
PCL 5 Developer's Guide



Edition 1
E0792

5961-0546
Printed in U.S.A. 7/92

Notice

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental consequential damages in connection with the furnishing, performance, or use of this material.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated without the prior written consent of Hewlett-Packard Company.

Copyright © 1992 by HEWLETT-PACKARD CO.

Intellifont is a registered trademark of Agfa, a division of Miles, Inc.

Type Director is a trademark of Agfa, a division of Miles, Inc.

CG Times is a product of Agfa, a division of Miles, Inc.

Univers, Times Roman, and Helvetica are trademarks of Linotype AG or its subsidiaries.

ITC Garamond & ITC Bookman are registered trademarks of International Typeface Corp.

PCL is a trademark of Hewlett-Packard Company.

PANOSE is a trademark of Elseware Corporation.

TrueType is a trademark of Apple Computer, Inc.

Printing History

First Edition — July 1992

Table of Contents

Chapter 1

Introduction

How to Use This Manual	1-1
Manual Organization.	1-1
Manual Terms and Conventions	1-6
Using the Examples.	1-6
Differences Between the PCL 4 and PCL 5 Printers . .	1-7
Additional LaserJet 4 Features.	1-10

Chapter 2

Setting Up A Print Job

Introduction	2-1
Printer Job Language (PJL)	2-1
Brief Overview of the LaserJet Print File Structure . .	2-3
The Print Environment	2-7
Overview.	2-7
Changing the Print Environment.	2-9
Reverting to Environment Default Values	2-11
LaserJet Job Setup.	2-13
Making Copies of the Job.	2-30
LaserJet IIISi Job Setup Commands	2-31
Memory Usage	2-33
Chapter Summary	2-37

Chapter 3

Page Setup

Introduction	3-1
Selecting a Paper Source	3-1
Selecting a Physical Page Size	3-2
The Actual Printable Area	3-4
Changing Orientation	3-9
Print Direction	3-11
Managing the Text Area	3-13
Controlling Top Margin	3-15
Establishing the Bottom Margin	3-16
Perforation Skip for Print-and-Space Formatting	3-17
Controlling the Left Margin	3-19
Controlling Right Margin	3-20
Changing Character Spacing (HMI)	3-21
Modifying Line Spacing (VMI)	3-22

Chapter 4

Cursor Positioning

Introduction	4-1
Current Active Position (CAP)	4-1
Print-and-Space Cursor Positioning	4-5
Absolute vs. Relative Positioning	4-6
Units of Movement	4-7
Saving the Cursor Position	4-10
Positioning the Cursor at the Limits of the Page	4-11
How the Paper Path Affects Cursor Placement	4-16

Chapter 5

Using Fonts

Introduction	5-1
Selecting Fonts	5-2
General Font Management	5-9
Working With Fonts	5-14
Justifying Text When Using Proportional Fonts	5-14
Adjusting Line Spacing for Point Size	5-17
Transparent Print Data and Special Characters	5-18
Underlining Characters	5-19
Font Headers	5-20

Chapter 6

AutoFont Support

Introduction	6-1
TFM File Structure	6-2
TFM Tag Descriptions	6-11
Font Identification	6-12
Font Parameters	6-23
Character Parameters	6-27
Kerning Information	6-29
Device Data	6-35
PANOSE Numbers	6-36
“Glue” File Description and Usage	6-37
File Format	6-37
Font Entries	6-38
Supported Fonts	6-47
Comparing Past Font Support With TFM Support	6-49

AutoFont Support	6-52
Hard-coding TFM Data	6-52
TFM Reader Integration	6-53
Available Tools for TFM Reader Integration	6-55
The TFM Reader Program	6-56
End-User Considerations	6-57
Using the TFM Reader	6-58
TFM Reader Data Flow	6-59
Modifying the TFM Reader	6-61
Accessing the TFM Data Structure	6-62
Supplied TFM Files	6-66
File Naming Convention	6-67
Sample TFM Implementation	6-68
Selecting Fonts Using TFM Information	6-73
Locating the TFM Files	6-79
The Glue File	6-80
Creating TFM Files	6-81
Available Tools	6-82
The TFM Writer	6-82
PANOSE Numbers	6-88
Modifying the TFM Writer	6-88
Compiling the TFM Writer	6-100

Chapter 7

Intellifont[®] Integration

Why Integrate Intellifont?	7-1
Font File Formats	7-1
Adding Intellifont to Your Application Software	7-5
Screen Fonts	7-13

Chapter 8

AutoFont Support For TrueType

Introduction	8-1
TrueType Information for TFM tags	8-1
The Development Environment for the TFM Writer	8-10
Compiler Version	8-10
Description of Files	8-11
Compiling the TFM Writer	8-12
Generating a TrueType TFM.	8-12

Chapter 9

Raster Graphics

Introduction	9-1
The Raster Graphics System	9-1
Using Raster Graphics	9-3
Raster Compression Modes	9-5
Compression Mode Performance	9-15
End Raster Graphics	9-16
Positioning the Cursor for Raster Graphics	9-17
Merging Text With Raster Graphics	9-17
Auto-Rotation of Raster Images	9-20

Chapter 10

Vector Graphics

Introduction	10-1
Additional LaserJet 4 Features	10-1
The Picture Presentation Directives	10-2
PCL Picture Frame & HP-GL/2 Plot Size	10-3
The Picture Frame Anchor Point	10-4
Basic Steps for Importing an HP-GL/2 Plot	10-4
Basic Steps for Creating an HP-GL/2 Plot	10-5
When to Use Vector Graphics vs. Raster Graphics . . .	10-6
HP-GL/2 & PCL Orientation Interactions	10-7
Vector Graphics Limits	10-9
The Scaling Factor and the Picture Frame	10-10
Memory usage and HP-GL/2	10-12
Using HP-GL/2 Commands	10-13

Chapter 11

The Print Model

The Print Model—Filling With Patterns	11-1
How the Print Model Works	11-3
Using Rectangular Area Fill	11-5
Patterning Other Images	11-7
Using the Print Model Commands	11-7
User-Defined Patterns	11-13

Chapter 12

Using Macros

Introduction	12-1
Creating Macros	12-2
Using Macros	12-2
General Macro Management	12-4
HP-GL/2 in the Macro Environment	12-6
Summary of Rules Concerning Macros	12-6

Chapter 13

Tips for Efficient Programming

Introduction	13-1
General Tips	13-1
Combining Escape Sequences	13-1
Job Setup	13-2
For Non-PJL LaserJet Printers	13-2
For PJL LaserJet Printers	13-2
Page Setup	13-4
General Print Job Initialization	13-5
Non-PJL Example	13-7
LaserJet IIISi Example	13-7
LaserJet 4 Example	13-8
Using Fonts	13-8
Font Support	13-9
Raster Graphics	13-9
The Print Model	13-10
Vector Graphics	13-11
Macros	13-12

Chapter 14

Common Problems & Their Solutions

Introduction	14-1
Missing Characters or Graphics	14-1
Running Out of Memory (Error 20)	14-1
Print Overrun (Error 21)	14-3
Reset (E _C E) Deleting Temporary Fonts and Macros . .	14-3
PJL-Specific Problems	14-3
Reset (E _C E) Causing Printing of Partial Pages	14-4
Clipped Graphic Images	14-4
HP-GL/2 Images Not Printing Properly	14-5

Appendix A

LaserJet Printer Features and Compatibility

Feature Support Table.	A-1
LaserJet Compatibility Issues	A-10

Appendix B

Programming Examples

Introduction	B-1
Fonts and Print Direction	B-1
Printing Rules Using the Print Model	B-3
Print Model Font Effects	B-3
HP-GL/2 Font Effects	B-6
HP-GL/2 Graphics	B-9
Raster Graphics Compression.	B-10

Appendix C

Integrating Rambo

Introduction	C-1
The Rambo Command Line	C-5
Example Implementation Using Rambo	C-12
Printing the Test Files	C-18

Appendix D

Using the BUILDSYM Utility

Introduction	D-1
The BUILDSYM Kit	D-1
How Does BUILDSYM Work?	D-2
Using BUILDSYM	D-5
Creating a Symbol Set Definition File (.SYM File) ...	D-6

Appendix E

Using the FASST Utility

Overview	E-1
The FASST Kit	E-1
File Formats	E-2
FASST Shell Usage	E-3
FASST Integration	E-5
Major Functions	E-6
Modifying the Code	E-9
Testing Data	E-11

Appendix F

Using the SRTool Utility

Introduction	F-1
Installing and Running SRTool.	F-2
Initialization.	F-2
The User Interface.	F-5
PCL Status Readback	F-10
PJL Status Readback	F-13
PCL Command Macros	F-14
PJL Command Macros	F-17
Options	F-18

Appendix G

Kerning Information

Introduction	G-1
Pair Kerning.	G-2
Sector Kerning.	G-3
Track Kerning	G-6

Appendix H

HP MSL Character Number Table

Index

Contents

How to Use This Manual	1-1
Manual Organization	1-1
Manual Terms and Conventions	1-6
Using the Examples	1-6
Differences Between the PCL 4 and PCL 5 Printers ..	1-7
Additional LaserJet 4 Features	1-10

How to Use This Manual

This manual is designed for software developers as a complementary document to the *PCL 5 Printer Language Technical Reference Manual*. While the Technical Reference Manual provides an authoritative description of each PCL printer language command and its syntax and operation, this manual provides more of a tutorial with examples that illustrate how to use PCL commands in the design of LaserJet-compatible software.

Manual Organization

Each chapter contains a discussion of features related to a specific application and provides examples to illustrate the concepts. Examples stand out easily and use a step-by-step approach to explain the commands.

Some printer features are simple to grasp and require little explanation while others require several pages of explanations and examples. Wherever possible, examples are designed to closely match situations that you will experience as a software developer.

Note



Since this is primarily a PCL 5 printer document, all examples apply to the PCL 5 LaserJet printers (LaserJet III, IIIP, IIID, IIISi, 4). Other LaserJet printers may or may not support particular features. Check Appendix A for feature support across the entire LaserJet printer family.

Chapter 1. Introduction

This chapter gives a brief overview of each section of the manual. Also included is a general discussion of how examples are presented. In addition, there is a brief description of the major PCL 5 LaserJet printer features to give you an idea of new ways to enhance software applications, especially if you have written LaserJet software in the past.

Chapter 2. Setting Up a Print Job

This chapter describes the most effective way to begin a print job, especially when multiple users will be using the

printer. The “print environment” and the reset feature are discussed in detail, along with discussions about printing multiple copies and memory usage. Printer Job Language (PJM) is also discussed as it pertains to setting up a print job.

Chapter 3. Page Setup

This chapter expands the job setup discussion and explains page formatting. Features that can be changed from page to page are covered, such as page orientation, margins, page size, page source, print direction, and logical pages.

Chapter 4. Cursor Positioning

This chapter discusses the Current Active Position (CAP) and the various ways to position the cursor on the page. It gives examples for placing text at the edges of the page and explains the push/pop feature that allows you to store the Current Active Position.

Chapter 5. Using Fonts

This chapter covers selecting and managing fonts and also explains font applications such as justification, adjusting line spacing, and accessing special characters.

Chapter 6. AutoFont Support

This chapter discusses integrating LaserJet font metrics support into software applications. It explains the use of tools available to help you add enhanced font metrics support to your application. The Tagged Font Metric (TFM) specifications are also included in this chapter.

Chapter 7. Intellifont Integration

Chapter 7 discusses adding the Intellifont font scaling technology to your application. The benefits of adding Intellifont are discussed, along with the integration options and available tools for incorporating Intellifont. The HP Font Installer utility is also covered regarding its role in Intellifont integration.

Chapter 8. AutoFont Support for TrueType

This chapter supplements the Chapter 6 AutoFont Support discussion by including TFM tag and development information for generating TFM files from TrueType typefaces.

Chapter 9. Raster Graphics

This chapter explores the Raster Graphics System, an improved method of printing raster graphics that provides for more efficient raster files; enhancements include the raster height and width commands, a Y offset command, and four raster data compression modes.

Chapter 10. Vector Graphics

This chapter covers the HP-GL/2 vector graphics capabilities of the PCL 5 LaserJet printers, and gives guidance for when to use vector graphics instead of raster graphics.

Chapter 11. The Print Model

This chapter discusses the Print Model, which allows applications to fill images, rectangles and fonts with shades and patterns. Using the Print Model, the printer can print special effects, including composite images; these effects are demonstrated in detail. Control over the layering of images is determined by the printer's transparency modes, which are also demonstrated.

Chapter 12. Macros

This chapter elaborates on macros, a powerful LaserJet printer feature that is particularly useful for electronic forms applications. Topics discussed include when to use macros, how they are used, and pertinent macro examples.

Chapter 13. Tips for Efficient Programming

This chapter offers guidance for optimizing printer performance. As in most programming languages, there is usually more than one way to perform a particular function. This chapter helps you enhance your LaserJet software or

printer driver by giving you some hints as to how to implement certain features.

Chapter 14. Common Problems and Their Solutions

This chapter contains categorized answers to common PCL programming problems. Look through this chapter for insights into potential problems before you begin coding or when you are trying to solve an existing problem.

Appendix A. LaserJet Printer Features & Compatibility

Although all LaserJet printers have many features in common, this appendix clarifies the feature differences between, for example, the LaserJet series II and the PCL 5 LaserJet printers.

Appendix B. Programming Examples

Hewlett-Packard provides sample programs that demonstrate several features of the PCL 5 LaserJet printers. The programs are written in the C language with both the source code and executable code included on disk. This appendix shows illustrations of the samples that these programs print, along with a short explanation of what each program demonstrates.

Appendix C. Integrating the Rambo Utility

This appendix discusses adding the Rambo utility to your application for creating both “bound” and “unbound” scalable typeface files that can be downloaded to the PCL 5 LaserJet printers.

Appendix D. Using the BUILDSYM Utility

Appendix D discusses the BUILDSYM utility, which makes it easy to create customized symbol sets for your applications (for the HP LaserJet IIIP and LaserJet 4 printers).

Appendix E. Using the FASST Utility

The FASST utility provides optimal raster data compression and makes it easy to add this functionality to your software.

Appendix F. Using the SRTool Utility

Appendix F explains how to use the SRTool utility to send PCL escape sequences and PJP commands to the LaserJet 4 printer. SRTool is designed for use as a development tool for adding PCL and PJP status readback to your applications.

Appendix G. Kerning Information

This appendix explains sector kerning, track kerning, and pair kerning. It is provided as background information for the kerning tags discussed in Chapter 6.

Appendix H. HP MSL List

The HP MSL list is used for cross-referencing Agfa (CG) character numbers with HP Master Symbol List (HP MSL) character numbers. This appendix is useful when using the BUILDSYM utility.

