general theory of zi (parameter character) and Zn (terminator character); only Zn is effective.

QMS Magnum-V Enable/Disable *Esc * t # F*

Enables or disables the QMS Magnum-V language interpreter.

Value(#)	=	0	Disable QMS Magnum-V
	=	1	Enable QMS Magnum-V
Default	=	0	
Range	=	0,1	

Transfer to the On-Line State *Esc n*

Causes no action in a printer. This sequence should not be used. A more workable solution is for the printer to power up in the same off/on-line state it was in when it powered down if the machine has nonvolatile RAM, or to power up in the on-line state for recoverability.

Transfer to the Off-Line State *Esc o*

Transfers the printer to the off-line state if presently on-line. This escape sequence can be used by devices which require operator intervention. Upon receipt of this escape sequence, all buffered data should be processed.

Head View Enable/Disable *Esc & k # v/V*

Controls head view mode. When head view is enabled, the print head moves away from the current active position so the most recent print can be viewed when not actively printing. Printers with a keyboard interface require this feature.

Value(#)	=	0	Enables head view mode
	=	1	Disables head view mode

Default	=	1
Range	=	0,1

Appendix B: Product Support Matrix

Appendix B is a table describing the differences in PCL implementation for recent HP devices.

In the table, the value in parentheses following a command parameter identifies the parameter value field value.

Printer Control

Command	Code	Range																	
Dry Timer	***Esc &b #T***	No. of seconds																	
Gray Balance	***Esc &b #B***	Default (off) (0)																	
		Enable (1)																	
		Disable (2)																	
Media Destination	***Esc &l# G***	Automatic (0)																	
		Dest tray 1 (1)																	
		Dest tray 2 (2)																	
		Dest tray 3 (1)																	

Media Source	***Esc &l#H***	Eject page (0)																	
		Tray 1 (1)																	
		Manual feed (2)																	
		Man envel feed (3)																	
		Tray 2 (4)																	
		Optional source (5)																	
		Envelope feeder (6)																	
		Autoselect (7)																	
Media Type	***Esc &l#M***	Plain paper (0)																	
		Bond (1)																	
		Special (2)																	
		Glossy (3)																	
		Transparency (4)																	
Negative Motion	***Esc &a#N***	0,1																	
Number of Copies	***Esc &l#X***	1-99																	
Peripheral Configuration	***Esc &b#W [data]***	0-32767																	
Presentation Mode	***Esc *r#F***	0,3																	

Printer Control (continued)

Command	Code	Range																	
Print Mode (text)	***Esc &k #W***	Left to right (0)																	
		Bidirectional (1)																	
		Right to left (2)																	
		Text scale OFF (5)																	
		Text scale ON (6)																	
Print Quality	***Esc *o# M***	Economy (-1)																	
		Normal (0)																	
		Presentation (1)																	
Registration (Left)	***Esc &l# U***	(-32767)-(+32767)																	
Registration (Top)	***Esc &l# Z***	(-32767)-(+32767)																	
Reset	***Esc E***	~																	
Self test	***Esc z***	~																	
Simplex/Duplex	***Esc &l# X***																		
Unit of Measure	***Esc &u #D***	~																	

Text and Characters

Command	Code	Range																	
Display Functions	***Esc Y***	Turn on																	
	Esc Z	Turn off																	
End-of-Line Wrap	***Esc &s#C***	Turn on (0)																	
		Turn off (1)																	
Escapement Encapsulated Text	***Esc &p#W***	4-32767																	
Underline	***Esc &d#D***	Default (0)																	
		Single fixed (1)																	
		Double fixed (2)																	
		Single floating (3)																	
		Double floating (4)																	
	Esc &d@	Turn off																	
Text Parsing Method	***Esc &t#P***	1-byte processing (0,1)																	
		2-byte (2)																	
		2-byte for chars, 1-byte for codes (21)																	

		Shift JIS (31)																	
		Depends on 8th bit of first byte (38)																	
Text Path Direction	***Esc &c #T***	Horizontal (0)																	
		Vertical (-1)																	
Transparent Data Mode	***Esc &p #X[data]***	Number of bytes																	

Text and Characters (continued)

Command	**Code**	**Range**																	
Line Termination	***Esc &k #G***	CR⇒CR; LF⇒LF; FF⇒FF																	
		CR⇒CR+LF; LF⇒LF; FF⇒FF																	
		CR⇒CR; LF⇒CR+LF; FF⇒CR+FF																	
		CR⇒CR+LF; LF⇒CR+LF; FF⇒CR+FF																	

Page Control

Command	**Code**	**Range**																	
Clear Horizontal Margins	***Esc 9***	~																	
Job Separation	***Esc &l1T***																		
Line Spacing	***Esc &l#D***	Number of lines (lines / inch)																	
Top Margin	***Esc &l#E***	0-Page length (number of lines)																	
Left Margin (column #)	***Esc &a#L***	0-Right margin																	
Right Margin (column #)	***Esc &a#M***	L margin to R logical page limit																	
Orientation	***Esc &l#O***	Portrait (0)																	
		Landscape (1)																	
		Reverse portrait (2)																	
		Rev landscape (3)																	

Page Control (continued)

Comm and	Cod e	Range																	
Page Length	***Esc &l# P***	0 to page size (number of lines)																	
Side Selecti on	***Esc &a #G***																		
Page Size	***Esc &l# A***	Default-current (0)																	
		Executive (1)																	
		US Letter (2)																	
		US Legal (3)																	
		Ledger 11x17 (6)																	
		ISO A3 (27)																	
		ISO A4 (26)																	
		ISO A5 (25)																	
		ISO & JIS A3 (27)																	
		JIS B5 (45)																	
		JIS B4 (46)																	
		Hagaki postcard (71)																	
		Oufuku-Hagaki postcard (72)																	

		Card, 4x6 in (74)																	
		Card, 5x8 in (75)																	
		Monarch Envelope (80)																	
		# 10 Envelope (81)																	
		Int'l DL Envelope (90)																	
		Int'l C5 Envelope (91)																	
		Int'l C6 Envelope (92)																	
		Intl. B5 Envelope (100)																	
		Custom (101)																	

Page Control (continued)

Command	**Code**	**Range**																	
Perforation Skip Mode	***Esc &l#L***	0,1																	
Text Length	***Esc &l#F***	0-(Page length - Top margin)																	
Vertical Motion Index (VMI)	***Esc &l#C***	0-Page length																	

CAP Movement

Command	**Code**	**Range**																	
Horizontal Motion Index (HMI)	***Esc &k#H***	0-32767																	

Move CAP Horizontal (Columns)	***Esc &a #C***	(-32767)-(+32767)																	
Move CAP Horizontal (Decipoints)	***Esc &a #H***	(-32767)-(+32767)																	
Move CAP Horizontal (Dots)	***Esc *p# X***	(-32767)-(+32767)																	
Move CAP Vertical (Rows)	***Esc &a #R***	(-32767)-(+32767)																	
Move CAP Vertical (Decipoints)	***Esc &a #V***	(-32767)-(+32767)																	
Move CAP Vertical (Dots)	***Esc *p# Y***	(-32767)-(+32767)																	
Print Direction	***Esc &a #P***	0, 90, 180, 270																	
Push/Pop CAP	***Esc &f# S***	0-1																	
Space	***SP***	~																	

Font Selection

NOTE: **Primary** commands below become **Secondary** by replacing the left parenthesis with a right parenthesis.