Manual Terms and Conventions

The PCL Printer Language consists of a set of control codes and escape sequences that are used to control the LaserJet printers. The PCL Printer Language syntax is described in detail in the *PCL 5 Printer Language Technical Reference Manual*.

Throughout this developer's guide, PCL escape sequences are shown in bold 10-point type using a font that allows you to easily distinguish between the lower-case L (*ℓ*) and the number 1. Likewise, the zero character (*∅*) contains a slash while the upper-case letter O does not. The escape character (ASCII 27) is represented as “*Esc*”.

Using the Examples

Software developers use several different programming languages to send escape sequences to the LaserJet printers. Since there are a variety of languages used, almost all examples are given in a “generic” format instead of in a specific programming language. However, there are some cases where it helps tremendously to see how a feature is implemented with a specific programming language. HP provides some program examples on disk that are coded in the C Programming Language, including both source code and executable code.

To help communicate the logic behind many of the programming decisions, the examples in this manual include comments that provide insight into the reasoning behind the commands used.

Note



These examples show commands on separate lines for clarity—in reality, printer commands should not be sent separately, but all in one string. For example:
`␣␣␣&l1X␣&l2A␣&l0O␣&l2E␣(8U␣(s1p14v0s3b4148T`
Also note that the # parameter listed in many of the examples indicates an ASCII value for the value chosen, not the hexadecimal equivalent.

Wherever possible, examples contain cross-references to other related examples in the book. For instance, a raster graphics example may contain cursor positioning commands for positioning the graphic image on the page that may be of help to someone learning cursor positioning.

Differences Between the PCL 4 and PCL 5 Printers

This section provides brief descriptions of the major enhancements that PCL 5 provides to the PCL Printer Language. LaserJet 4 features are discussed in the section that follows this one. See Appendix A, or the *PCL 5 Comparison Guide*, for feature differences between the PCL 5 printers.

Scalable Fonts

The PCL 5 LaserJet printers provide font scaling from .25 point to 999.75 point, in quarter-point increments. All PCL 5 printers contain the scalable CG Times and Univers typefaces in four treatments (and some contain even more typefaces); additional typefaces can be downloaded to the printer or accessed by way of scalable typeface cartridges.

AutoFont Support (TFM Files)

AutoFont Support provides developers with an efficient method of reading LaserJet font metrics, for PCL 4 *and* PCL 5 devices; one driver can be written to cover all LaserJet printer fonts, eliminating the need to develop new drivers for every new font product. TFM files are distributed by

HP for internal LaserJet printer fonts and are included as AutoFont Support disks with purchased font products.

HP-GL/2 Functionality

PCL 5 incorporates HP-GL/2 functionality that allows the printing of HP-GL/2 vector graphics files. The HP-GL/2 implementation available in PCL 5 LaserJet printers provides features not available in other HP-GL devices. (For example, access to high-quality bitmapped and scalable fonts, as well as use of PCL fill patterns.) Raster graphics and vector graphics, as well as standard PCL text, can be printed on the same page, providing a high degree of compatibility with many existing HP-GL/2 applications.

The LaserJet 4 printer adds even more HP-GL/2 features, including support for bezier curves, another fill type for filling polygons (non-zero winding fill), and PCL-compatible label origin. In addition, the LaserJet 4 printer, unlike the other PCL 5 printers, allows the use of HP-GL/2 commands in macros.

Multiple Print Directions

The print direction can be specified as either 0, 90, 180, or 270 degrees for applications that require multiple print directions, for both text and graphics, on a single page.

Raster Data Compression

The PCL 5 LaserJet printers have up to four raster data compression modes to help increase performance and minimize disk space requirements for graphics-intensive applications. The HP-developed FASST utility is provided (Appendix E) to help your application optimize the raster data sent to each LaserJet printer.

Print Model

The Print Model feature allows graphic images (including fonts) to be filled with shades of gray or patterns. A common use of this capability is for the printing of reverse type (white letters on a black background) and shaded or pattern-filled fonts. Another use is for creating special effects with images by filling them with patterns or for overlaying multiple images.

Compressed Fonts (Bitmaps)

The PCL 5 LaserJet printers support font compression. (Font compression is discussed in the *PCL 5 Printer Language Technical Reference Manual*.)

Macro Cartridge Support

With the PCL 5 LaserJet printers, cartridges containing macros can be plugged into the printer's font cartridge slots for applications requiring quick access of forms, logos, letterhead, and frequently used graphics. (See Chapter 12 for more information about macro cartridges.)

PJL Support

The LaserJet IIISi, PaintJet XL300, and LaserJet 4 printers offer various levels of support for Hewlett-Packard's Printer Job Language (PJL). The LaserJet IIISi and PaintJet XL300 printers provide base-level PJL support, which mainly features printer language switching. The LaserJet 4 printer offers a more extensive set of PJL features, including status readback and the ability to set the control panel from a remote location. PJL is discussed in Chapter 2 and Appendix F of this manual, and in the *PJL Technical Reference Manual*.

Additional LaserJet 4 Features

The LaserJet 4 printer adds several features to the existing PCL 5 feature set. Some of the significant LaserJet 4 enhancements are listed below. See Appendix A for a comparison of the differences in support for the various PCL 5 LaserJet printer features.

- **Variable Resolution**—the LaserJet 4 printer offers resolutions of both 600 and 300 dots per inch. Text and graphics are supported at both resolutions.
- **PJL Support**—the LaserJet 4 printer supports 18 PJL commands, providing capabilities such as status readback, remote control panel operation, control over displayed messages, language switching, and more. (This manual provides basic PJL information; see the *PJL Technical Reference Manual* for detailed information.)
- **Additional Resident Typefaces**—the LaserJet 4 printer comes standard with many scalable typefaces, including various treatments of CG Times, Univers, CG Omega, Clarendon Condensed, Coronet, Antique Olive, Garamond, Albertus, Arial, and Times New Roman.
- **Bezier Curves**—additional HP-GL/2 features include support for bezier curves, another fill type for filling polygons (non-zero winding fill), and PCL-compatible label origin. Like the LaserJet IIIP printer, the LaserJet 4 printer supports user-defined patterns for screened vectors and for filling polygons.
- **Units of Measure command**—instead of using dots as a unit of measurement, the LaserJet 4 printer offers the Unit of Measure command, which affects all “dot” moves. This command allows you to select a PCL Unit size in specified increments from 96 to 7200 units/inch.
- **Number of Copies**—like the LaserJet IIISi printer, the LaserJet 4 allows the number of copies to be set to any number from 1 to 32767.
- **Page Size**—the LaserJet 4 printer supports the B5 envelope size, along with four other envelope sizes and four paper sizes.

Contents

Introduction	2-1
Printer Job Language (PJL).	2-1
Brief Overview of the LaserJet Print File Structure	2-3
PJL Printers	2-3
Non-PJL Printers	2-6
The Print Environment	2-7
Overview	2-7
Changing the Print Environment	2-9
Reverting to Environment Default Values	2-11
LaserJet Job Setup	2-13
Job Setup for Non-PJL Printers	2-13
Job Setup for the LaserJet IIISi Printer	2-16
Job Setup for the LaserJet 4 Printer	2-22
Making Copies of the Job	2-30
LaserJet IIISi Job Setup Commands	2-31
LaserJet IIISi Job Offset	2-31
LaserJet IIISi Output Bin Selection	2-32
Memory Usage	2-33
Page Protection (Error 21).	2-33
Page Protection for the LaserJet 4 Printer	2-34
PJL and Perishable Data	2-34
Freeing Memory for Fonts, Macros, Graphics	2-35
Chapter Summary	2-37

Introduction

At the beginning of every print job, there are some basic setup commands that must be issued before sending print data to LaserJet printers. This chapter helps you set up your print job correctly, so that you can minimize surprises and maximize compatibility with other applications that may be running on the same workstation or network. Along with some explanation of the reasoning behind the proper way of setting up a print job, this chapter provides some examples you can use as a template, or at least a guide, for developing your applications.

The first part of the chapter gives you a description of P JL, the Printer Job Language supported by some PCL 5 printers, and an overview of the way LaserJet print jobs are structured. Then the print environment is discussed as a preparation for setting the printer to a desired state. The remaining portion focuses on job setup, including discussion of two LaserJet IIISi printer features (*job offset* and *output bin selection*). In addition, this chapter provides information on memory usage and avoiding print overrun using the *page protection* feature.

Printer Job Language (P JL)

With the widespread use of both PCL printer language and PostScript on networks, the need for programmatic language switching is very important. Manually switching from one language to another using control panel settings is not an optimal solution in most cases, since several users are usually involved and there is usually a distance between the users and the printer.

In order to provide system language switching, the HP LaserJet IIISi and LaserJet 4 printers have a Printer Job Language that resides logically above the PCL and PostScript languages. The base set of P JL commands, supported by the LaserJet IIISi printer, allows application software to easily and reliably switch between PCL and PostScript.

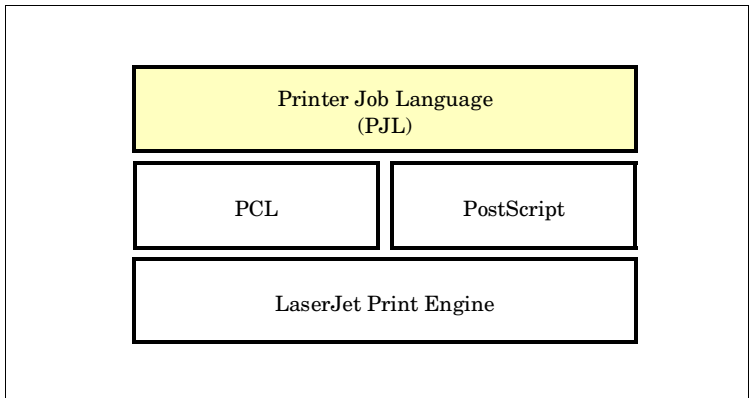


Figure 2-1. PJL Resides Above PCL and PostScript

PJL is not limited to just language switching, however. The LaserJet 4 implementation of PJL offers language switching plus a broader range of features including comprehensive printer status readback capabilities and the ability to manipulate control panel settings and create custom display messages. The PJL information in this chapter is provided as a brief overview and to help you create well-formed jobs—that is, jobs that print smoothly on single-user workstations as well as multi-user networks.

If you are writing applications for PJL printers, read the PJL information later in this chapter and study the examples for the LaserJet IIISi and LaserJet 4 printers. For a more detailed look at PJL, see the *PJL Technical Reference Manual*.

Note 

Since PJL printers are frequently used on networks, it is extremely important that jobs be well-formed. This chapter, and the *PJL Technical Reference Manual*, provide examples of many well-formed PJL jobs. Follow these examples to ensure smooth printer operation on all types of systems, from single-user workstations to networks.

Brief Overview of the LaserJet Print File Structure

Before we get into specific programming functions, let's discuss the basic structure of a typical LaserJet print job. Since job structure for LaserJet printers supporting PJP is different than for non-PJP printers, structure is discussed separately.

PJP Printers

For PJP printers (LaserJet 4 and LaserJet IIISi), most jobs are structured as follows:

- PJP Commands
- PCL Job Setup Commands
- Page Setup Commands
- Print Data
- PJP Commands

Figure 2-2 illustrates the general structure of a LaserJet print file using PJP and PCL:

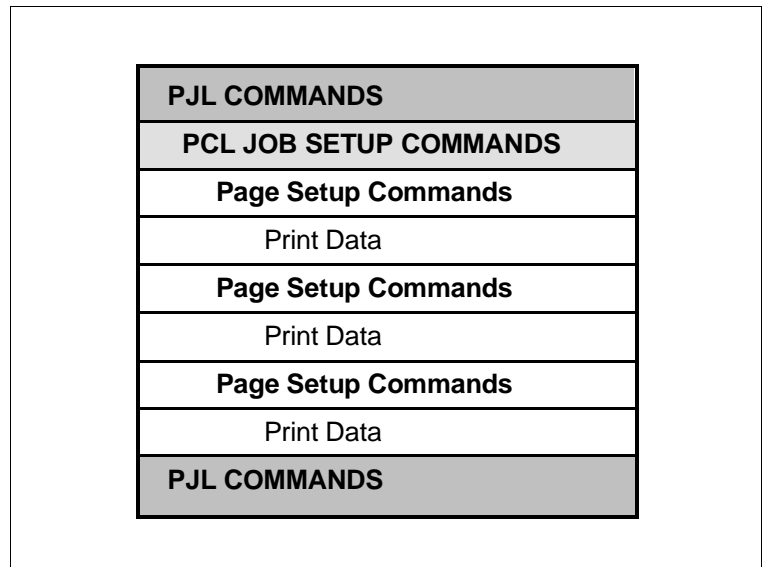


Figure 2-2. File Structure for PJP Printers

PJL Commands signal the beginning and end of the print job and specify the printer language (personality) used to print the job. There are three basic PJL commands that are supported by all PJL printers (the LaserJet 4 printer has many more PJL commands that offer many more functions, however, the three commands listed below provide the basis for generating well-formed print jobs):

- Universal Exit Language (UEL) Command:
Ⓔ%*-12345X*

The UEL command's function is to transfer control to PJL. The command should be used at the very beginning and end of every PJL job. For the UEL command at the beginning of the job, the X at the end of the command must always be immediately followed by another PJL command (or the @PJL command prefix). There can be no spaces or characters between the X and the @ that begins the next command. On the other hand, the UEL command at the end of the job is the final part of the job and does not require anything to follow it.

- ENTER Command:
@PJL ENTER LANGUAGE = *personality*

This command selects the printer language to be used. The command line always ends with a required line feed character (<LF>). (The line feed character can be immediately preceded by a carriage return character [<CR>].)

- COMMENT Command:
@PJL COMMENT *words*

The COMMENT command allows you to add comments to your PJL code. This command also ends with a required line feed (<LF>) character. (The line feed character can be immediately preceded by a carriage return character [<CR>].)

Note

Most of the examples in this document begin with the `^E` reset command. Software developed for the LaserJet IIISi and LaserJet 4 printers should have PJJL commands before and after the `^E` commands that precede and succeed the job. The examples in this chapter for the LaserJet IIISi and LaserJet 4 printers demonstrate how these commands are properly used. For more PJJL information, consult the *PJJL Technical Reference Manual*.

PCL Job setup commands usually control the output of the entire print job and are grouped together at the beginning of a print file. They include functions such as reset and selecting the number of copies.

Page setup commands are associated with either a single page or with groups of pages. They include functions such as orientation, line spacing or VMI (vertical motion index), paper size, paper length, paper source, margins and text length.

Print data is the data that the user wants to print on the page, along with commands that select fonts, call macros, and position the data. Included with the document data are also any downloaded fonts or macros, usually placed in the print file immediately following the page setup commands.

Note

After the first page, page setup commands do not need to be sent unless a change from the previous page settings is desired. For example, if the top margin is set at the beginning of a print job, it is not necessary to send another top margin command unless you need to change the top margin.