Command		Range																
Alphanumeric ID		0-5, 100																
Symbol Set (Primary)		See Chap 9																
Spacing (Primary)		Fixed (0)																
		Proportional (1)																
Pitch (Primary)		> 0 (2 decimal places)																

Height (Primary)		> 0 (2 decimal places)																
Style (Primary)		0-32767																
Stroke Weight (Primary)		-7 to 7																
Typeface (Primary)		See Chap 9																
Quality (Primary)		Letter (2)																
		NLQ (1)																
Shift In (Select Primary Font)		~																
Shift Out (Select Secondary Font)		~																

Font Downloading

Comm and	Cod e	Range																	
Charact er Code	Esc *c# E	0-32767																	
Font ID	Esc *c# D	0-32767																	
Downlo ad Char	Esc (s# W[dat a]	0-32767																	
Downlo ad Font	Esc)s# W[dat a]	0-32767																	
		Format 0																	
		Format 10																	
		Format 11																	
		Format 15																	
		Format 16																	
		Format 20																	
Font Control	Esc *c# F	0-6																	
Select Primary Font by ID	Esc (#X	0-32767																	
Select Primary Font by ID	Esc)#X	0-32767																	

User-Defined Symbol Set

Command	Code	Range																	
Download Symbol Set	***Esc (f#W[data]***	0-32767																	
Symbol Set Control	***Esc *c#S***	0-5																	
Symbol Set ID	***Esc *c#R***	0-32767																	

Raster

Command		**Range**																	
Compression Method		Unencoded, row-based (0)																	
		Run-Length (1)																	
		TIFF PackBits (2)																	
		Delta Row (3)																	
		Unecoded, block-based (4)																	
		Adaptive (5)																	
		CCITT Group 3 one-dimensional (6)																	
		CCITT Group 3 two-dimensional (7)																	
		CCITT Group 4 (8)																	
		Replacement Delta Row (9)																	
Configure Raster Data		Format 1 (obslete)																	
		Format 2																	
		Format 3																	

		Format 4																	
		Format 5																	
Seed Row Source		Same plane of previous row (0)																	
		Previous plane (1)																	
		2nd previous (2)																	
		3rd previous (4-plane only) (3)																	
End Raster Graphics		~																	
Resolution		75 dots per inch																	
		100 dots per inch																	
		150 dots per inch																	
		200 dots per inch																	
		300 dots per inch																	
		600 dots per inch																	

Raster (continued)

Command	Code	Range																	
Raster Depletion	***Esc *o# D***	None (0)																	
		25% (2)																	
		50% (3)																	
		25% with gamma (4)																	
		50% with gamma (5)																	
Raster Quality	***Esc *r# Q***	Default (0)																	
		Draft (1)																	
		High (2)																	
Mechanical Print Quality	***Esc *o# Q* (same as below)**	Fast (-1)																	
		Normal (0)																	
		Better (1)																	
		Best (2)																	
Raster Shingling	***Esc *o# Q* (same as above)**	0% (0)																	
		50%, two pass (1)																	

		25%, four pass (2)																	
Raster Print Mode	***Esc *p# N***	Default (no break) (0)																	
		Bidirectional (1)																	
		Left to right (2)																	
		Right to left (3)																	
		Smart bidirection (4)																	
Raster Height (Destination)	***Esc *t# V***	0-32767 pixels																	
Raster Height (Source)	***Esc *r# T***	0-32767 pixels																	
Raster Width (Destination)	***Esc *t# H***	0-32767 pixels																	
Raster Width (Source)	***Esc *r# S***	0-32767 pixels																	

Raster (continued)

Command	**Code**	**Range**																	
Scale Algorithm	***Esc *t# K***	Enhance images with light background (0)																	

			Enhance images with dark background (1)																	
	Start Raster	***Esc *r# A***	At logical page left limit (0)																	
			At CAP (1)																	
			At logical page left limit, scaling ON (2)																	
			At CAP, scaling ON (3)																	
	Transfer Raster data (Plane)	***Esc *b# V***	0-32767																	
	Transfer Raster (Row/Block)	***Esc *b# W***	0-32767																	
	Raster Y Offset	***Esc *b# Y***	(-32767) to (32767) pixels																	

Raster Color

Command	Code	Range																	
Color Component (First)	***Esc *v#A***	0-32767																	
Color Component (Second)	***Esc *v#B***	0-32767																	
Color Component (Third)	***Esc *v#C***	0-32767																	
Color Index	***Esc *v#I***	0 to $(2^{\# \text{bits/index}})$-1																	
Color Lookup Tables	***Esc *l#W***	0 or 770																	
Configure Image Data	***Esc *v#W***	6,18																	
Download Dither Matrix	***Esc *m#W***	6-32767																	
Gamma Number	***Esc *t#I***	0.0-32767.0																	
Monochrome Print Mode	***Esc &b#M***	0,1																	
Palette ID	***Esc &p#I***	0-32767																	

Palette Configuration	***Esc *d#W***	9-32767																	
Palette Control	***Esc &p #C***	0,1,2,6																	
Push/Pop Palette	***Esc *p#P***	0,1																	
Render Algorithm	***Esc *t#J***	0-14 (0-12 for CLJ)																	
Select Palette	***Esc &p #S***	0-32767																	
Simple Color	***Esc *r#U***	KCMY palette (-4)																	
		CMY palette (-3)																	
		Mono palette (1)																	
		RGB palette (3)																	
Viewing Illuminant	***Esc *i#W***	8																	

Macros

Command	**Code**	**Range**																	
Macro ID	***Esc &f#Y***	0-32767																	
Macro Control	***Esc &f#X***	0-10																	

Print Model

Command	Code	Range																	
Download Pattern	***Esc *c# Wdata]***	0-(2^{31}-1)																	
Foreground Color (DJ5xx uses for text only)	***Esc *v# S***	0 to (2**bits/index)-1 (Current palette)																	
		White (0)																	
		True Black (1)																	
		Cyan (2)																	
		Magenta (4)																	
		Blue (6)																	
		Yellow (8)																	
		Green (10)																	
		Red (12)																	
		Composite Blk (14)																	
Logical Operation	***Esc *l# O***	0-255																	
Pattern Control	***Esc *c# Q***	0,1,2,4,5																	
Pattern ID	***Esc *c# G***	1-100 for Shading																	
		1-6 for cross-hatch																	

		0-32767 for user-defined																	

Print Model (continued)

Command	Code	Range																	
Pattern Reference Point	***Esc *p#R***	0,1																	
Pattern Type	***Esc *v#T***	Solid black or foreground color (0)																	
		White (1)																	
		HP shaded (2)																	
		HP cross-hatch (3)																	
		User pattern (4)																	
Pixel Placement	***Esc *l#R***	0,1																	
Fill Rectangle	***Esc *c#P***	Solid black or foreground color (0)																	
		White (1)																	
		HP shaded (2)																	
		HP cross-hatch (3)																	
		User pattern (4)																	

		Current pattern (5)																
Rectangle Size, Horiz (Decipoints)	***Esc *c# H***	0-32767																	
Rectangle Size, Horizontal (Dots)	***Esc *c# A***	0-32767																	
Rectangle Size, Vertical (Decipoints)	***Esc *c# V***	0-32767																	
Rectangle Size, Vertical (Dots)	***Esc *c# B***	0-32767																	
Transparency Mode (Pattern)	***Esc *v# O***	0,1																	
Transparency Mode (Source)	***Esc *v# N***	0,1																	

Status

Command	Code	Range																	
Display Functions Off	***Esc Z***	~																	
Display Functions On	***Esc Y***	~																	
Echo	***Esc *s# X***																		
Flush All Pages	***Esc &r# F***	0,1																	
Inquire Status Readback Entity	***Esc *s# U***																		
Free Memory Space	***Esc *s# M***	1																	
Self-test	***Esc z***	~																	
Status Readback Location Type	***Esc *s# T***																		
Status Readback Location Unit	***Esc *s# U***																		

Language Switching

Command	Code	Range																	
Comment (PJL)	@PJL COMMENT <text>																		
Enter Language (PJL)	@PJL ENTER LANGUAGE = (PCL | POSTSCRIPT | HPGL2)																		
Enter HP-GL/2 Mode	***Esc %# B***	Stand-alone plotter (-1)																	
		Use previous HP-GL/2 pen position (0)																	

		Use current PCL CAP (1)																	
		Use PCL coordinate system and old HP-GL/2 pen position (2)																	
		Use PCL coordinate system and current PCL CAP (3)																	
Enter PCL mode	***Esc %# A***	0,1																	
Exit Langua ge / Start PJL	***Esc %# X***	-12345																	

Picture Frame

Command	Code	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																	
HP-GL/2 Plot Horizontal Size	***Esc *c#K***	0-32767																	
HP-GL/2 Plot Vertical Size	***Esc *c#L***	0-32767																	

Control Codes

Command	**Code**	**Range**																	
Back Space	***BS***	~																	
Carriage Return	***CR***	~																	
Escape	***ESC***	~																	
Formfeed	***FF***	~																	
Horizontal Tab	***HT***	~																	
Line Feed	***LF***	~																	
Null	***NUL***	~																	

HP-GL/2

Command	Code	Range																	
HP-GL/2 Kernel																			
Bezier	***BR, BZ***																		
Fill Type	***FT***	Odd / Even																	
		Non-zero winding																	
Label Origin	***LO***	1-9 / 11-19																	
		21																	
Merge Control	***MC***																		
Color Range	***CR***																		
Number of Pens	***NP***																		
Pen Color	***PC***																		
Pixel Placement	***PP***																		
Screened Vectors	***SV***																		
Transparency Mode	***TR***																		
Primary Font Selection by ID	***FI***																		

Secondary Font Selection by ID	***FN***																		
Scalable or Bitmap Fonts	***SB***																		
Label Mode	***LM***																		

NOTE: Except for not allowing font downloads, OfficeJet is the same as the DeskJet 520.

GLOSSARY

Absolute Color: Absolute colors are specified according to NTSC (National Television System Committee) red, green, and blue primaries with illuminant D6500. (see relative color)

Absolute Movement: Motion specified with respect to the coordinate system origin (0,0). The PCL language contains both relative and absolute CAP movement commands. *Relative* motion references the present CAP. *Absolute* motion references (0,0), the intersection of the logical page left edge and the top margin. Absolute moves are specified with unsigned value fields; relative moves are specified with signed value fields.

Addressability: The ability of a device to position a dot. This is different than *resolution*, which refers to the number of dots that can be placed horizontally or vertically in a square inch (dots per inch). For example, a device with 300 dpi resolution may have 600 dpi addressabilty.

Algorithmic Bolding: A method of bolding characters which is not involved in the font selection process.

Anchor Point: The upper left corner of the picture frame, which is set to the current active position (CAP) in the PCL environment at the time the picture frame anchor point command is executed.

Anisotropic Mapping: A mapping in which the units along each axis are not equal. This term is often used to describe the mapping from user units to plotter units.

ASCII: The acronym for ***A***merican ***S***tandard ***C***ode for ***I***nformation ***I***nterchange. ASCII is a standard character set mapping for character sets addressable by 7 bits.

Aspect Ratio: The ratio of the height to the width (y to x) of a rectangular area, such as a window.

Auto Macro Overlay: The macro that is executed as the final operation each time a page is printed. The overlay environment is a combination of the user default and the current modified print environments. The modified print environment is saved so it can be restored following completion of the macro overlay.

Baseline: An imaginary line to which most of the characters of a font are aligned.

Baud Rate: The rate at which information is transferred between two devices. To communicate properly, both devices must be configured to the same baud rate.

Big Endian: This byte-ordering method stores the most significant byte in the lowest address. This method is used by Motorola. See *Little Endian*.

Bit: The binary 1's and 0's, i.e., the data that is downloaded to the printer to specify dots or pixels.

Bitmap Character: A character on a dot matrix, consisting of a pattern of dots.

Bitmap Font: A font whose character size (height and pitch) is fixed (static). A static bitmap font is a raster representation of characters. Each character is described in terms of pixels.

Bitmap: A region of memory which is treated as a rectangular array of pixels. Same as raster.

Black Reference: In direct color specification, the black reference represents the minimum output of a primary that a device can produce. See *White Reference*.

Bottom Margin: Distance from the bottom of the text area to the bottom of the logical page. The bottom margin is set indirectly by setting text length when perforation skip mode is enabled.

Bound Font: A bound font is one that is restricted to a single symbol set. See *Unbound Font*.

Cap Height: A percentage of the em of a font used to calculate the distance from the capline to the baseline. For fonts containing no letters (such as collections of clip-art graphic objects), the cap height percentage is 70.87%.

CAP: See *Current Active Position*.

Capline: An imaginary line corresponding to the top dot-row of an unaccented upper-case letter, usually, "H".

Ceiling Function: A mathematical function which yields the least integer greater than or equal to a given real number. For example, a value of 2.1 yields a value of 3 after applying the ceiling function.

Character Cell: Defines the area that holds the character and its required components.

Character Code: An 8-bit binary code that identifies a downloaded character.

Character Definition: Contains all the information necessary to download an individual character. The character definition consists of the *character header*, the *character descriptor*, and the binary data that specifies the character's shape—raster data for bitmap fonts or outline data for scalable fonts.

Character Descriptor: A block of data following the *character header*. It describes the position and size of an individual character, as well as the implied movement after printing the character.

Character Downloading: When downloading a font to the printer, the first stage is the downloading of the font definition. The second stage is the downloading of the individual characters in the font.

Character Header: The first two byte fields, Format and Continuation, of every character definition. The Format field describes the type of character to follow (e.g., bitmap, Intellifont, TrueType, etc). The Continuation field indicates previous blocks in the character definition: zero means none

Character ID: See Character Code.

Character List: An unordered group of characters used to construct symbol sets.

Character Motion Index (CMI): In horizontal text path direction, CMI designates the width of columns used for inter-character movement; in vertical text path direction, CMI designates the height of rows used for inter-character movement.

Character Set: See Symbol Set.

Checksum: A calculation performed on the bytes in a font or character definition to ensure downloading accuracy.

Chord Angle: The angle in degrees through which a chord is drawn. The chord angle determines the number of chords (and thus the smoothness) used to draw a circle or an arc.

Chord: A straight line joining two points on an arc or on the circumference of a circle.

CID: Configure Image Data command (*Esc*v#W*)

Clean Page: A page in which no print command (including a printable space) and no command affecting CAP has been sent.

Clip: The operation of removing any portion of a graphical image which extends beyond a specified boundary.

CMI: See *Character Motion Index*.

CMY: The subtractive primary colors used for printing: Cyan, Magenta, and Yellow.

CMYK: The subtractive primary colors—Cyan, Magenta, and Yellow—plus Black.