Non-PJL Printers

For non-PJL printers, jobs are structured the same as PJL printers, only without the PJL commands at the beginning and end of the job.

- PCL Job Setup Commands
- Page Setup Commands
- Print Data

Figure 2-3 illustrates the general structure of a LaserJet print file for printers without PJL:

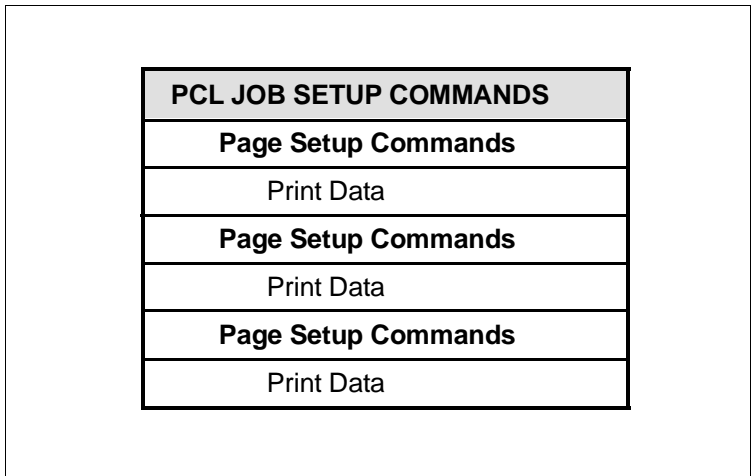


Figure 2-3. File Structure for Non-PJL Printers

The Print Environment

Overview

Before your application begins printing, many printer features may have been set from a prior application or from someone changing control panel settings. These feature settings are usually not the same as those desired for your application.

Whatever the current settings, your software application must be able to set the LaserJet printer to a desired state to avoid problems. If this is not done, and the printer's features are set differently than what the application "expects," problems can occur. For example, if the printer is being used by several people and the previous user sets the control panel (or sends a PJI DEFAULT COPIES command) to print 65 copies of a job, your application will also print 65 copies (unless you take the measures we will discuss to prevent previous jobs or control panel settings from interfering).

At any time during printer operation, the printer's current feature settings are referred to collectively as the *print environment*. The printer constantly maintains five different print environments:

- **Factory Default Environment**—the Factory Default Environment is the set of default features that were programmed into the printer when it was built. The print environment settings are the same as the Factory Default Environment when the printer is first powered on before the operator control panel settings are changed or any printer commands are sent from an application. Any of the Factory Default environment settings can be overridden using either the control panel keys, the PJI SET command, or printer language commands. The print environment settings revert to the Factory Default Environment when a RESET MENU is performed from the control panel or when the PJI INITIALIZE command is received.

- **User Default Environment**—the User Default Environment contains the control panel settings. The User Default Environment settings are modified when the control panel settings are changed or when the PJJL DEFAULT command is used. The User Default Environment reverts to the Factory Default values when a RESET MENU or PJJL INITIALIZE command occurs.
- **PJJL Current Environment (PJJL printers only)**—at the beginning of a PJJL job, the PJJL Current Environment is the same as the User Default Environment. Once PJJL is entered, the PJJL Current Environment is modified using the PJJL SET command. Features set using the SET command are active for the duration of the PJJL job. (If the PJJL JOB command is used, the PJJL job ends as soon as the printer receives the EOJ command; if the JOB command is not used, the PJJL job ends when the next UEL command is encountered.) The PJJL Current Environment feature settings override the User Default (control panel) settings, which in turn are overridden by printer language commands.
- **Modified Print Environment**—at the beginning of a PCL job, the PCL reset (E_cE) causes the PJJL Current Environment features to be loaded into the Modified Print Environment (for non-PJJL printers, the User Default Environment features are loaded). The Modified Print Environment settings are changed using printer language commands, such as PCL commands.
- **Overlay Print Environment**—the overlay print environment is active whenever an automatic macro overlay is enabled. Once the macro overlay is finished, the Modified Print Environment is again active.

Note



The *PCL Comparison Guide* lists the features contained in each print environment for each HP LaserJet printer.

Changing the Print Environment

It helps to think of the print environment as a hierarchical structure, with the Factory Default settings being overridden by any control panel settings or PJJL DEFAULT commands, creating the User Default Environment. For PJJL printers, PJJL SET commands override the User Default environment settings, creating the PJJL Current Environment. Printer commands sent by an application override the User Default and PJJL Current Environments, creating the Modified Print Environment. In those cases when the automatic macro overlay feature is enabled, then the overlay environment overrides the Modified Print Environment.

Any feature that hasn't been set by the application defaults to the PJJL Current Environment value (if any have been set using the PJJL SET command). Any feature that hasn't been set using the PJJL SET command defaults to the User Default (control panel) value. For example, say the user default setting for the number of copies is 1, and a user sets the number of copies to 6 from the control panel. If the user's application doesn't send a command to set the number of copies, the printer will print 6 copies of the next print job.

Changing Environment Settings

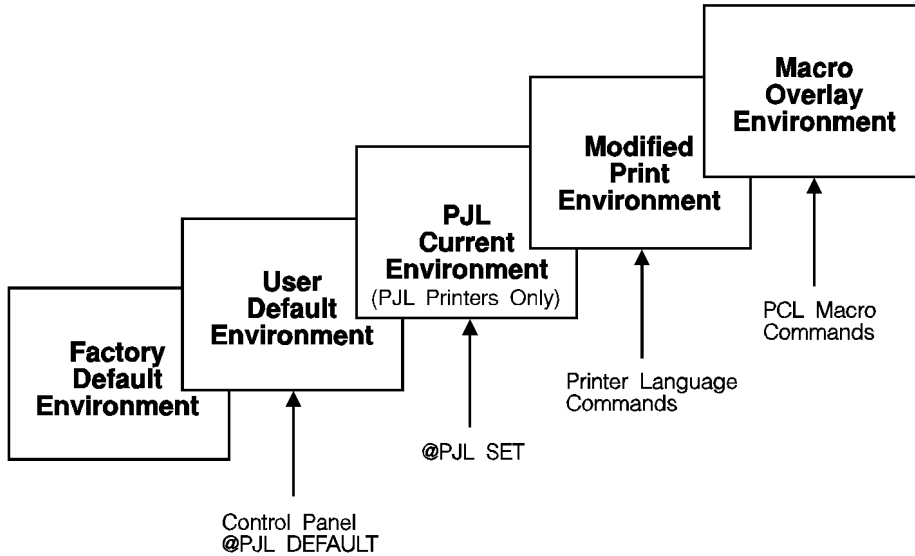


Figure 2-4. Changing Environment Settings

Reverting to the Print Environment Default Values

Resets cause the printer to default to its environment settings. Depending on the type of reset performed, the features default to either the *Factory Default Environment*, the *User Default Environment*, or the *PJL Current Environment*.

Control Panel Reset Menu or **PJL INITIALIZE**—pressing the Reset Menu key on the control panel (until 09 RESET displays) or sending the PJL INITIALIZE command resets the print environment to the *Factory Default Environment*; it erases cached fonts and temporary downloaded fonts and macros, without affecting the configuration settings.

Power Cycling the Printer—switching the printer power off erases perishable data, including all macros and downloaded, scaled or rotated fonts. Switching the printer power on resets the printer to the *User Default Environment*.

Control Panel Reset—resetting the printer from the control panel (07 RESET) resets the printer to the *User Default Environment* and erases any temporary downloaded fonts and macros, as well as any cached fonts, from printer memory. Any data that is stored in the print buffer is erased without being printed.

PJL RESET, JOB, EOJ, or UEL commands—sending one of these commands to the printer resets the printer to the *User Default Environment* and erases any cached fonts and any temporary macros or temporary downloaded fonts. These commands perform the same function as the control panel reset, except with these commands the printer prints any data that is stored in the print buffer before clearing it.

EscE Reset, Language Reset, or PJL ENTER commands—the EscE reset, or any other printer language reset, and the PJL ENTER command default the print environment to the *PJL Current Environment*. The PJL Current Environment settings remain in effect for the duration of the PJL job.

UEL Command (within a PJJ JOB)—when the Laser-Jet 4 printer receives a UEL command after it receives a JOB command (but before the next EOJ command), the print environment defaults to the *PJJ Current Environment* settings. When the EOJ is received, the print environment defaults to the *User Default Environment* settings.

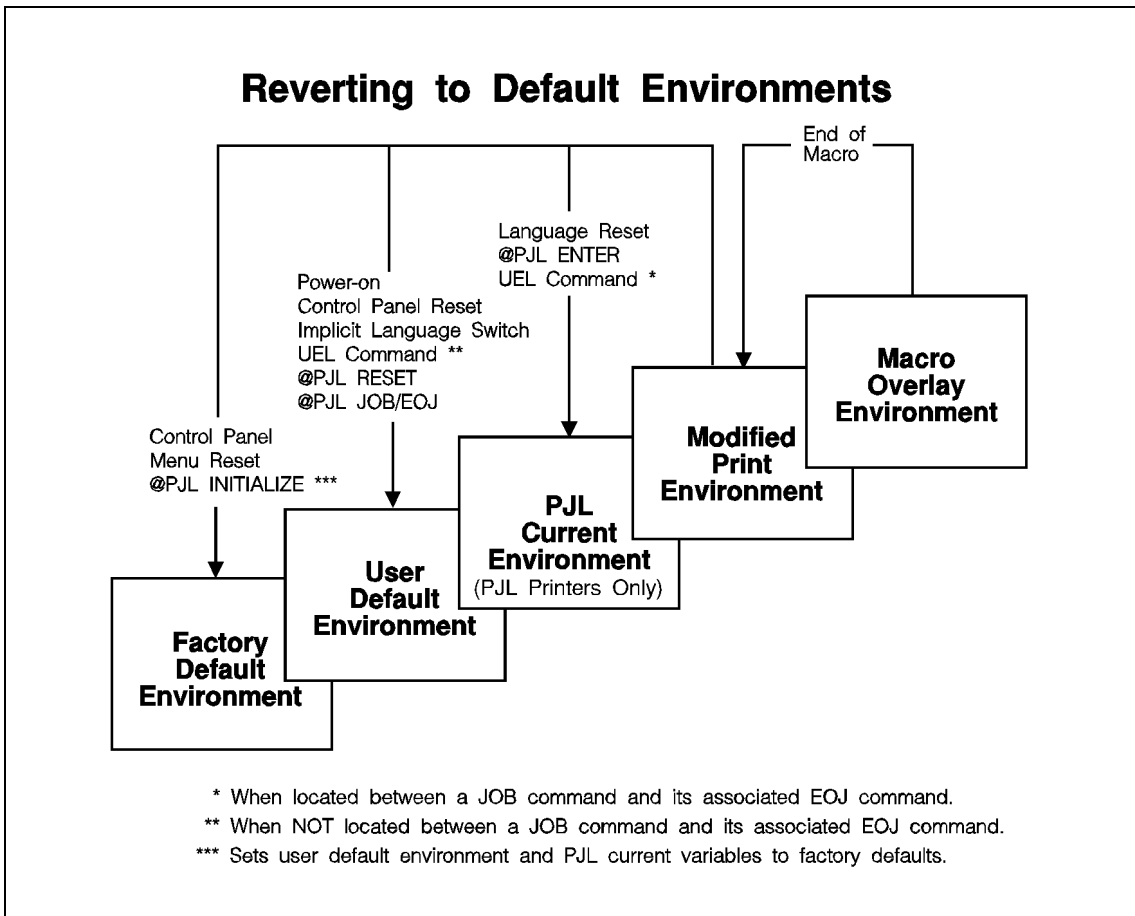


Figure 2-5. Reverting to Default Environment Settings

LaserJet Job Setup

Job setup for the LaserJet printers involves four major functions:

- Resetting the printer at the beginning of the job
- Setting the printer features to a desired state
- Sending the print data
- Resetting the printer at the end of the job

Since these functions are accomplished somewhat differently for different printers, this section divides the PCL 5 LaserJet printers into three groups for this discussion:

- Job Setup for Non-PJL Printers
- Job Setup for the LaserJet IIISi Printer
- Job Setup for the LaserJet 4 and other printers that support the PJL SET command

Job Setup for Non-PJL Printers

The job setup procedure for non-PJL printers, including the LaserJet III, IIID, and IIIP, consists of the following four steps:

- 1) Reset the printer at the beginning of the job using the `ESC` reset.
- 2) Send printer commands to set the printer to a desired state and override any environment settings that could have been set using the control panel.

If your application will not allow selection of a certain feature that is also selectable from the control panel, *do not* send a command to set that feature. It will prevent the user from selecting that feature from the control panel because the programmatic commands always override the control panel settings.

For example, if your application doesn't allow the user to select the number of printed copies, but it does send the number of copies escape sequence (`ESC`X), then the user is locked out from the number of copies feature. (He can't select it from the application and the application escape sequence overrides the number of copies setting on the

control panel.) In this case, the application should either implement the feature by allowing users to select the feature or should refrain from sending the number of copies command.

Below is a list of environment variables that you should consider making available to users. Each is selectable from the control panel and automatically defaults to the control panel setting with the `ESC E` command. The PCL commands listed in the table below set the environment variables to the value specified (for the duration of the PCL job).

Environment Variable Description	PCL Command
Number of Copies	<code>ESC &l#X</code>
Duplex printing (duplex printers only)	<code>ESC &l#S</code>
Tray/Manual Feed (paper source)	<code>ESC &l#H</code>
Paper (Page Size)	<code>ESC &l#A</code>
Orientation (portrait, landscape)	<code>ESC &l#O</code>
Form Length (Number of lines per page—VMI)	<code>ESC &l#C</code> or <code>ESC &l#D</code>
Symbol Set	<code>ESC (ID</code>
Selected Font	<code>ESC (s#p#h#v#s#b#T</code>

3) Send the print data.

4) Reset the printer at the end of the job.

The following example demonstrates the job setup process for non-PJL printers.

Example: Job Setup Without PJI

This example shows a sure way for a full-featured application to set non-PJI LaserJet printers to a desired state. This solution is for the LaserJet III, IIID, and IIIP printers and PCL 4 LaserJet printers.

Note



Some of the commands listed in this example are page setup commands and are covered in more detail in the following chapter. They are used in this example because they are part of the print job setup process (the page setup commands for the first page are grouped together at the beginning of the print file).

JOB SETUP COMMANDS

<code>^cE</code>	Reset printer to the User Default environment.
<code>^c&l3X</code>	Set the number of copies to 3.
<code>^c&l0S</code>	Select single-sided printing.

PAGE SETUP COMMANDS

<code>^c&l1H</code>	Select paper tray as paper source.
<code>^c&l26A</code>	Select A4-size paper (210 x 297mm).
<code>^c&l0O</code>	Set the orientation to portrait.
<code>^c&l8C</code>	Set the vertical motion index (VMI) to 6 lines per inch.

PRINT DATA

<code>^c(8U</code>	Set the symbol set to Roman-8.
<code>^c(s0p10h12v0s0b3T</code>	Select fixed-spaced, 10-pitch, 12-pt., upright, medium, Courier font.
PRINT DATA	Send print data.
<code>^cE</code>	Reset printer at end of print job. This helps prevent problems for the following job.

Note



Resets should be used at the beginning and end of the job, but they are not recommended anywhere else in the job. Besides resetting features, these commands eject the current page if they are received after printable data.

For another example of a job setup string, see the *General Print Job Initialization* discussion in Chapter 13.

Job Setup for the LaserJet IIISi Printer

Job setup for the LaserJet IIISi printer is similar to that for non-PJL printers, however the LaserJet IIISi printer supports PJL language switching and has some extra job setup features that are not found in the other PCL 5 printers (job offset and output bin selection). LaserJet IIISi print jobs should be structured as follows:

- 1) Send a UEL command ($\text{\%E}c\%-12345X$) at the beginning 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 \%E reset before sending PCL commands.
- 4) Send printer commands to override any environment settings that could have been set using the control panel and to set the printer to a desired state.

If your application will not allow selection of a certain feature that is selectable from the control panel, *do not* send a command to set that feature. It will prevent the user from selecting that feature from the control panel because programmatic commands always override the control panel settings.

For example, if your application doesn't allow the user to select the number of printed copies, but it does send the number of copies escape sequence ($\text{\%E}c\&11X$), then the user is locked out from the number of copies feature. (He can't select it from the application and the application escape sequence overrides the number of copies setting on the

control panel.) Instead, the application should either implement the feature by allowing users to select the feature or should refrain from sending the number of copies command.

Below is a list of environment variables that you should consider controlling with printer commands. Each is selectable from the control panel and automatically defaults to the control panel setting with the `ESC E` reset or UEL command.

Environment Variable Description	PCL Command
Number of Copies	<code>ESC &l#X</code>
Duplex printing	<code>ESC &l#S</code>
Tray/Manual Feed (paper source)	<code>ESC &l#H</code>
Paper (Page Size)	<code>ESC &l#A</code>
Orientation (portrait, landscape)	<code>ESC &l#O</code>
Output Bin Selection	<code>ESC &l#G</code>
Job Offset	<code>ESC &l#T</code>
Form Length (Number of lines per page—VMI)	<code>ESC &l#C</code> or <code>ESC &l#D</code>
Symbol Set	<code>ESC (ID</code>
Selected Font	<code>ESC (s#p#h#v#s#b#T</code>

5) Send the print data.

6) Reset the printer at the end of the job using `ESC E` followed by the UEL command (`ESC %-12345X`).

The following example demonstrates job setup for the LaserJet III Si printer.

Example: Job Setup for the LaserJet IIISi Printer

This example shows commands for setting the LaserJet IIISi printer to a desired state. The approach is almost the same as with the non-PJL printers, with the exception of the UEL and PJL ENTER LANGUAGE commands before the job and the UEL command at the end. Note that in all PJL jobs there should be no space between the X at the end of the UEL command and the @ that begins the next command.

PJL COMMANDS

Esc%-12345X@PJL <CR><LF>

@PJL ENTER LANGUAGE = PCL <CR><LF>

Universal Exit Language (UEL) command acts as a reset and gives control to PJL. The ENTER command gives control to the PCL printer language.

JOB SETUP COMMANDS

EscE

Reset printer.

Esc&l1X

Set the number of copies to 1.

Esc&l1T

Enable job offset so that each PCL job is physically offset from the preceding job.

Esc&l1G

Select the upper output bin.

PAGE SETUP COMMANDS

Esc&l1H

Select paper tray as paper source.

Esc&l2A

Select letter-size paper.

Esc&l0O

Set the orientation to portrait.

Esc&l8C

Set the vertical motion index (VMI) to 6 lines per inch.

PRINT DATA

`Esc(OU`

Set the symbol set to ASCII.

`Esc(s1p14v1s0b4168T`

Select a proportional, 14-point, italic, medium, Antique Olive font.

PRINT DATA

Send print data.

`EscE`

Reset printer at end of print job. This restores the printer to the User Default Environment for the next print job, helping to prevent problems for the following job.

PJL COMMAND

`Esc%-12345X`

Universal Exit Language (UEL) command ends the job and returns control to PJL.

Note



UEL commands and resets should be used at the beginning and end of the job, but they are not recommended anywhere else in the job. Besides resetting features, these commands eject the current page if they are received after printable data.

Example: Job Setup for the LaserJet IIISi Printer Using Language Switching

This example shows job setup commands for a LaserJet IIISi print job that switches printer languages. Note that in all PJP jobs there should be no space between the X at the end of the UEL command and the @ that begins the next command.

PJP COMMANDS

ESC%-12345X@PJP <CR><LF>

The UEL command acts as a reset and gives control to PJP. Note how it is immediately followed by the @ of the next PJP command.

@PJP ENTER LANGUAGE = PCL<CR><LF>

The ENTER command gives control to the PCL printer language.

JOB SETUP COMMANDS

ESC E

Reset the printer.

ESC &l 1X

Set the number of copies to 1.

PAGE SETUP COMMANDS

ESC &l 1H

Select paper tray as paper source.

ESC &l 2A

Select letter-size paper.

ESC &l 0O

Set the orientation to portrait.

ESC &l 8C

Set the vertical motion index (VMI) to 6 lines per inch.

PRINT DATA

ESC (0U

Set the symbol set to ASCII.

ESC (s1p37v0s3b4148T

Select proportional, 37-point, upright, bold, Univers font.

PRINT DATA

Send print data.

EscEsc%-12345X

Reset and UEL command to end the PCL job and return control to PJL.

PJL COMMANDS

Esc%-12345X@PJL <CR><LF>

@PJL COMMENT End of PCL job <CR><LF>

@PJL COMMENT Beginning of PostScript job <CR><LF>

@PJL ENTER LANGUAGE = POSTSCRIPT<CR><LF>

PRINT DATA

5 setlinewidth<CR>

PostScript print data.

100 100 moveto<CR>

...

<EOT>

PJL COMMANDS

Esc%-12345X

UEL command to end the job and return control to PJL.

Job Setup for the LaserJet 4 Printer

The LaserJet 4 printer, with its extensive PJJ feature set and status readback capabilities, allows you to request printer status and set control panel features such as resolution enhancement (RET), page protection, and resolution.

Note



The LaserJet 4 printer is the only LaserJet printer capable of status readback.

Listed below are two suggested approaches you can use to programmatically set the LaserJet 4 printer to a desired state (an example is included for each approach):

- Reset the printer to the *User Default Environment* or *PJJ Current Environment* and then send commands that override any environment settings that could have been set using the control panel or PJJ.
- Reset the printer to the *User Default Environment* or *PJJ Current Environment* and then request status information using the PJJ and/or PCL status readback commands. Then send the necessary commands to set the printer to the desired state.

In both of these methods, the reset defaults all environment variables to the *User Default Environment* or *PJJ Current Environment*.* Once a reset occurs, all that needs to be done to achieve a desired state is to send commands to override any undesirable settings caused by changes to the control panel settings or changes made using the PJJ SET command. If the printer has status readback capability, the application can first inquire the status and then send only those commands that are necessary to achieve the desired state.

- * The UEL command resets the printer, causing the *User Default Environment* values to take effect if no PJJ JOB command has been sent; the *PJJ Current Environment* values take effect if the printer has received a PJJ JOB command but hasn't received the accompanying EOJ command.

Follow the steps below for LaserJet 4 job setup:

- 1) Send a UEL command ($\epsilon_c\%$ -12345X) at the beginning of the job to give control to PJJL.
- 2) *If status readback is enabled*, use the PJJL INQUIRE command to request the status of environment variables. Compare the returned status information with the desired feature settings. *If status readback is not enabled*, assume that none of the environment feature settings are set as desired and set them using PJJL SET or PCL commands.

The PJJL status readback commands allow you to request the name of the printer, its configuration, and a list of environment settings. The PCL status readback commands allow you to request information on available fonts, macros, user-defined patterns, and symbol sets.

Note



In order to receive printer status, the application must be equipped to handle bi-directional communications. The hardware specification for adding this capability is available through the Hewlett-Packard Developers Support Program. Contact your HP support representative for more information.

Appendix A lists the PCL and PJJL status readback commands supported by the LaserJet 4 printer. Appendix F discusses using the SRTool utility for both PCL and PJJL status readback information. For more information on PJJL status readback, see the *PJJL Technical Reference Manual*. For more information on PCL status readback, see the *PCL 5 Printer Language Technical Reference Manual*.

- 3) Use the PJJL SET command to override any current settings that would interfere with the way you want the job processed.

Only use the PJJL SET command to override features that cannot be set using PCL. For the LaserJet 4 printer, these are: RET, PAGEPROTECT, RESOLUTION,

PERSONALITY, and TIMEOUT. (Since PERSONALITY and TIMEOUT do not affect the printed output, do not send these commands unless you have a specific need to modify the timeout duration or default personality.) Use PCL to set other features to a desired state.

- 4) Use the P_JL ENTER LANGUAGE command to enter PCL or another desired printer language.
- 5) Reset the printer using E_cE reset before sending PCL commands.
- 6) Send printer commands to set the printer to a desired state and to override any environment settings that could have been set using the control panel.

If your application will not allow selection of a certain feature that is also selectable from the control panel, *do not* send a command to set that feature. It will prevent the user from selecting that feature from the control panel because the programmatic commands always override the control panel settings.

For example, if your application doesn't allow the user to select orientation, but it does send the orientation escape sequence ($\text{E}_c\&l\#O$), then the user is locked out from the orientation feature. (He can't select it from the application and the application escape sequence overrides the orientation setting on the control panel.) Instead, the application should either implement the feature by allowing users to select the feature or should refrain from sending the orientation command.

The following table lists environment variables that you should consider controlling with printer commands. Each is selectable from the control panel and/or with the P_JL SET command. These features automatically default to either the control panel setting or the P_JL Current Environment setting when the E_cE reset or UEL commands are received (These features are also defaulted during other P_JL reset conditions such as when the printer receives the P_JL RESET, JOB, or EOJ commands).

Environment Variable	Variable Description	PCL Command	PJL Command
COPIES	Number of copies	$\text{E}_c\&l\#X$	@PJL SET COPIES *
PAPER	Page size (letter, A4, legal, etc.)	$\text{E}_c\&l\#A$	@PJL SET PAPER *
ORIENTATION	Paper orientation (portrait, landscape)	$\text{E}_c\&l\#O$	@PJL SET ORIENTATION *
FORM	Number of lines per page (VMI)	$\text{E}_c\&l\#C$ or $\text{E}_c\&l\#$	@PJL SET FORMLINES *
MANUAL FEED	Manual feed mode	$\text{E}_c\&l\#H$	@PJL SET MANUALFEED *
RET	Resolution Enhancement	None	@PJL SET RET
PAGEPROTECT	Page protection (prevents Error 21)	None	@PJL SET PAGEPROTECT
RESOLUTION	Print resolution	None	@PJL SET RESOLUTION
PERSONALITY	Default printer language (PCL, PostScript, etc.)	None	@PJL SET PERSONALITY**
TIMEOUT	I/O timeout duration	None	@PJL SET TIMEOUT **
FONT SOURCE	Default PCL font source	Font selection	@PJL SET FONTSOURCE *
FONT NUMBER	Default PCL font number	Font selection	@PJL SET FONTNUMBER *
PITCH	PCL font pitch	$\text{E}_c(s\#H$	@PJL SET LPARM:PCL PITCH*
PTSIZE	PCL point size	$\text{E}_c(s\#V$	@PJL SET LPARM:PCL PTSIZE*
SYMSET	PCL symbol set	$\text{E}_c(ID$	@PJL SET LPARM:PCL SYMSET*

* These PJL commands should only be used by spoolers and printer utilities. For other applications, use the PCL command to modify the feature.

** Personality and timeout do not affect printed output. Only use these commands when it is necessary to modify the timeout or default personality.

7) Send the print data.

8) Reset the printer at the end of the job using E_cE followed by the UEL command (as shown here): $\text{E}_c\text{E}\%?-12345\text{X}$.

The following two examples demonstrate job setup for the LaserJet 4 printer. The first example uses status readback and the second one does not.

**Example:
Job Setup for the
LaserJet 4 Printer
Using Status
Readback**

The following example demonstrates how an application can set the LaserJet 4 printer to a desired state. This application uses the PJJ INQUIRE command to request status information and the PJJ SET command to modify some printer settings.

PJJ COMMANDS

$\text{E}_c\%?-12345\text{X}@PJJ <\text{CR}><\text{LF}>$

@PJJ INQUIRE COPIES <\text{CR}><\text{LF}>
@PJJ INQUIRE PAPER <\text{CR}><\text{LF}>
@PJJ INQUIRE ORIENTATION <\text{CR}><\text{LF}>
@PJJ INQUIRE FORMLINES <\text{CR}><\text{LF}>
@PJJ INQUIRE MANUALFEED <\text{CR}><\text{LF}>
@PJJ INQUIRE RET <\text{CR}><\text{LF}>
@PJJ INQUIRE PAGEPROTECT <\text{CR}><\text{LF}>
@PJJ INQUIRE RESOLUTION <\text{CR}><\text{LF}>

@PJJ SET RESOLUTION = 600 <\text{CR}><\text{LF}>
@PJJ SET PAGEPROTECT = ON <\text{CR}><\text{LF}>

Universal Exit Language (UEL) command that acts as a reset and gives control to PJJ. Notice that the X in the UEL command is immediately followed by the @ symbol of the following PJJ command or PJJ command prefix (as shown here).

Send PJJ INQUIRE commands to request the current status of environment variables. Within a few seconds, the status is returned to the host computer. The application can then compare the returned status with the desired values for each variable.

The returned status indicates that several variables needed modification in order to be set as desired. In this example, status readback indicates that just the RESOLUTION, PAGEPROTECT, COPIES, PAPER,

and ORIENTATION values needed to be modified. Use the PJJ SET command to set those variables that cannot be set using PCL (PAGE-PROTECT and RESOLUTION). Later in the setup, PCL commands will be used to set the remaining variables (COPIES, PAPER (page size), ORIENTATION).

@PJJ ENTER LANGUAGE = PCL <CR><LF>

Enter the PCL printer language.

JOB SETUP COMMANDS

E_cE

Reset printer.

E_c&l1X

Set the number of copies to 1.

PAGE SETUP COMMANDS

E_c&l1H

Select paper tray as paper source.

E_c&l3A

Select legal-size paper.

E_c&l1O

Set the orientation to landscape.