Color Depletion: See *Depletion*.

Color Matching: Matching printed colors to other devices, especially a monitor. This may go beyond merely making colors the same; it may involve color modification to satisfy the user's perception, i.e., "color appearance matching".

Color Mode: The PCL language offers four color modes: black and white (default), simple, PCL imaging, and HP-GL/2 imaging.

Color Palette: The set of colors in a device that are currently available for selection.

Color Primaries (Components): The three basic colors in some color models (e.g. RGB or CMY) from which all other colors can be produced. A mixture of any two cannot produce the third.

Color Saturation: See *Saturation*.

Color Space: The color model that is used to specify colors. Examples are: Device RGB, Device CMY, CIE L*a*b*, etc.

Column: Columns are PCL coordinate system units measured along the x-axis of the PCL logical page. The width of a column is defined by the current pitch or horizontal motion index (HMI). HMI is the distance between consecutive fixed-pitch characters. For example, at 12 cpi, one column is 1/12 of an inch, and at 10 cpi, one column is 1/10 of an inch. In a proportionally spaced font, one column is the width of the font's space character.

Compaction: See *Compression*.

Compression: An encoding scheme wherein a single datum represents multiple pieces of the original information thereby reducing the size of the data necessary to represent the original information.

Conditional Formfeed: A conditional page eject that occurs if the page is dirty, i.e., if printable data or a command affecting CAP has been sent. Orientation and Page Size are examples of commands that perform conditional page ejects.

Continuation Block: Used for downloading character definitions whose byte count exceeds 32767.

Control Character: See *Control Code*.

Control Code: A character that typically initiates a device function. For example CR (ASCII decimal value 13), LF (ASCII decimal value 10), and FF (ASCII decimal value 12).

CPI: Characters per inch.

CRD: Configure Raster Data command (*Esc*g#W*).

Current Active Position: Also CAP. The currently active printing position. The position at which the next character will be printed. CAP moves one column right after printing a character and one dot row down after printing graphics. CAP can also be moved explicitly anywhere within the logical page by using PCL control codes and escape sequences. After power on or reset, until printable data or a command affecting CAP is received, CAP is *floating*, which means it moves in accordance with orientation, top margin, left margin, and line spacing commands. However, once a printable character or a command affecting CAP is received, CAP is *fixed*, which means it is not affected by changes to the orientation, top margin, left margin, or line spacing.

Current Unit: In HP-GL/2, if the HP-GL/2 plot size is the same as the picture frame size and no user scaling is specified (IW), current units are plotter units. If the HP-GL/2 plot size is different than the picture frame size but user scaling is not in effect, units are PCL picture frame units; if user scaling is specified, units are user units.

Cursive: Type style that resembles handwriting but has no connecting characters as for example found in script faces. It can be written by pencil, brush, or other writing tools. The cursive style is used for typesetting systems that cannot connect characters.

Cursor: See *Current Active Position*.

Decipoint: A unit of measurement that equals 1/10 of a point or 1/720th of an inch.

Decompression: Decompression is a decoding scheme that expands compressed data into its original form.

Default: A value used in lieu of a programmatically selected value. A factory default is a value programmed into the device at the factory; this value is stored in read-only memory (ROM) and cannot be changed by the user or operator. A user default is a default that is selectable via a control panel.

Delta X: HP's term for *escapement*.

Depletion: An ink jet technique that selectively removes pixels from a graphics image to reduce ink placed on the page. Depletion may be used to selectively reduce certain color intensities to achieve a more linear color intensity curve.

Destination Raster: In raster scaling, this is the size of the raster area that will actually be printed, as opposed to the raster size specified by the data that is sent. See *Source Raster*.

Destination: The composite of all primitives currently rendered on the page. Whenever a new page is selected, the destination is set to white.

Device Best: The rendering that HP believes will provide the best output for a particular device.

Device CMY: Device-dependent color space based upon Cyan, Magenta, and Yellow primaries.

Device Coordinates: See HP-GL/2 Coordinate System Units and PCL Coordinate System Units.

Device RGB: Device-dependent color space based upon Red, Green, and Blue primaries.

Device-Dependent Color: Color specified relative to a device's native rendering mode. Each device receiving a device-dependent color specification will produce a different color.

Device-Independent Color: Based upon the tristimulus values of human vision. Such a specification is translated into a device's native space in such a way that the resultant is independent of the device.

Direct Color: Colors are selected by specifying the values of each primary composing a pixel. See *indexed color*.

Dirty Page: A page which has received a print command or a command affecting CAP. CAP is then fixed and any command (such as Orientation or Page Size) that causes a *conditional* page eject will eject the current page and move CAP to the intersection of the left margin and top of form on the next page.

Dither: In this halftoning technique, colored dots are arranged in a matrix pattern that the eye integrates to produce colors other than those available in the printer's default palette. Spatial resolution is traded for color resolution.

Dot: The smallest mark the printer can make. Its size and spacing are device specific (e.g., on the LaserJet III one dot equals 1/300 inch). The number of dots printed per inch is referred to as the printer's *resolution*. The ability of a device to position a dot is its *addressability*, which may be different.

Download: The process of transferring entities such as fonts, characters, symbol sets, macros, graphics, or other data from a host computer to the device's user memory.

Dual Fixed Spacing: In Asian printing, a font may consist of two different spacings, one for Asian characters and one for Latin characters, which are usually half the width of Asian characters.

Edge: The outline of a polygon.

Em: A square unit of typographic measure whose size is equal to the height of the font in decipoints.

En: A unit of typographic measure whose size is equal to one half an em, or half the point size of the font. Approximately equal to the width of the letter "n."

Error Diffusion: A halftoning technique in a pixel is printed at the closest color value that the printer can produce, and the local error is propagated to unprinted neighboring pixels.

Escape Character: ASCII decimal value 27 (identified in this document by "Esc"), which is always the first character of a PCL escape sequence, identifies a character string as a command, not as data to be printed.

Escape Sequence: A combination of characters preceded by ASCII decimal value 27 that represents a device command. When this character appears, the device reads it and its associated characters as a command to be performed and not as data to be printed. (See PCL Commands.)

Escapement: The distance CAP moves after printing a character. HP's term is *Delta X*.

Factory Default Environment: The group of all of the device's factory default configuration settings.

Factory Default: Factory default refers to configuration values that are programmed into a device at the factory. These values or defaults are in use unless they are overridden using either the control panel or PCL commands.

Fill Type: The shading pattern used to fill a polygon.

Fixed Spaced Font: A font whose characters all take up the same amount of space.

Flash: A type of ROM that can be erased and rewritten.

Font Definition: Includes the font descriptor and any subsequent data segments.

Font Descriptor: A block of data which describes font design characteristics. The font descriptor consists of fields defining the height, width, style, typeface, and character set of the font. Although some devices do not use all the data in the font descriptor, a font creator should use valid values in all the font descriptor fields to insure font compatibility in all devices.

Font Downloading: The process of loading a font from the host into the printer. First the font definition is loaded, and then each character is loaded.

Font Header: Another name for *Font Descriptor*.

Font ID: A unique identification number which should be designated prior to downloading a font descriptor. The font ID is used to identify and reference the font. If an existing font is already associated with this ID, the existing font will be deleted during the download of the font descriptor.

Font Operator: A font operation, such as algorithmic bolding or orientation, that is independent of the font selection process.