E_c&l8C

Set the VMI to 6 lines per inch.

PRINT DATA

E_c(9U

Set the symbol set to Windows.

E_c(s1p12v0s0b4148T

Select proportional 12-point, upright, medium, Univers font.

PRINT DATA

Send print data.

E_cE

Reset printer at end of PCL job.

PJJ COMMAND

E_c%-12345X

Universal Exit Language (UEL) command ends the job and returns control to PJJ.

Note



UEL commands and resets should be used at the beginning and end of the job, but they are not recommended anywhere else in the job. Besides resetting features, these commands eject the current page if they are received after printable data.

Example: Job Setup for the LaserJet 4 Printer Without Status Readback

The following example demonstrates how an application can set a LaserJet 4 printer to a desired state. Since this example uses no status readback, we assume that no features are already set as desired. Consequently, all features that affect printed output must be set by the application.

PJL COMMANDS

```
Esc%-12345X@PJL <CR><LF>
```

```
@PJL SET RET = ON <CR><LF>  
@PJL SET PAGEPROTECT = ON <CR><LF>  
@PJL SET RESOLUTION = 600 <CR><LF>
```

```
@PJL ENTER LANGUAGE = PCL <CR><LF>
```

The Universal Exit Language (UEL) command acts as a reset and gives control to PJL. The UEL command must be followed immediately by a PJL command or the PJL command prefix (as shown here), leaving no spaces between the X that terminates the UEL command and the @ that begins the PJL command.

Since there is no status readback, assume that all features need to be set as desired. Use the PJL SET command to set those variables that cannot be set using PCL (RET, PAGEPROTECT, RESOLUTION). Use PCL to set the remaining variables (copies, page size, orientation, text length and paper source).

Enter the PCL printer language.

JOB SETUP COMMANDS

ⒺE	Reset printer.
Ⓔ&l1X	Set the number of copies to 1.

PAGE SETUP COMMANDS

Ⓔ&l1H	Select paper tray as paper source.
Ⓔ&l3A	Select legal-size paper.
Ⓔ&l1O	Set the orientation to landscape.
Ⓔ&l8C	Set the VMI to 6 lines per inch.
Ⓔ&l66F	Set the text length to 66 lines.

PRINT DATA

Ⓔ(9U	Set the symbol set to Windows.
Ⓔ(s1p12v0s0b4119T	Select 12-point, upright, medium, CG Century Schoolbook font.
PRINT DATA	Send print data.
ⒺE	Reset printer at end of PCL job.

PJL COMMAND

Ⓔ%-12345X	Universal Exit Language (UEL) command ends the job and returns control to PJL.
-----------	--

Note



UEL commands and resets should be used at the beginning and end of the job, but they are not recommended anywhere else in the job. Besides resetting features, these commands eject the current page if received after printable data.

Making Copies of the Job

Setting the number of copies is a job setup function since it affects the entire print job. There are three methods of printing multiple copies of a print job:

- Sending the number of copies command using `Ec&l#X`
- Sending the PJL SET COPIES command
- Sending the job the desired number of times

An analysis of each option follows:

Sending the number of copies command (`Ec&l#X`)—using this command, like using the PJL SET command, is easy for the programmer to implement and prints multiple copies quickly. However, the printed output is not collated. For example, sending a software request for five copies (`Ec&l5X`) of a three-page job would print five copies of page one followed by five copies of page two followed by five copies of page three. Applications that generally take a longer time to print, as do many graphics applications, are better suited for using the *number of copies* command for printing multiple copies.

Sending the PJL SET COPIES command—this method provides the same uncollated results as the PCL number of copies command. However, the PJL SET COPIES command is recommended for spooling applications, not for most applications that generate print data (spreadsheets, graphics programs, etc.). For most applications that generate print data, use the PCL number of copies command instead. (To create spooling applications for the LaserJet 4 printer, consult the *PJL Technical Reference Manual*.)

Sending the Print Job Multiple Times—To print collated copies, your application should print the output file the required number of times. This option provides correct-order output to the user, but is extremely time-consuming and memory-intensive for the CPU, is slower for the user, and is I/O-intensive. This option is recommended for less I/O-intensive applications, such as word processing, where users frequently print multiple-page documents.

LaserJet IIISi Job Setup Commands

The HP LaserJet IIISi printer has two job setup commands that are not supported by any of the other PCL 5 LaserJet printers. These commands are:

- Job Offset (also in LaserJet 500 PLUS, LaserJet 2000)
- Output Bin Selection

These features are discussed below, with examples that demonstrate their use.

LaserJet IIISi Job Offset

In order to differentiate multiple print jobs in the standard face-down output stacker, the LaserJet IIISi printer can be instructed programmatically to physically offset print jobs using the job offset (`^c&l1T`) command. Receipt of the job offset command causes the printer exit rollers to shift 1/2-inch relative to the previous job. This feature makes it easier for users to locate their output in the output bin.

Note



The HP LaserJet 500 PLUS and the LaserJet 2000 printers also support the job offset command. The LaserJet 500 PLUS printer physically offsets jobs as does the LaserJet IIISi, and the LaserJet 2000 printer inserts a job separation sheet (without a physical offset).

Because this is a PCL printer language command, PostScript jobs should not contain the command. However, printers with the PostScript option can support this feature via the control panel. When the “JOB OFFSET” variable is configured to “ON”, the exit rollers are toggled 1/2-inch every time a CTRL-D (or `^c%-12345X`) is encountered.

Note



The control panel job offset setting has no effect on PCL jobs.

In your application, job offset should be a user-selectable option. When selected, the job offset command should be sent

at the beginning of each job, following the printer reset command as shown in the following example.

**Example:
LaserJet IIISi
Job Offset**

```
Ec%-12345X@PJL <CR><LF>  
@PJL ENTER LANGUAGE = PCL <CR><LF>
```

Send the UEL Command to give control to PJL. (*Note that there is no space between the X at the end of this command and the @ that begins the next command.*) Send the PJL ENTER command to enter PCL.

EcE

Reset the printer.

Ec&l1T

Set the job offset so that it toggles between each PCL print job.

PRINT DATA

Send PCL print data to the printer.

EcEEc%-12345X

Reset the printer and send the UEL Command to end the job.

**LaserJet IIISi
Output Bin Selection**

The LaserJet IIISi printer supports two different output bin selections. The default output bin is the top, 500-sheet, face-down, correct-order output bin. The lower 50-sheet, face-up, reverse-order output bin is also programmatically selectable (send Ec&l1G to select the upper bin or Ec&l2G to select the lower bin).

Selection of the lower output bin may be appropriate in several instances. In anticipation of accessory paper handling devices, such as stackers, collators, and other devices, lower output bin selection allows users to take advantage of this advanced functionality.

Unlike some of the other LaserJet printer models, the lower output bin may not be selected by manually operating an output bin selector knob. The lower output bin can be selected only using the control panel or with printer commands. Consider providing output bin selection as a user-selectable option for your customers.

Memory Usage

Page Protection (Error 21)

In the process of preparing a page for printing, LaserJet printers process the print data in sectioned bands of the page. Once the print engine starts, the page data for that page must be rasterized at a rate sufficient to keep up with the print engine. If the formatting for a particular page is very complex, the image processor is sometimes unable to keep up with the speed of the print engine and an Error 21 (print overrun) occurs.

In the PCL 5 LaserJet printers, print overruns can be avoided by enabling PAGE PROTECT from the control panel, or, for the LaserJet 4 printer, using the PJP SET PAGEPROTECT command. (Some PCL 5 printers may require additional memory to enable this feature. See your printer user's manual.) With the *page protection* mode selected, the printer rasterizes the whole page before it begins printing it, avoiding any possibility of a print overrun.

Note



PCL 4 LaserJet printers do not have this feature. For these printers, the only way to correct an Error 21 is to reduce the complexity of the page being formatted.

HP-GL/2 data is inherently complex to rasterize, so page protection is recommended for HP-GL/2 jobs. If your application uses HP-GL/2 commands, it is a good idea to recommend the page protection feature of the PCL 5 LaserJet printers in your user documentation and through your support services. However, for the LaserJet 4 printer, even complex HP-GL/2 pages can print without page protection. Recommend page protection for the LaserJet 4 printer only for those cases when Error 21 occurs.

Page Protection for the LaserJet 4 Printer

Unlike the other PCL 5 LaserJet printers, which require setting page protection from the control panel, the LaserJet 4 printer also allows you to enable or disable it using the PJJ SET or PJJ DEFAULT commands.

The LaserJet 4 printer allows you to set page protection for different page sizes, and to set resolution to 600 dpi regardless of the installed memory. However, before passing the incoming job to the page description language, the printer must analyze the memory requirements to see if it can accommodate the requested page protection and resolution settings. If there is not sufficient memory to satisfy the request, the printer reconfigures itself to allow the job to print, and posts a clearable warning message indicating the change. Once the job has been passed to the page description language, resolution and page protection cannot be reconfigured until a job boundary has been detected.

In situations where there is not enough memory to print a large raster image, the printer first tries to compress the image without losing image data. If there is still not enough memory, the printer uses a compression technique which actually eliminates some of the image data. This technique usually allows the image to print acceptably, even though some of the image data is lost. (If this happens, the printer displays “W5 Image Adapt” on the control panel and sends an unsolicited status message to the host.)

PJJ and Perishable Data

PJJ allows applications to control many printer features that were previously only controllable from the control panel. Note, however, that some PJJ commands reconfigure the printer’s memory, erasing perishable data such as downloaded fonts, macros, and graphics. The following commands cause perishable data to be erased if a change from the current status occurs:

- @PJJ ENTER LANGUAGE
- @PJJ SET PAGEPROTECT
- @PJJ SET RESOLUTION

Freeing Memory for Fonts, Macros, and Graphics

In some applications, users download a large number of fonts and macros to the printer. This is especially true in a network situation. Since the printer has a limited amount of memory, users may find that there isn't enough memory to download their entire job, causing an Error 20 (memory overflow). When an out-of-memory condition occurs, the job will not print as the user expects. (Users find this situation frustrating since they must either clear the printer's memory and re-issue the job or add more memory to their printer.)

Note



If the fonts and macros are downloaded with a “permanent” status, then an `ESC E` won't clear them from memory; however, *all* fonts and macros are erased from memory when the power is turned off (or when the page protection status, resolution, or printer language is changed).

Your application can provide a simple way to clear permanently downloaded fonts and/or macros from the printer. Hewlett-Packard suggests a menu selection in software applications for clearing all permanently downloaded fonts or macros from memory. HP also recommends that the user's request for clearing the fonts or macros be answered with a response asking them if they are sure they want to clear all fonts and macros from the printer.

To clear all permanently downloaded fonts from memory, send the `ESC *cOF` command following the initial `ESC E` reset command (but before any printable characters). To delete all macros, send the `ESC &f6X` command.

Example: Erasing Fonts and Macros

This example shows how to erase downloadable fonts and macros as part of the job setup process:

In this scenario, the user attempts to print a job and finds an Error 20 preventing the job from printing successfully. The application offers a menu feature that allows the user to erase all downloaded fonts and macros. The user enters YES in both menu fields that indicate the desire to erase fonts and macros. The application, responding to the user's request, then asks if the user is sure of his/her intent and warns that all downloaded fonts and macros will now be erased. The user responds YES. The application then sends the following commands to the printer.

Note



This example shows how to avoid a memory overflow condition when the overflow is the result of too much previously downloaded data. In conditions where the memory overflow is due to the current job downloading too much data, the following example will still not prevent memory overflow.

<code>EcE</code>	Reset the printer. The following commands override any control panel settings that could be set differently than desired (number of copies, paper source, page size, orientation, VMI, symbol set and font).
<code>Ec&l2X</code>	Set the number of copies to 2.
<code>Ec&l1H</code>	Select paper source.
<code>Ec&l3A</code>	Select legal-size paper (8.5" x 14").
<code>Ec&l1O</code>	Set the orientation to landscape.
<code>Ec&l6C</code>	Set the vertical motion index (VMI) to 8 lines per inch.
<code>Ec(8U</code>	Set the symbol set to Roman-8.

<code>ⒺⒸⓈ1Ⓟ14Ⓢ0Ⓢ0Ⓟ4101Ⓣ</code>	Select a 14-point, upright, medium, CG Times font.
<code>ⒺⒸ&Ⓛ1Ⓢ42Ⓣ</code>	Set top margin to 1 line and text length to 42 lines.
<code>ⒺⒸ*Ⓒ0Ⓣ</code>	Delete all soft fonts per user request.
<code>ⒺⒸ&Ⓣ6Ⓝ</code>	Delete all macros per user request.
<code>ⓅⓇⓂⓂ ⓉⓂⓂⓂ</code>	Send print data.
<code>ⒺⒸⒺ</code>	Reset printer at end of print job.

Chapter Summary

Here are some hints that summarize the main points covered in this chapter:

- For PDL applications, use the PDL UEL command to begin and end each job. The UEL command at the beginning of the job should be followed immediately by the @ of the next PDL command, for example:

```
?%-12345X@PDL Comment **Job #1** <CR><LF>
```

- For PDL applications, use the PDL ENTER LANGUAGE = PCL command immediately before the (ⒺⒸⒺ) that precedes the PCL print data. (See the LaserJet 4 and LaserJet IIISi examples in this chapter.)
- Send a programmatic reset (ⒺⒸⒺ) at the beginning and end of PCL jobs, as shown in all of the examples in this manual.
- Send printer commands to override any possibility of unwanted feature settings due to the control panel (but don't send commands that override control panel features unless your software allows the user to select those features).

- For PJJ printers, only use PJJ commands to set features that cannot be set using PCL, such as PAGEPROTECT, RET, and RESOLUTION. The PERSONALITY and TIMEOUT variables, since they do not affect the printed output, should not be set unless you specifically need to change the default personality or timeout duration.
- Remember that a language switch, change of page protection status, or change of resolution erases perishable data.
- Consider offering your users the option of erasing permanently downloaded fonts and/or macros by way of a menu selection.

Note



For a good example of a job setup string, see the *General Print Job Initialization* discussion in Chapter 13.

Contents

Introduction	3-1
Selecting a Paper Source	3-1
Selecting a Physical Page Size	3-2
The Page Size Command	3-2
The Page Length Command	3-3
The Actual Printable Area	3-4
Changing Orientation	3-9
Print Direction	3-11
Managing the Text Area	3-13
Controlling Top Margin	3-15
Establishing the Bottom Margin	3-16
Perforation Skip for Print-and-Space Formatting	3-17
Controlling the Left Margin	3-19
Controlling Right Margin	3-20
Changing Character Spacing (HMI)	3-21
Modifying Line Spacing (VMI)	3-22

Introduction

Page setup is a part of the job setup process and defines the structure around which pages are formatted. While job control commands are usually sent at the beginning of the print file, page control commands may be associated with a single page or with groups of pages. Page control commands determine the selection of the paper source, paper size, orientation, margins, and text spacing. This chapter describes how to address page control in your application.