Font: A font is a set of printable images specified by a collection of attributes. A font has an assigned name and is further described by its attributes**:** typeface, spacing, height, pitch, posture, stroke weight, character set, and orientation. If one attribute in the set is changed, the font becomes a different font.

Full-Width Character: A character whose escapement equals the font height. For TrueType, a full-width character's escapement in design units equals the font's scale factor.

Glyph: A graphical symbol.

Graphical Image: A graphical image is any item affecting the appearance of a page. Examples include printed characters, rules, area-fill patterns, raster graphics, and vector graphics.

H1, H2: H1 and H2 are the lower left and upper right corners of the Hard-Clip Limits respectively.

Hard-Clip Limits: The hard clip limits are the boundaries of the printing area beyond which the device cannot print, i.e., the mechanical limits of the device. Same as the printable area.

Hatch: Hatch is a particular fill type made of parallel lines.

Header: For downloaded fonts, this is another name for the font *descriptor*, which is the first part of the font definition, excluding data segments; it defines the characteristics common to all the characters in the font. For downloaded characters, the header is the first two byte fields (Format and Continuation) in every character definition.

Height: The height of a font is the measurement of the body of the type in points. A PCL point is 1/72th inch.

Hifipe: A pixel encoding method in which the bits composing a pixel correspond to intensities as well as colors.

Hiragana: Cursive form of Japanese characters representing syllables. See also *Katakana*.

HMI: Obsolete. See *Character Motion Index (CMI).*

Horizontal Motion Index (HMI): Obsolete. See *Character Motion Index (CMI).*

HP-GL/2 Coordinate System Units: The units of the HP-GL/2 coordinate system are the units of the x and y axes of the HP-GL/2 coordinate system. These units may be plotter or user. See *Current Unit*.

HP-GL/2 Mode: HP-GL/2 mode is the state during which a PCL device interprets the incoming data stream as HP-GL/2 commands instead of PCL commands.

HP-GL/2: HP-GL/2 is an acronym for *H*ewlett *P*ackard *G*raphics *L*anguage. HP-GL/2 is the command set that HP's graphic devices understand; it defines a standard for printers' and plotters' vector graphics features.

Illuminant: The light source used when measuring or viewing color.

Indexed Color: Indexed color is a method of specifying color wherein a pixel is represented by an index into the programmable color palette.

Ink Jet Printing: A printing process that uses fine jets of ink to reproduce a digital, computer-generated image on paper.

Intellifont: Compugraphic's scalable typeface data system capable of providing bitmap font data on any resolution device and at any point size.

Internal Unit: Internal units are the units used within the device (i.e., they are not user accessible). The device maps all other units (e.g., dots, decipoints, plotter units, etc.) to this unit of measure. All positioning is kept in internal units and converted to physical dot positions when printed.

ISO Character Set: ISO character sets are 7-bit sets containing European versions of the Roman alphabet (e.g., ISO-German contains umlaut vowels, ISO-French contains e accent grave, etc.) based on the standards produced by the *I*nternational *S*tandards *O*rganization (ISO).

Isotropic Mapping: A mapping in which the units along each axis are equal. This term is often used to describe the mapping from user units to device units.

Join: The point where two line segments connect.

Kanji: The ideographic characters normally used for the Japanese language. Derived from Chinese writing. A Japanese printer commonly uses around 5,500 characters; it requires up to 22,000 different characters for high quality work.

Katakana: One of two sets of japanese characters representing syllables used to write words for which there are no appropriate Kanji characters. Non-cursive form. See also *Hiragana*.

Kerning: The extension of a character's shape beyond its positioning and escapement points.

Label Mode: The HP-GL/2 mechanism for printing text.

Label Terminator: A label terminator is the final character in an HP-GL/2 label string. This character terminates label mode, so that the device interprets subsequent characters as HP-GL/2 instructions.

Landscape: A rectangular image in which the width is noticeably greater than the height. See *Orientation*.

Laser Printing: A printing process that utilizes a laser to write the image on a drum. Toner particles adhere to the uncharged places on the rotating drum and are transferred from the drum to the paper by a fusing process.

Left Raster (Graphics) Margin: The left raster margin, which may be different than the left text margin, starts at either CAP or the left edge of the logical page.

Left Margin: Distance between the left edge of the logical page and the left edge of the text area. The term "left margin" may also refer to left edge of the text area.

Line Motion Index (LMI): In horizontal text path mode, LMI sets the vertical spacing between lines of print (the vertical distance CAP will move for a linefeed). In vertical text path mode, LMI sets the horizontal distance CAP will move for a linefeed.

Little Endian: This byte ordering method stores the least significant byte at the lowest address. This technique is used by most manufactures other than Motorola. See *Big Endian*.

LMI: See *Line Motion Index*.

Logical Page Coordinate System: The PCL coordinate system used for PCL cursor movement and placement of graphical images. The logical page coordinate system can be changed by defining a new logical page or by changing the print direction. The logical page coordinate system is defined by the width and height of the logical page. Points on the x-axis (whose origin is at the left edge of the logical page) increase in value across the width of the logical page; points on the y-axis (whose origin is at the top margin) increase in value down the length of the logical page.

Logical Page: The PCL logical page (also referred to as the addressable area) defines the area in which the cursor can be positioned. Logical page size may be changed by PCL commands; and it can be larger than the printable area, which is determined by the technology of the printing device.

LPI: Lines per inch.

Macro: A group of user-created PCL commands and data that can be executed repeatedly by a single command, using an assigned macro ID.

Mask: A single-bit-per-pixel primitive which acts as a stencil; the pattern is pushed through the 1's bits in the mask to form the source.

Mode: In the PCL language, a feature that is defined as a mode remains in effect until an *EscE* reset. Most PCL commands act like modes.

Modified Print Environment: This collection of all the current feature settings is stored in RAM and may be modified by printer language commands.

Munsell Value: A unit of brightness defined in the Munsell Color System with the property that equal changes in value corresponds to equal changes in perceived brightness. For hardcopy, nearly all colors are contained in the value range of 0 to 10. Black and white have Munsell values of approximately 2.5 and 9.5, respectively.

Non-Raster Color: Color that is specified in non-raster mode. The palette must be used.

NTSC: The acronym for *N*ational *T*elevision *S*ystem *Co*mmittee. The NTSC developed the standard color model that applies to all government-regulated broadcast systems in the United States.

Orientation: Specifies the position of the logical page on the physical page. Assume, for example, that the physical page is a rectangular sheet of paper with one of the short sides up. Then, *portrait* orientation means the logical page origin (0,0) is toward the top left corner of the physical page, with the X direction to the right and the Y direction downward. Printing is across the "width" of the page. *Landscape* orientation means the origin is rotated counterclockwise by 90 degrees toward the lower left corner of the physical page. Then, the X direction is upward and the Y direction is to the right. Printing is across the "length" of the physical page.

Overlay Environment: A collection of current PCL variables and user-default variables that define the current print environment before the macro overlay is executed.

P1, P2: The points that take on the user unit values specified with the HP-GL/2 scaling instruction, SC. P1 and P2 always represent an absolute plotter-unit location and define opposite corners of a rectangular printing area.

Page: One side of a sheet of paper or other printable media.

Palette ID: A unique identification number assigned to a palette for management purposes.

Palette: See *Color Palette*.

Pantone: A registered trademark name for a system of color matching in designer’s and printer’s materials such a inks, papers, marker pens, and overlay films. Each color is assigned a specific PMS reference color coding number.

Pattern ID: A unique identification number for user-defined and HP-defined patterns.