If you are not yet familiar with the concepts of *physical page*, *logical page*, and *printable area*, please look over the section titled “The Page” in the *PCL 5 Printer Language Technical Reference Manual*. It provides a good visual explanation of these terms and will prepare you for the discussion in this chapter.

Selecting a Paper Source

The paper source command allows your job to select paper from either the paper trays, the manual feed input, or from an envelope feeder. The following example shows a potential situation using this command:

Example: Selecting a Paper Source

This example demonstrates selecting paper from the manual feed input for the first page of a print job, and then from the paper cassette for the remaining pages. It is assumed that the proper job setup commands have already been sent to the printer.

Background: Using the application software, the user specifies that the first page of his print job will be fed manually and the remaining pages fed from the upper cassette. When the user begins to print the job, the printer (and possibly the software) prompts the user to insert paper in the manual feed slot. (Although this will work automatically on some PCL 5 printers by putting a sheet of paper in the manual feed slot before printing, using the manual feed com-

mand as shown below eliminates potential problems caused when printing with different sizes of paper.)

$\text{\textcircled{E}}\H$	Select the manual feed input as the paper source.
PRINT DATA	Send print data for the letterhead page.
$\text{\textcircled{E}}\H$	Feed paper from the paper cassette (the tray location varies with the printer model).
PRINT DATA	Send print data for the remaining pages.

Selecting a Physical Page Size

There are two methods of selecting the physical page size: sending the *page size* command or sending the *page length* command.

The Page Size Command

The *page size* command ($\text{\textcircled{E}}\&#A$) designates the page size, which in turn defines the size of the default logical page. This command is recommended instead of the page length command because it explicitly selects physical page size and it allows for the selection of various sizes of envelopes (which cannot be selected with the page length command). It also provides a quick way to specify the physical page size without requiring a VMI command. (The page size command is not recognized by the LaserJet, LaserJet PLUS, and LaserJet 500 PLUS printers.)

Note



When the page size command is used, it must be transmitted at the beginning of the page prior to any printable data; otherwise, when the printer receives the command, the current page is closed and printed.

Example: Using the Page Size Command

This example sets the physical page size using the page size command (it is assumed that the proper job control commands have already been sent to the printer, as explained in Chapter 2):

`Ⓔ&l3A`

Select legal page size. The logical page length is automatically defaulted to 14 inches (without sending the page length command).

`Ⓔ&l00`

Select portrait orientation.

The Page Length Command

The *page length* command (`Ⓔ&l#P`) sets the size of the logical page length in number of lines (at the currently active VMI), which automatically defines the physical page size.

If you do wish to use the page length command, it should be preceded by a command that sets the orientation to portrait and a command that sets the VMI (line spacing).

In landscape orientation, page lengths for legal and letter-size papers are identical. To select legal-size paper in landscape orientation using the page length command, the user should switch to portrait orientation, set the page length for legal paper, and then return to landscape orientation (see the following example). This situation can be avoided by using the *page size* command instead of the *page length* command.

Note



When the page length command is used, it must be transmitted at the beginning of the page prior to any printable data; otherwise, when the printer receives the command, the current page is closed and printed.

The example below demonstrates how the page length command should be used:

Example: Using the Page Length Command

This example shows the commands that are necessary to select a legal-size landscape page using the page length command (without these commands, letter-size paper would be selected). (It is assumed that the proper job control commands have already been sent to the printer, as explained in Chapter 2):

<code>Esc&tl00</code>	Set the logical page orientation to portrait.
<code>Esc&tl8C</code>	Designate a VMI of 8 (6 LPI).
<code>Esc&tl84P</code>	Set the logical page length to 84 lines to select legal paper (14 inches at 6 LPI = 84 lines).
<code>Esc&tl10</code>	Set the logical page orientation to landscape.

The Actual Printable Area

The actual printable area on the physical page is determined by the physical limits of the printer and the logical page. The area where you can move the cursor and print is approximately the size of the physical page, minus a small distance on all edges of the paper.

Figure 3-1 shows the difference between the *physical page*, the *logical page*, and the *printable area*. A short explanation of each may help in understanding the difference between these areas:

- The *physical page* is simply the entire area occupied by the page on which you are printing, the actual paper size.
- The *logical page* defines the entire addressable area on the page.

- The *printable area* defines the area on the page where the printer is physically capable of printing a dot.

These brief definitions are accurate, but they need some elaboration to make the concept clear. Notice that the logical page (addressable area) extends all the way to the top and bottom of the physical page. This entire area is addressable, but the areas within 50 dots of the top and bottom of the page are not printable (the unprintable region). Although it is possible to address portions of the unprintable region, attempting to print there will result in data loss.

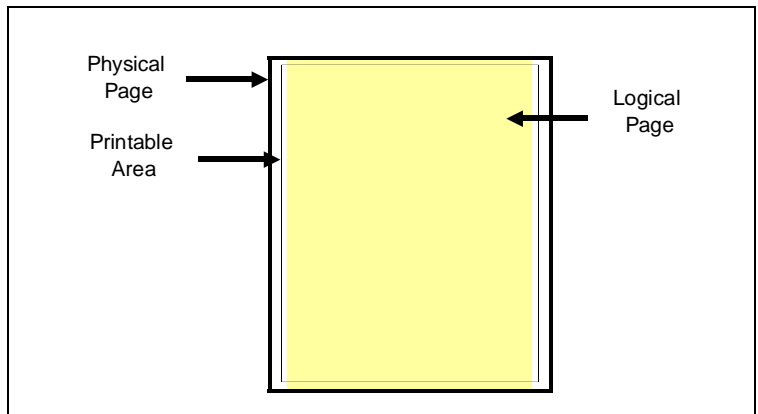


Figure 3-1. The Logical Page and Printable Area

To help further explain printable area, notice that on the left and right sides of the page, the logical page lies within the printable area. That means that you can move the cursor only up to the edges of the logical page, not the edge of the printable area. The cursor can be moved to the edges of the logical page (the left- and right-most addressable limits), but any character that begins printing within the logical page boundary and extends beyond it will not be clipped until it reaches the edge of the printable area. In other words, if part of a character extends to the left or right of the logical page boundary, it will still be printed up to the printable area boundary, as long as part of the character lies within the logical page.

To summarize these points:

- The cursor can be moved anywhere on the logical page, but data will be lost if printing is attempted in the area within the top 50 dots and bottom 50 dots on the page.
- On the *left* side of the page, the cursor can be moved to the edge of the logical page and text or graphics can be printed there. If a character is started within the logical page but extends beyond the left margin (for example, backward-slanted italics), the character will not be clipped until it reaches beyond the left *printable area boundary*. (Graphics will *not* extend beyond the left edge of the left logical page boundary.)
- On the *right* side of the page, the cursor can be moved to the edge of the logical page. If a line of text approaches the right edge of the logical page, and a character extends beyond the right *logical page boundary*, it will not be clipped until it reaches the right *printable area boundary*. This also holds true for raster graphics and image fill (the Print Model); rules, however, are clipped at the logical page boundary.

Note



Chapter 2 of the *PCL 5 Printer Language Technical Reference Manual* contains a table that lists the sizes of the logical page and printable area boundaries for each page size.

Example: Seeing the Printable Limits

Here is a simple exercise that prints a gray shading pattern to the edges of the printable area on the top and bottom, and to the edges of the logical page on the left and right; this example also prints text so that you can see how characters are clipped in the unprintable region. If you try this example, you will be able to see exactly how close you can print to the edges of the page for a particular paper size.

<code>^cE</code>	Reset the printer.
<code>^c&l0E</code>	Set top margin to 0.
<code>^c&l0L</code>	Disable the perforation skip mode.
<code>^c*p0x0Y</code>	Move the cursor to the top left corner of the logical page (0,0).
<code>^c*c2700a4200B</code>	Specify a rectangular area to be filled with a shade of gray. A size of 9 inches by 14 inches is specified so that the example will work with any of the standard paper sizes, including legal. The command specifies the rectangle size in dots (9 inches x 300 dots/inch = 2700 dots; 14 inches x 300 dots/inch = 4200 dots).
<code>^c*c15G</code>	Specify an area fill ID (15% gray).
<code>^c*c2P</code>	Fill the specified rectangular area with a shaded fill pattern.
<code>^c(0U</code>	Select the ASCII symbol set.
<code>^c(s1p65v1s3b4101T</code>	Select a 65-point, bold, italic, CG Times font.
<code>^c*p188Y</code>	Move the cursor down 5/8-inch from the top of the page so you can see how the following text will be clipped at the top.
This is a test and it clips.	Send text that will clip at the top and the right edges of the page. When you print this, notice how the

last character is clipped at the right printable area boundary (beyond the logical page boundary indicated by the shading on the printout).

EscE

Reset to end job and eject the page.

Keep in mind that, although the size of the printable area is the same for every printer of the same model, the distance

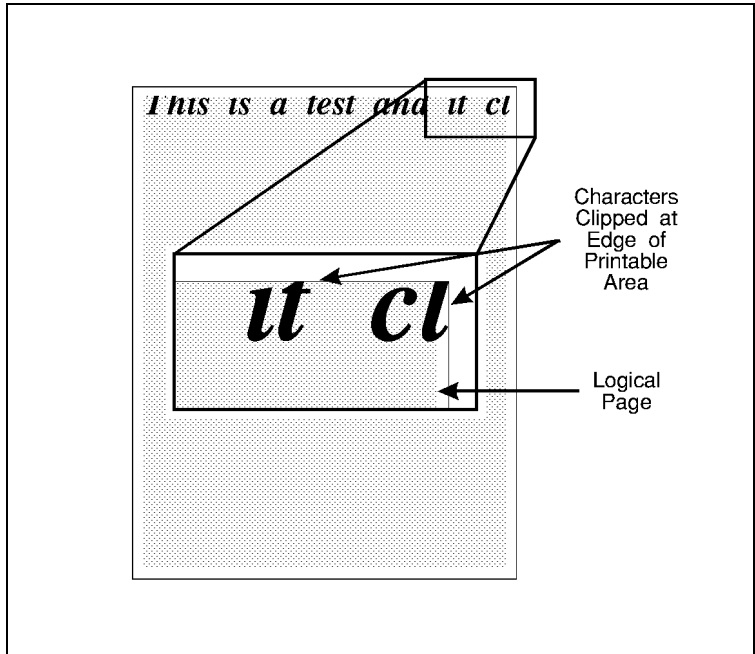


Figure 3-2. Seeing the Printable Limits

Note



between the edge of the physical page and the printable area may vary slightly from printer to printer due to paper registration tolerances.

Changing Orientation

To provide printing flexibility, the PCL 5 LaserJet printers support printing in four orientations:

- Portrait
- Landscape
- Reverse Portrait
- Reverse Landscape

When the orientation command is sent to the printer, the printer defaults the following:

- Logical page
- Print direction
- Page length
- Text length
- Top margin, left margin, right margin
- HMI and VMI
- Auto macro overlay disabled

When the orientation command is sent, the orientation of fonts is automatically changed to match that of the logical page. The orientation of raster graphics is also rotated to match the new orientation, if the *raster* graphics presentation mode is set to 0 ($E_C^*r\cancel{F}$).

Note



The default HP-GL/2 coordinate system rotates with the logical page orientation, but can be modified using HP-GL/2 directives (the RO command); this orientation interaction is discussed in Chapter 10 (see Figure 10-3).

If the printer has received any printable data prior to receiving an orientation command, the page is immediately closed and printed and a new page is opened with the selected orientation. Because the printer ejects any print data and defaults the features listed above, the orientation command should always be sent at the beginning of a page and

followed by commands that set those features for which a non-default value is desired.



Only one orientation is allowed per logical page, however multiple print directions may be used (see the following *Print Direction* discussion).

Example: Changing Orientation

This example demonstrates a print job containing both portrait and landscape pages; the first page is a letter-size page in portrait orientation and the second page is a legal-size page in landscape orientation. (The example is simplified in that it assumes the default values for print direction, text length, margins, and line spacing.)

<code>ⒺE</code>	Reset the printer.
<code>Ⓔ&Ⓛ1X</code>	Set the number of copies to 1.
<code>Ⓔ&Ⓛ2A</code>	Select a letter-size page.
<code>Ⓔ&ⓁⓀO</code>	Select portrait orientation.
PRINT DATA	Send print data for the first page. (portrait)
<code>Ⓔ&Ⓛ3A</code>	Select a legal-size page.
<code>Ⓔ&Ⓛ1O</code>	Select landscape orientation.
PRINT DATA	Send print data for the second page (landscape).
<code>ⒺE</code>	Reset printer at end of job.



The example above demonstrates the efficiency of the *page size* command. If the *page length* command was used instead, several more commands would have been necessary to accomplish the same goal.

Print Direction

The *print direction* command allows you to switch the print direction without ejecting a page. With this command, you can print more than one direction of text and graphics on the same page. The print direction can be rotated counter-clockwise 0, 90, 180, or 270 degrees in relation to the current orientation of the logical page (which has a print direction of 0 degrees).

An important point to remember when changing print direction is that the margins are *translated* with the print direction. In other words, if the print direction changes by 90 degrees, the left margin becomes the new top margin, the top margin becomes the new right margin, etc..

The CAP (current active position) stack, a list of previous cursor positions created using the *push/pop cursor position* command, is also translated. The positions in the CAP stack are translated so that they remain at the same physical location on the page. Translating the CAP stack enables you to store and restore exact physical page positions, regardless of the current print direction.

Note



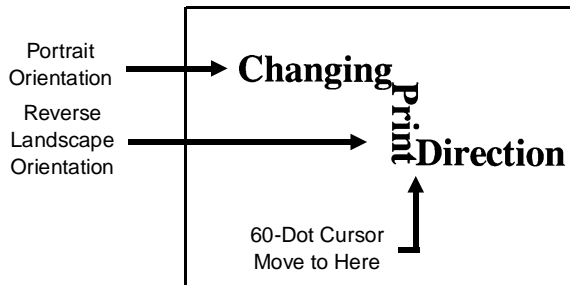
Changing print direction defaults the HMI.

Example: Two Print Directions on a Page

This example shows how to print text in two directions on the same page using the *print direction* command:

<code>^cE</code>	Reset printer
<code>^c&l00</code>	Select portrait orientation. This also sets the print direction to its default value, 0 degrees.
<code>^c(0U</code>	Select the ASCII symbol set
<code>^c(s1p24vs3b4101T</code>	Select a bold, 24-point CG Times font. The CG Times typeface is one of the PCL 5 LaserJet printer's standard scalable typefaces. After

the print data is received, the printer scales the selected typeface to 24 point. (To save memory and time, only those characters that are to be printed are scaled.)