Pattern: The pattern is combined with a mask to form a source. The pattern is either a combination of the current foreground color with the selected single-bit-per-pixel texture, or a multi-bit-per-pixel user-defined area fill material (the multi-bit pattern contains the color and texture).

PCL Commands: PCL commands provide access to printer features. Once a PCL command sets a parameter, that parameter remains set until the same PCL command is repeated with a new value or the printer is reset. There are three types of PCL commands**:** control codes, two-character escape sequences, and parameterized escape sequences. (See *Control Code* and *Escape Sequence*.)

PCL Coordinate System Units: The PCL coordinate system units are the units of the x and y axes of the PCL coordinate system. These units may be dots, decipoints, or, columns and rows.

PCL Printer Language: The PCL language is HP’s standard language for printer control. The PCL language defines a standard for printer features and feature access by software applications. It provides the highest level of communication between the system and the printer. PCL is designed to be independent of the host system, device drivers, I/O interface, and network communications. Its purpose is to bring together all HP printers under a consistent control structure that provides feature compatibility from printer to printer.

PCL Unit: A user-definable (Unit of Measure command) unit that may used for CAP moves or rule sizes. The factory default is 1/300". PCL Units were formerly referred to a "dots", but were renamed to prevent confusing with a device’s physically printed dots, which are determined by *resolution*.

Pen Position: See *Current Active Position*.

Pen Status: In HP-GL/2, the pen status indicates the (up/down) state of the currently selected pen.

Perforation Region: The distance from the bottom of the text area of one page to the top of the text area of the next page. Enabling (default) perforation skip prints text to print to the end of the text area of the current page and start again at the text are on the next page. Disabling perforation skip prints text to the bottom physical edge of the page, into the unprintable region, and onto the top physical edge of the next page. Text length and margins are ignored, and data may be lost. The remainder of a partial line is not carried over to the next page. This term is used because of the need on many continuous feed devices to avoid printing on an area that may be perforated to aid page separation after printing.

Permanent (font, pattern, palette, macro): A permanent entity is not deleted by an *EscE* reset. It is deleted after a UEL only if a personality other than PCL is entered. It is device-specific whether it is deleted by a control panel reset.

Physical Device Units: See *HP-GL/2 Coordinate System Units* and *PCL Coordinate System Units*.

Physical Page: The actual sheet of paper or media—what the user thinks of as a "page". Physical page size is determined by the size of the installed media.

Pitch: The number of characters printed in a horizontal inch. Pitch applies only to fixed-spaced fonts, since the number of characters per inch varies for proportionally-spaced fonts.

Pixel: The acronym for ***pic***ture ***el***ement. A pixel is the smallest definable picture element within a bitmap that can be assigned color or intensity. A pixel may consist of one or more dots. In the PCL language a raster image is divided into pixel rows which describe a strip of the image one pixel high.

Plane: A method of organizing raster data in which all the pixels in a row each receive one bit, then they all receive the next bit, etc., until each pixel in the row is completely defined. The corresponding bits for each pixel compose a *plane*. Another method of raster encoding is to send all the bits for each pixel, then send all the bits for the next pixel, etc. This method is called *pixel encoding*.

Plot Size: The plot size defines the dimensions of the area in which an HP-GL plot was originally rendered. When an HP-GL/2 plot is imported into the PCL environment, the plot is scaled to fit within the rectangle known as the picture frame.

Plotter Unit: Plotter units are the default HP-GL/2 units. Each plotter unit is 0.025 mm (1/1016 inch). There are 40 plotter units per millimeter.

Point: A PCL point is a typographical unit of measurement that equals 1/72th inch. Font height is measured in points.

Polygon: A polygon is a closed figure of arbitrary shape.

Portrait: Orientation of a page with the longest dimension in the vertical direction. See *Orientation*.

Primary (Secondary) Font: The primary font is the font whose attributes most closely correspond to the designated primary font attributes that is accessed via the control code SI (ASCII decimal value 15). The secondary font is the font whose attributes most closely correspond to the designated secondary font attributes that is accessed via the control code SO (ASCII decimal value 14). Two fonts can therefore be designated simultaneously and selected by alternating between them using the control codes SI and SO.

Primitive: A graphical item which is used in the construction of a page. Primitives include characters, rules, raster, underlines, and HP-GL/2 primitives (vectors, polygons, circles, etc.).

Print Direction: See *Logical Page Coordinate System*.

Print Environment: The group of all of the device's current feature settings. The device maintains four print environments**:** the factory default environment, the user default environment, the modified print environment, and the overlay environment.

Print Mode: Refers to user-selectable or automatic device-chosen print quality modes that are determined by media type, throughput, resolution, etc.

Print Overrun: As data is received by the printer, it is processed and stored in an intermediate format. The intermediate representation of the page is later processed and printed. During the printing of the page, the paper moves through an asynchronous printer (e.g., a laser printer) at a constant speed. A print overrun occurs when a page cannot be printed because the page's intermediate data cannot be processed and printed fast enough to keep up with the page as it moves through the printer.

Printable Area: The area of the physical page in which the printer is able to place a dot. The physical page refers to the size of the media installed in the device (i.e., same as the hard-clip limits). The printable area is usually determined by the technology of the printing device. Note that the text area and logical page may be larger than the printable area. Attempting to print characters outside the printable area results in data loss.

Printer Commands: See *PCL Commands*.

Proportionally Spaced Font: See *Spacing*.

Radix Number: A number which has an assumed radix point between bits 1 and 2 to provide 1/4 units of measure.

Raster Area: Raster margins, which may be different than text margins, are set explicitly or simply clipped at the logical page or printable area boundaries.

Raster Color: Raster color may be specified *directly* or using a palette *index*. A palette must always be used to specify non-raster color.

Raster Mode: A restricted state in which only certain raster commands are allowed; non-raster text commands may end raster mode. Raster margins may differ from text margins.

Raster Scaling: A raster graphic may be scaled if the Start Raster command (*Esc*r#A*) is sent with a value of two or three. The scale factor depends upon the size of the raster area that is sent (*Source Raster*), the size of the raster area actually printed (*Destination Raster*) and the device resolution.

Raster: A raster image is a picture composed of groups of pixels. Pictures in newspapers or on televisions are examples of raster images.

Relative Color: Color specified in a space defined by user-defined white and black reference values for each of the three primaries.

Relative Movement: Movement specified relative to the current active position.

Rendering: Rendering is the process of combining a source with the destination as directed by the selected drawing mode.

Resolution: The resolution of a raster device is a measurement of the maximum number of dots that can be printed in a horizontal inch by the maximum number of dots in a vertical inch.

RGB: Red, green, and blue. The primary, additive colors of light which can be mixed to form all other colors.

Right Margin: The distance between the left edge of the logical page and the right edge of the text area. The term "right margin" may also refer to right edge of the text area.

ROP: Logical operation (AND, OR, etc.) used in the print model process.

Rounding Function: The rounding function is a mathematical function which yields the closest integer to the given real number. For example, 3.49 yields a value of 3 and 3.5 yields a value of 4 when rounded.

Row: Rows are PCL coordinate system units measured along the y-axis of the PCL logical page. The distance between rows is defined by the current line spacing (lpi) or vertical motion index (VMI). VMI is the distance between consecutive lines of text. For example, at 6 lpi, one line is 1/6 of an inch, and at 8 lpi, one line is 1/8 of an inch.