Changing	Print "Changing" in the portrait orientation
$\text{E}_c\&a270P$	Set the print direction to 270 degrees (reverse landscape)
Print	Print the word "Print" in the reverse landscape orientation.
$\text{E}_c\&a0P$	Set the print direction to 0 degrees (portrait)
$\text{E}_c*p+60X$	Move the cursor 60 dots to the right. Since changing print direction doesn't change the physical cursor position, the cursor had to be moved to the right to avoid printing on top of the word "Print".
Direction	Print the word "Direction".

Managing the Text Area

Once you have selected a logical page orientation, you can define the text area by sending the top, left, and right margin commands, as well as sending the text length and perforation skip mode commands.

Note



There is no bottom margin command. The bottom margin is automatically calculated by subtracting the text length and top margin (at the current VMI) from the logical page length.

As shown in Figure 3-3, the 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 margin distance from the edge of the physical page must be calculated by adding the selected margin distance to the distance between the edge of the physical page and the edge of the logical page.

Note



Consult the *Logical Page and Printable Area Boundaries* table in Chapter 2 of the *PCL 5 Printer Language Technical Reference Manual*. It lists values for each orientation and page size.

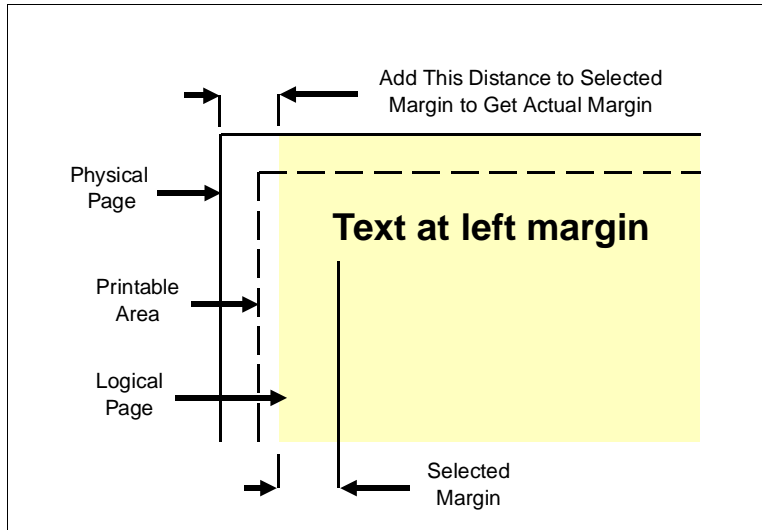


Figure 3-3. Adjusting the Margin Distance

For example, to set a left margin of 1.5 inches on a portrait page, you would set the left margin command for the value that would give you 1.5 inches minus the 75-dot (.25-inch) distance between the edge of the physical page and the logical page boundary (1.5 inches minus .25 inches or 1.25 inches). That equates to about 13 columns (column 12) at 10 characters per inch, instead of 15 columns.

Note



The text area affects the placement of text and defines the default HP-GL/2 picture frame. Raster graphics and cursor movement commands are independent of the text area. Using cursor positioning, text can be printed outside the text area.

Controlling Top Margin

The top margin determines the distance between the top of the logical page and the start of text. The default top margin is 3 lines at 6 lines per inch (.5 inches), however the top margin can be set to values from 0 to the size of the logical page.

The top margin is specified in number of lines, and the distance between these lines is determined by the current line spacing. The default cursor position is calculated as follows:

Top margin in inches + (.75 x VMI in inches) = position of first line of text

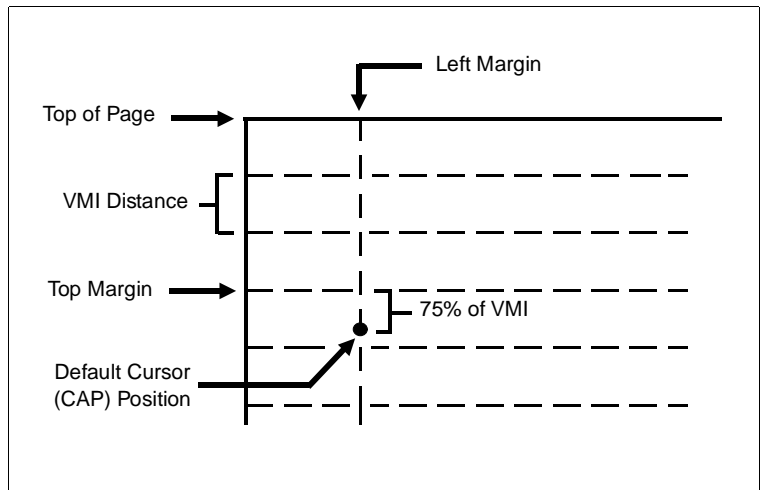


Figure 3-4. The Default Cursor Position

For example, a top margin command of $E_c \& l 6 E$ sets the printer to a top margin of 1 inch (6 lines x 6LPI = 1 inch; the VMI for 6 LPI = 8), resulting in a cursor position of 1.12 inches from the top of the physical page:

1 inch top margin + (.75 x (8/48 inches)) = cursor position of 1.12 inches

Note



The top margin command should precede the text length command since the top margin command resets text length.

Establishing the Bottom Margin

Setting the *text length* ($E_c \& \ell \# F$) provides a method for setting a bottom margin and controls the number of lines printed per page. The bottom margin coincides with the beginning of the perforation skip region. Although there is not an actual perforation, the perforation skip region refers to the distance between the bottom of the text area on one page to the top of the text area on the next page.

The largest acceptable text length value is equal to the logical page length minus the top margin; for example, for letter-size paper with a 1/3-inch top margin ($VMI = 8$), the largest text length is 64 lines (66 lines [11 inches] – 2 lines [1/3 inch]). However, since the top margin distance plus the text length establishes the bottom margin, an acceptable bottom margin can allow printing in the unprintable region, potentially resulting in clipping (which is *not* acceptable). Using the above example, setting the top margin to 2 and specifying a text length of 64 results in data clipping (assuming a line spacing of 6 LPI and a 12-point font size). Likewise, specifying a top margin that is too small causes characters to be clipped at the top of the page.

To avoid losing data, the text area must not extend into the unprintable region. In other words, if the unprintable region is 50 dots at the bottom and 50 dots at the top of the page, the text length must be set so that text starts after and stops before reaching the 50-dot boundary.

Example: Avoiding the Unprintable Region

This example demonstrates a method of using the maximum printable area while preventing the possibility of losing data in the unprintable region at the bottom of the page.

<code>^c&l2A</code>	Designate letter-size paper.
<code>^c&lO</code>	Designate portrait orientation.
<code>^c&l8C</code>	Designate VMI of 8 (6 LPI).
<code>^c&l1E</code>	Set top margin to 1/6th of an inch.
<code>^c&l64F</code>	Set the text length to 64 lines. The resulting bottom margin is 1 line (66 lines for a full page - 1 line top margin - 64 lines text length = 1 line bottom margin).

Note



The line feed and half line feed commands are the only cursor movements restricted by the bottom margin. If a line feed or half line feed causes the cursor position to enter the perforation skip region, the current page is closed and printed. On the other hand, if a cursor movement command moves the cursor into the perforation skip region, the printer does *not* automatically eject a page.

Using the Perforation Skip Region for Print-and- Space Formatting

For simple print-and-space formatting, the text length can be used to limit the number of lines printed on a page. When a line feed or half line feed causes the cursor to extend into the perforation skip region, the printer closes and prints the current page. This method eliminates the need for the software to manage the number of lines before sending a form feed command.

Example: Print-and-Space Formatting

This example uses the perforation skip region for print-and-space formatting by setting the text length and then allowing the printer to automatically eject the page when the perforation skip region is entered:

<code>ⒺE</code>	Reset the printer.
<code>Ⓔ&ℓ2A</code>	Set page size to letter (this automatically defaults the logical page to 66 lines at the standard VMI [6 LPI]).
<code>Ⓔ&ℓ0O</code>	Set the orientation to portrait.
<code>Ⓔ&ℓ8C</code>	Set the VMI to 8 (6 LPI).
<code>Ⓔ&ℓ50F</code>	Set the text length to 50 lines, leaving a top margin of 3 lines and a bottom margin of 13 lines. (Reset enables the perforation skip mode and sets the top margin to 1/2 inch (3 lines); 66 lines [page length] - 3 lines [top margin] - 50 lines [text length] = 13 lines bottom margin).
TEXT FOR THE FIRST LINE . . .	Send text to the printer.
CR-LF	Send a carriage return and line feed to move to the left margin on the next line.
TEXT FOR THE 50th LINE . . .	Send text to the printer.
CR-LF	Send a carriage return and line feed to move to the left margin on the next line. The line feed command moves the cursor into the perforation skip region (bottom margin), causing the printer to print and eject the current page, and then begins formatting the following page.

Controlling the Left Margin

The left margin is specified in columns and the distance between these columns is determined by the current HMI (horizontal motion index). For fixed-pitch fonts, the default HMI value corresponds to the print pitch; for proportionally spaced fonts, the default HMI corresponds to the width of the space character. HMI can also be adjusted to the desired value using `^c&k#H`, where # equals the number of 1/120-inch increments per column (see “Changing Character Spacing (HMI)” later in this chapter).

Note



The first column is column 0. The margin value indicates the column in which the first character is printed. For example, setting the left margin with the `^c&a5L` command sets the left margin to the 6th column. In this case, the printer spaces over 5 columns and begins printing in the 6th column (leaving a 5-column margin). The actual margin distance is dependent on the current HMI.

Example: Setting the Left Margin

This example shows how to set the left margin to 1 inch from the edge of the logical page (which is 1 inch plus 75 dots from the edge of the physical page). It is assumed that the printer environment is properly established as described in Chapter 2.

<code>^c&l2A</code>	Select a letter-size page. This defaults the logical page size to 66 lines (11 inches x 6 LPI = 66 lines).
<code>^c&l0O</code>	Select portrait logical page orientation.
<code>^c&l8C</code>	Designate a VMI of 8 (6 LPI).
<code>^c&k12H</code>	Set horizontal motion index (HMI) to 10 characters per inch.
<code>^c&l10L</code>	Set left margin to one inch, based on the current HMI. Remember, the first column is column 0.
PRINT DATA	Place text.

Controlling Right Margin

Except in simple print-and-space applications, the right margin is not set, but is managed by the application. The application takes the user's desired right margin setting and calculates where to place the carriage return based on the total character and word spacing in each line of text.

Example: Controlling Right Margin

This example demonstrates a simple print-and-space application that effectively sets a right margin without using the right margin command. Instead, the right margin is controlled by the application using font spacing information (TFM data).

Background: The user has set the left and right margins to one inch using the software application's page formatting menu. Assuming letter-size paper, this determines a column-width of 6.5 inches.

<code>E_cE</code>	Reset the printer.
<code>E_c&l2A</code>	Select a letter (8-1/2 x 11) page size.
<code>E_c&l0O</code>	Designate portrait orientation.
<code>E_c&l8C</code>	Designate a VMI of 8 (6 LPI).
<code>E_c&a10L</code>	Set the left margin to one inch (remember that the first column is column 0 and that the default HMI is 10 characters per inch [depending on the default font].)
The quick brown fox jumped over the lazy dog...	Place text until the total width of character and word spaces equals 6.5 inches (see Chapter 6 for examples of how to obtain font spacing information).
<code>CR-LF</code>	Send a carriage return and line feed to move the cursor to the left margin on the next line.
More text on the next line...	Place text until the total width of character and word spaces equals 6.5 inches.

CR-LF

Send a carriage return and line feed to move the cursor to the left margin on the next line.

Variation: Instead of reading TFM data to determine the right margin, the application could use a fixed-pitch font and then count the number of characters per line until it reaches the value that equals 6.5 inches.

Changing Character Spacing (HMI)

The horizontal motion index (HMI) determines the width of columns used for setting margins, horizontal cursor positioning (column moves only), and the spacing between characters in a fixed-pitch font. As discussed in the *PCL 5 Printer Language Technical Reference Manual*, the HMI is set using the `? & k # H` command, where # equals the number of 1/120-inch increments per column. (For proportionally spaced fonts, the HMI affects only the width of the space control code for font spacing.)

Note



Designating font attributes defaults the HMI to that of the currently selected font. It is not necessary to send an overriding HMI unless a value other than the default is desired.

Example: Character Spacing

This example shows how to modify the character spacing of a fixed-space font from 10-pitch to 12-pitch:

This is 10 characters per inch.

This is 12 characters per inch.

<code>Ec(8U</code>	Select the Roman-8 symbol set.
<code>Ec(s0p10h12vsb3T</code>	Select the 10-pitch Courier font (10 characters per inch).
This is 10 characters per inch.	Send a line of 10-pitch text.
<code>Ec&k10H</code>	Set HMI to 10/120ths of an inch or 12 characters per inch.
This is 12 characters per inch.	Send 12-pitch text to the printer.

Modifying Line Spacing (VMI)

The line spacing can be modified using either the vertical motion index (VMI) or line spacing commands, or by changing the printer's front panel FORM= setting. The VMI command is recommended for most applications since it offers a greater degree of accuracy (VMI is specified in units of 1/48 inch and is valid to four decimal places).

Please keep the following points in mind when designating line spacing:

- Ensure that line spacing is to a known state, such as the factory default setting (VMI = 8; 6 LPI), prior to using the page length command. The page length command uses the current VMI to determine the length of the logical page, which in turn defines the physical page size.

- Ensure that line spacing is at a known value before setting the top margin. The top margin (in inches) is calculated as top margin (in lines) x (current VMI)/48. Therefore, the value of VMI is critical to the top margin setting.

Example: Job Control and Page Setup

This example shows a recommended way of setting up a job, including job control and page control commands. (For a good example of a job setup string, also see the *General Print Job Initialization* discussion in Chapter 13.)

Note



Jobs destined for PjL printers should also include the UEL command as the very first command, and the PjL ENTER command immediately before the initial $\text{E}c\text{E}$ (as described in Chapter 2).

$\text{E}c\text{E}$	Reset the printer.
$\text{E}c\&l1\text{X}$	Set number of copies to 1.
$\text{E}c\&l2\text{A}$	Set the page size to letter (8.5" x 11").
$\text{E}c\&l\emptyset\text{O}$	Set the orientation to portrait.
$\text{E}c\&l8\text{C}$	Designate a VMI of 8 (6 LPI).
$\text{E}c\&l2\text{E}$	Set top margin to 2 lines (1/3 inch).
$\text{E}c\&k12\text{H}$	Select an HMI of 12/120ths or 10 characters per inch.
$\text{E}c\&a1\emptyset\text{L}$	Set the left margin to 1 inch (a one-inch margin is actually one inch from the left edge of the logical page plus 1/4 inch. The 1/4 inch [75 dot] value is the distance between the left edge of the physical page and the logical page. This value was obtained from the <i>Logical Page and Printable Area Boundaries</i> table in

the *PCL 5 Printer Language Technical Reference Manual*.)