Rule: Also Rectangular Area Fill. In the PCL language, rectangular area fills are a special case of source images. which may be filled by patterns or textures. Source Transparency Mode has no effect on rules, since the rectangular area is conceptually viewed as a solid black (all 1's) source.

Saturated Color: A pure color without any white or black. One of the printer's basic colors, without any halftoning or mixing.

Scalable Font: A font whose character size (height and pitch) can be altered (scaled). Scalable fonts are described in terms of the outlines of the characters. The outlines are represented using vectors, arcs and curves, splines, etc. Scaling is achieved by applying linear transformations to the outlines. Additional information describing the relationships between character shapes and contours may be stored with a font; this information is applied during the scaling process to insure the integrity of the font design.

Secondary Font: See *Primary Font*.

Sheet: A physical piece of paper or other printable media.

Shingling: An ink jet interface technique that uses checkerboard masking to remove banding effects.

Soft Clip Limits: The boundaries of the printing area defined by the HP-GL/2 IW command, beyond which no programmed printing can occur (also referred to as a window).

Soft Font: A font which can be downloaded from a computer to the device's memory .

Source Raster: In raster scaling, this is the size of the raster area specified by the data that is sent, as opposed to the size of the raster area that is actually printed. See *Destination Raster*.

Spacing: Fonts have either fixed or proportional spacing. Fixed-spaced fonts are those for which the inter-character spacing is constant. Proportionally-spaced fonts are those for which the inter-character spacing varies with the natural shape of a character.

Stack: Last-in-first-out (LIFO) data structure. The PCL language has commands which can push (store) or pop (restore) CAP or the color palette.

Static Bitmap Font: See *Bitmap Font*.

Stroke Weight: The thickness of the strokes that compose characters. Medium and bold are examples of stroke weights.

Style: Font style defines the angularity of the strokes of the character with respect to the x-axis. Examples of font styles are "upright" and "italics".

Symbol Set: A unique ordering of the characters in a font. Each character set is defined with a unique set of applications in mind. Character sets are created for many purposes. For example, the PC-8 (US) character set was designed to support US IBM-PC applications.

Symbol Set Downloading: For *bound* fonts, symbols sets are part of the font specification. For *unbound* fonts, symbol sets may be downloaded separately.

Temporary (font, pattern, palette, macro): A temporary entity is deleted by an *EscE* reset.

Text Area: The area between the four margins. The text area is entirely contained by the logical page; it may be the same size or smaller than the logical page. Text printing may be restricted to a specific area within the logical page using the left margin, right margin, top margin, text length and perforation skip mode commands.

Text Length: Distance from the top margin to the bottom of the text area. Text length has meaning only if perforation skip mode is enabled.

Texture: A single-bit-per-pixel element that is combined with the current foreground color to form the pattern.

Tiling: The process of replicating a selected pattern beginning at a specified point (e.g., the upper left corner of the PCL logical page) to cover the entire page. A mask is then positioned and the tiled pattern passes through the 1's bits in the mask to compose the source used in rendering.

Top Margin: Distance between the top of the logical page and the top of the text area. The term "top margin" may also refer to the top edge of the text area.

Top of Form: After power on or reset, CAP moves to the top of form at the left margin. Top of form is 3/4 of a line below the top margin, or top margin + (3/4 * line spacing). The phrase "move to top of form" means if the CAP is not at top of form, move it to top of form of the next logical page.

Tristimulus Values: Three primary colors (X, Y, Z) that can be combined, with positive weights, to define all sensations we experience with our eyes. Defined in 1931 by the Commission Internationale L'Eclairage (CIE).

TrueType: A font scaling technology like Intellifont, but from a different vendor.

Typeface: A generic name for graphics symbols having common design features. Each typeface has unique and distinguishing characteristics.

UEL: Universal Exit Language command (*Esc-12345X*), which exits the PCL language and turns over control to the PJL language.

Unbound Font: A font which is not restricted to a single symbol set. Bound fonts never contain more than 256 accessible characters, while unbound fonts may contain 400 or more of the symbols in Unicode or HP's Master Symbol List (MSL).

Unconditional Formfeed: A page eject that occurs whether or not printable data or a command affecting CAP has been sent (i.e., whether or not the current page is *dirty* or *clean*. The control code <FF> causes an unconditional page eject.

User Default Environment: Contains the factory default settings modified at the control panel or by the PJL DEFAULT command.

User Default: A feature setting that has been selected at the control panel.

User Units: HP-GL/2 coordinate system units that are defined by the scale instruction SC and the positions of P1 and P2.

Vertical Motion Index (VMI): Obsolete. See *Line Motion Index (LMI).*

VMI: Obsolete. See *Line Motion Index (LMI).*

VRC: Vector to Raster Converter.

White Reference: In direct color specification, the white reference represents the maximum output of a primary that a device can produce. See *Black Reference*.

Window: In HP-GL/2, the graphics window is the rectangle on the physical page within which graphics marks may be made. This window is the intersection of three rectangles**:** the rectangle defined by the hard-clip limits (printable area), the window defined by the current PCL picture frame, and the soft-clip or user window defined by the IW command.

Index by Command

Disable Underline	*Esc&d@*	5-12
Display Functions Mode OFF	*EscZ*	5-11
Display Functions Mode ON	*EscY*	5-11
Download Character	*Esc(s#W*	11-4
Download Dither Matrix	*Esc*m#W*	14-44
Download Font	*Esc)s#W*	10-4
Download Pattern	*Esc*c#W*	16-18
Download Symbol Set	*Esc(f#W*	12-6
Driver Function Configuration	*Esc*o#W*	6-17
Dry Timer	*Esc&b#T*	6-13
Duplex Page Side	*Esc&a#G*	6-4
Echo	*Esc*s#X*	19-12
Enable Underline	*Esc&d#D*	5-12
End Raster	*Esc*rC*	13-6
End Raster Graphics	*Esc*rB*	20-14
End-of-Line Wrap	*Esc&s#C*	5-13
Enter HP-GL/2 Mode	*Esc%#B*	17-2
Enter Language	*@PJL ENTER LANGUAGE*	2-6
Enter PCL Mode	*Esc%#A*	17-5
Escape	*<ESC>*	5-4
Escapement Encapsulated Text	*Esc&p#W*	5-14
Fill Rectangular Area	*Esc*c#P*	16-23
Finish Mode	*Esc*o#A*	14-51
Flush All Pages	*Esc&r#F*	19-11
Font Control	*Esc*c#F*	10-39
Font Height (Primary)	*Esc(s#V*	9-19
Font Height (Secondary)	*Esc)s#V*	9-19
Font ID	*Esc*c#D*	10-3
Font Pitch (Primary)	*Esc(s#H*	9-18
Font Pitch (Secondary)	*Esc)s#H*	9-18
Font Quality (Primary)	*Esc(s#Q*	20-25
Font Quality (Secondary)	*Esc)s#Q*	20-25
Font Selection by ID (Primary)	*Esc(#X*	9-43
Font Selection by ID (Secondary)	*Esc)#X*	9-43
Font Spacing (Primary)	*Esc(s#P*	9-17
Font Spacing (Secondary)	*Esc)s#P*	9-17
Font Stroke Weight (Primary)	*Esc(#B*	9-22
Font Stroke Weight (Secondary)	*Esc)#B*	9-22
Font Style (Primary)	*Esc(s#S*	9-20
Font Style (Secondary)	*Esc)s#S*	9-20
Font Symbol Set (Primary)	*Esc(ID*	9-8
Font Symbol Set (Secondary)	*Esc)ID*	9-8
Font Typeface (Primary)	*Esc(s#T*	9-23
Font Typeface (Secondary)	*Esc)s#T*	9-23
Foreground Color	*Esc*v#S*	14-41
Formfeed	*<FF>*	5-5
Free Space	*Esc*s#M*	19-10
Gamma Correction	*Esc*t#I*	14-49
GPIS Data Binding	*Esc*t#G*	20-8
GPIS Data Transfer	*Esc*t#W*	20-11
Gray Balance	*Esc*b#B*	6-13
Half Linefeed	*Esc=*	20-3
Head View Enable/Disable	*Esc&k#V*	20-28
Horizontal Picture Dimension	*Esc*t#H*	20-8

Picture Frame Anchor Point	Esc*c#T	17-9
Picture Frame Size (Horizontal)	Esc*c#X	17-10
Picture Frame Size (Vertical)	Esc*c#Y	17-10
Pixel Placement	Esc*l#R	16-14
Plot Size (Horizontal)	Esc*c#K	17-15
Plot Size (Vertical)	Esc*c#L	17-15
Primary and Secondary Pitch Mode	Esc&k#S	20-24
Primary Font	<SI>	9-7
Primary Font Placement	Esc(s#U	20-25
Primary Status Request	Esc^	20-27
Print Control	Esc*t#P	20-9
Print Direction	Esc&a#P	7-12
Print Enhancement (Underline)	Esc&k#E	20-25
Print Mode (Graphics)	Esc*p#N	20-9
Print Mode (Text)	Esc*p#N	20-10
Print Quality	Esc*o#M	6-14
Print with Paper Motion via VFC	Esc&l#V	20-5
Push/Pop CAP	Esc&f#S	8-11
Push/Pop Palette	Esc*p#P	14-37
QMS Magnum-V Enable/Disable	Esc*t#F	20-27
Raster Graphics Quality	Esc*r#Q	20-10
Raster Presentation	Esc*r#F	13-7
Raster Resolution	Esc*t#R	13-6
Raster Y Offset	Esc*b#Y	13-9
Render Algorithm	Esc*t#J	14-42
Reset	EscE	17-6
Reset	EscE	6-2
Return Model Number	Esc*r#K	20-27
Return Model Number	Esc*s#^	20-27
Right Margin	Esc&a#M	7-20
Scale Algorithm	Esc*t#K	13-23
Secondary Font	<SO>	9-7
Secondary Font Placement	Esc)s#U	20-25
Seed Row Source	Esc*b#S	13-21
Select Palette	Esc&p#S	14-39
Self Test	Escz	6-12
Set Horizontal Tab	Esc&a#T	20-3
Set Horizontal Tab at Current Column	Esc1	20-3
Set Left Margin at Current Horiz Position	Esc4	20-26
Set Right Margin at Current Horiz Position	Esc5	20-26
Set Vertical Tab (Absolute)	Esc&l#Y	20-4
Set Vertical Tab (Relative)	Esc&l1M	20-4
Shift In	<SI>	5-6
Shift Out	<SO>	5-6
Shift-In/Shift-Out Control	Esc&k#F	20-25
Simple Color	Esc*r#U	14-13
Simplex/Duplex	Esc&l#S	6-4
Source Raster Height	Esc*r#T	13-9
Source Raster Width	Esc*r#S	13-8
Space	<SP>	5-6
Start Raster	Esc*r#A	13-5
Symbol Set Code	Esc*c#R	12-5
Symbol Set Control	Esc*c#S	12-11

Text Color	*Esc&v#S*	20-14
Text Length	*Esc&l#F*	7-21
Text Parsing Method	*Esc&t#P*	5-17
Text Path Direction	*Esc&c#T*	7-16
Top Margin	*Esc&l#E*	7-21
Top Registration	*Esc&l#Z*	6-6
Transfer Raster by Plane	*Esc*b#V*	13-10
Transfer Raster by Row/Block	*Esc*b#W*	13-11
Transfer to the Off-Line State	*Esco*	20-27
Transfer to the On-Line State	*Escn*	20-27
Transparency Mode (Pattern)	*Esc*v#O*	16-10
Transparency Mode (Source)	*Esc*v#N*	16-10
Transparent Data Transfer	*Esc&p#X*	5-11
Underware Function Configuration	*Esc&i#W*	6-20
Unit of Measure	*Esc&u#D*	6-9
Universal Exit Language (UEL)	*Esc%-12345X*	2-6
Vector Graphics Mapping Mode	*Esc*t#N*	20-9
Vector Graphics Operating Mode	*Esc*t#M*	20-9
Vertical Picture Dimension	*Esc*t#V*	20-11
Vertical Raster Resolution	*Esc*r#V*	20-8
Vertical Rectangle Size (decipoints)	*Esc*c#V*	16-23
Vertical Rectangle Size (PCL Units)	*Esc*c#B*	16-22
Vertical Tab	*<VT>*	20-3
Viewing Illuminant	*Esc*i#W*	14-50
X Offset	*Esc*b#X*	20-14

Index by Escape Sequence

<BEL>	Bell	5-4
<BS>	Backspace	5-4
<CR>	Carriage Return	5-4
<ESC>	Escape	5-4
<FF>	Formfeed	5-5
<HT>	Horizontal Tab	5-5
<LF>	Linefeed	5-6
<NUL>	Null	5-6
<SI>	Primary Font	9-7
<SI>	Shift In	5-6
<SO>	Secondary Font	9-7
<SO>	Shift Out	5-6
<SP>	Space	5-6
<VT>	Vertical Tab	20-3
@PJL COMMENT	Comment	2-6
@PJL ENTER LANGUAGE	Enter Language	2-6
Esc%#A	Enter PCL Mode	17-5
Esc%#B	Enter HP-GL/2 Mode	17-2
Esc%-12345X	Universal Exit Language	2-6
Esc&a#C	Move CAP Horizontal (Columns)	8-8
Esc&a#G	Duplex Page Side	6-4
Esc&a#H	Move CAP Horizontal (Decipoints)	8-7
Esc&a#L	Left Margin	7-20
Esc&a#M	Right Margin	7-20
Esc&a#N	Negative Motion	6-8
Esc&a#P	Print Direction	7-12
Esc&a#R	Move CAP Vertical (Rows)	8-9
Esc&a#T	Set Horizontal Tab	20-3
Esc&a#U	Clear Horizontal Tab	20-3
Esc&a#V	Move CAP Vertical (Decipoints)	8-10
Esc&b#T	Dry Timer	6-13
Esc&b#W	Peripheral Configuration	6-10
Esc&c#T	Text Path Direction	7-16
Esc&c#W	Cluster Printing	A-6
Esc&d#D	Enable Underline	5-12
Esc&d@	Disable Underline	5-12
Esc&f#F	Media Eject Length	7-6
Esc&f#G	Page Width	7-7
Esc&f#S	Push/Pop CAP	8-11
Esc&f#X	Macro Control	18-7
Esc&f#Y	Macro ID	18-7
Esc&i#W	Underware Function Configuration	6-20
Esc&k#E	Print Enhancement (Underline)	20-25
Esc&k#F	Shift-In/Shift-Out Control	20-25
Esc&k#G	Line Termination	5-7
Esc&k#H	Character Motion Index (CMI — formerly HMI))	7-14
Esc&k#I	Character Set Selection Control	20-23

Subject Index