<code>Ec&tl61F</code>	Set the text length to 61 lines.
Text . . . Text	Print text until the total character and word spacing equals the desired right margin.
CR-LF	Move to the left margin on the next line.
Text . . . Text	Place text until the total character and word spacing equals the desired right margin.
CR-LF	Move to the left margin on the next line

Note



Not all page control commands need to be sent for each page. After the first page, page control commands should only be sent when a page control feature needs to be changed. Using page control commands on each page may cause an unwanted page eject if any printable data is in the buffer when the command is sent.

Contents

- Introduction 4-1
- Current Active Position (CAP) 4-1
 - CAP for Text. 4-1
 - CAP for Raster Graphics. 4-2
 - CAP for Vector Graphics 4-3
 - CAP for Rectangular Area Fill 4-4
- Print-and-Space Cursor Positioning 4-5
- Absolute vs. Relative Positioning 4-6
- Units of Movement 4-7
 - PCL Units vs. Decipoints vs. Columns/Rows 4-7
 - HP-GL/2 Plotter Units 4-10
- Saving the Cursor Position 4-10
- Positioning the Cursor at the Limits of the Page. . . . 4-11
- How the Paper Path Affects Cursor Placement 4-16

Introduction

Changing the current active printing position is referred to as cursor positioning. For some programs, such as simple print-and-space applications, cursor positioning commands are not necessary. Instead, text is printed until a *line feed* or *half line feed* control code causes the cursor to reach the perforation skip region and the printer automatically ejects the current page.

However, for many other applications, precise control of the current active position (CAP) is essential. This chapter describes the CAP for text as well as for the different types of LaserJet graphics. It also discusses the various methods of positioning the cursor and the advantages and disadvantages of each method.

Current Active Position (CAP)

The current active position is the current position of the imaginary printer “cursor”, or the location where the next character will be printed or graphics dot will be placed. After an image is printed, the CAP changes depending on what type of image was printed, whether text, raster graphics, vector graphics, or rules (rectangular area fill). The following discussion shows how the CAP changes with each image type.

CAP for Text

Before a character is printed, the CAP is along the baseline at the *character reference point*. After printing a character, the CAP moves to the right the horizontal escapement (delta X) of the character (see Figure 4-1). (This is also referred to as updating the cursor position.)

For fixed-pitch fonts, the cursor moves the HMI distance for each character printed. For example, if the HMI setting is 12/120ths of an inch, the cursor moves 12/120ths or 1/10th of an inch along the baseline for each character printed.

For proportional fonts, the amount of space moved for each character is the *horizontal escapement* distance, which the printer obtains from the character descriptor. Applications can get this information from the TFM file for the font (see Chapter 6 for more information on accessing data from the TFM files). For each character printed, the cursor moves a distance equal to the *horizontal escapement* of the character.

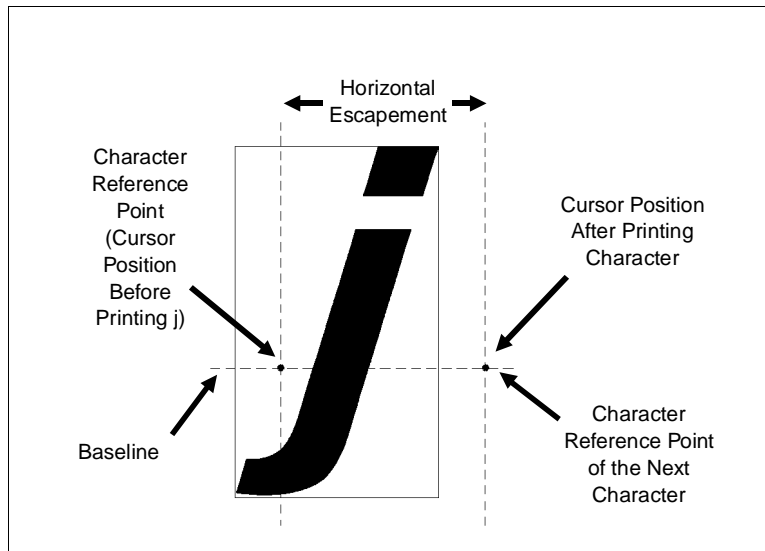


Figure 4-1. Cursor Positioning for Text

CAP for Raster Graphics

When printing raster graphics, the CAP is at the left graphics margin, starting at the current Y position. After printing the raster image, the CAP is located at the left graphics margin of the dot row following the last row of raster data (see Figure 4-2). (If a *raster height* command has been used, the CAP following an *end raster graphics* command is located at the left graphics margin and one dot row below the lowest part of the picture frame defined by the raster height command.)

CAP for Vector Graphics

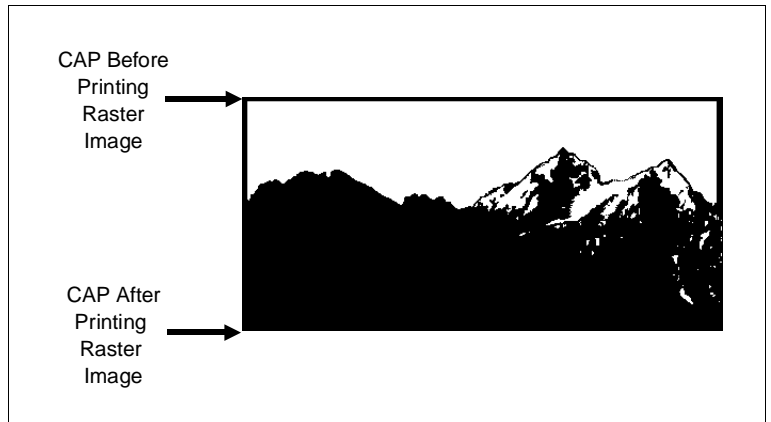


Figure 4-2. Raster Graphics Cursor Position

In HP-GL/2 mode, the CAP is referred to as the *pen location*. Whenever a plotting instruction is completed, the pen location is updated to the current point. The next instruction begins at the current pen location.

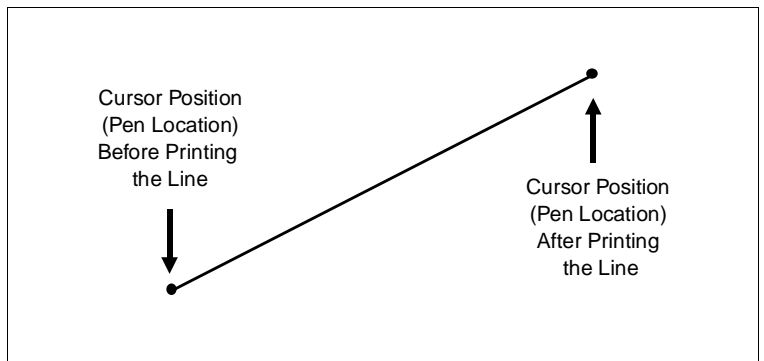


Figure 4-3. Cursor Positioning and Vector Graphics

There are some instructions, however, that do not update the current pen location. For example, after the circle (CI) instruction finishes drawing a circle, the CAP returns to the previous pen location (the circle origin). (The definition of each of the HP-GL/2 commands explains whether the CAP

is updated or restored—see the HP-GL/2 portion of the *PCL 5 Printer Language Technical Reference Manual*.)

When the printer enters HP-GL/2 mode from PCL mode, you can choose to have the HP-GL/2 pen location be the current PCL cursor position or the last HP-GL/2 pen location prior to entering PCL mode, depending on the value specified in the *Enter HP-GL/2 Mode* command ($\text{E}_c\%\#B$).

Likewise, when entering PCL from HP-GL/2 mode, the value specified in the *Enter PCL Mode* command ($\text{E}_c\%\#A$) determines whether CAP is the last PCL cursor location before entering HP-GL/2 mode, or the last pen location in HP-GL/2 mode before entering PCL mode. (See the *PCL 5 Printer Language Technical Reference Manual* for more information on *Enter HP-GL/2 Mode* and *Enter PCL Mode*.)

Note



The HP-GL/2 coordinate system is different than the PCL coordinate system. See Chapter 10 for more information on the HP-GL coordinate system.

CAP for Rectangular Area Fill

When printing *rectangular area fills* (rules), the CAP is at the upper left corner of the rectangle prior to printing. After the rule is printed, the CAP does not change; the cursor is in the same position as it was before printing the rule.

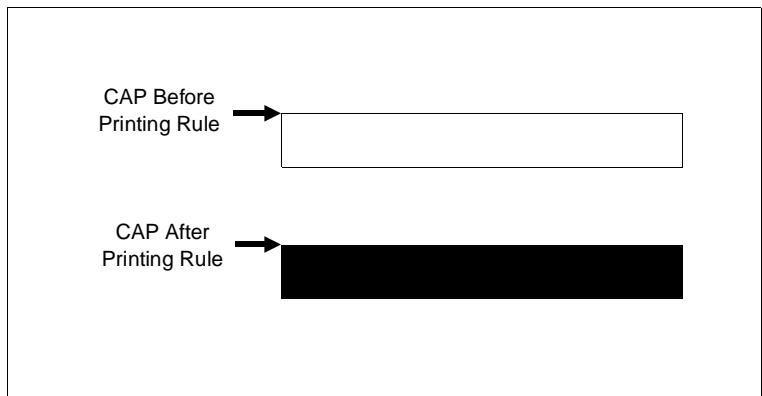


Figure 4-4. CAP Position for Rectangular Area Fill

Print-and-Space Cursor Positioning

Print-and-space applications move the cursor using the space, backspace, line feed, and carriage return control codes instead of the cursor positioning commands. Although this method of moving the cursor is not recommended for many applications, it does provide an easy way to print text for basic printing applications.

To use print-and-space positioning, set the top and left margins as well as the text length. With the perforation skip mode enabled (the default), the printer will eject a page as soon as a line feed or half line feed control code causes the cursor to enter the perforation skip region (the area between the bottom of the text area and the top of the text area of the next page).

Example: Print-and-Space Cursor Positioning

This example demonstrates print-and-space formatting using the text length command and the perforation skip mode:

<code>^cE</code>	Reset the printer (defaults features to user environment, including perforation skip mode enabled and top margin at 1/2 inch).
<code>^c&l2A</code>	Select letter-size paper.
<code>^c&lO</code>	Designate portrait orientation.
<code>^c&l8C</code>	Set VMI to 8 (6 LPI).
<code>^c&a5L</code>	Set left margin to 1/2 inch from left edge of logical page (the first character will be placed in column 5, or 6 columns from the edge of the logical page—the distance is 1/2 inch because 5 columns at 10 columns/inch = 1/2 inch; the first column is column 0).
<code>^c&l50F</code>	Set text length to 50 lines.

Text . . . Text	Place text until the number of characters is equal to the desired right margin.
CR-LF	Move to the left margin on the next line.
.	
.	
.	
Text . . . Text	Place text for the last line of the page until the number of characters is equal to the desired right margin.
CR-LF	Move to the left margin on the next line, causing the cursor to enter the perforation skip region. The page is ejected and succeeding text is formatted for a new page.

Absolute vs. Relative Positioning

Positioning the cursor can be accomplished by moving a distance relative to the current position, *relative positioning*, or from an absolute reference point, *absolute positioning*.

When using *absolute positioning* in PCL mode, the absolute reference point (0,0) is the intersection of the left edge of the logical page and the top margin. In HP-GL/2 mode, the absolute reference point is logical 0,0 of the current HP-GL/2 coordinate system. The default position for logical 0,0 is the lower left corner of the HP-GL/2 picture frame.

Use *relative positioning* when attempting to establish a relative distance between objects. For example, many word processing applications use relative horizontal positioning to justify a line of text. The application calculates the number of words that will be on a line, then calculates the amount

of space between each word. Then relative cursor moves are used after each word instead of a space character as shown in the example below.

```
Relative cursor  
commands  
or
```

Figure 4-5. Relative Positioning for Justifying Text

Use *absolute positioning* when attempting to guarantee an absolute position on the logical page. For example, placing a raster graphic in a certain spot on the page is usually easier using *absolute positioning* because the position is relative to a fixed position, not dependent on a relative distance from the previous image.

Units of Movement

PCL Units vs. Decipoints vs. Columns/Rows

The PCL 5 LaserJet printers provide flexibility in how you specify cursor moves on the PCL coordinate system. Units on the X axis may be PCL Units, decipoints, or columns; units on the Y axis may be PCL Units, decipoints, or rows. (See the *PCL 5 Printer Language Technical Reference Manual* if you need background information about cursor positioning.)

The printer uses an internal measuring unit of 1/7200th of an inch, and maps PCL Units, decipoints, columns, and rows to this internal unit. All positioning is kept in internal units and rounded to physical dot positions when data is printed. This eliminates error caused by successive rounding or truncation.

Note



The LaserJet 4 printer allows you to specify the size of PCL Units in specific sizes that range from 96 to 7200 units per inch (the default is 300). For all other LaserJet printers, the PCL Unit size is always 1/300th of an inch (the dot size).

Using PCL Units

The PCL Unit size defines the number of units per inch used in the following:

- Vertical and horizontal cursor position
- Vertical and horizontal rectangle size
- Horizontal movement after each character is printed

As mentioned above, for printers other than the LaserJet 4, the PCL Unit size cannot be specified and is set at 300 PCL Units per inch. Since the LaserJet 4 printer allows you to specify the size of a PCL Unit, the LaserJet 4 printer is discussed separately below.

For the LaserJet 4 printer:

The LaserJet 4 printer dot size is 1/600th of an inch. The default PCL Unit size is 1/300th of an inch, but you can use the *Unit of Measure* command to set the PCL Unit size to 600 PCL Units/inch or any of the valid values from 96 to 7200. For the LaserJet 4 printer, the most accurate PCL Unit Values are factors or integer multiples of 600, since the values are rounded to the nearest Unit of Measure value when converting to dots for printing. For example, if you set the Unit of Measure to 600 units/inch, the positioning values do not need to be rounded since they match the printer's resolution.

For all other LaserJet printers:

For all printers except the LaserJet 4, PCL Units are equivalent to the dot size, which is 1/300th of an inch. There is an advantage in using PCL Unit positioning commands instead of decipoint or row/column moves: PCL Unit commands are accurate to the dot because no rounding of

dot positions is performed. A disadvantage is that this method is more device-dependent than decipoints.

Using Decipoints

A decipoint is 1/720th of an inch or 1/10th of a typographical point. Using decipoints for cursor moves can provide greater device-independence, an advantage for applications such as desktop publishing where print files may be proofed on the LaserJet printer and then sent to a high-resolution imagesetter. A disadvantage of using decipoints is that when the printer converts decipoints to dots for printing, the user has little control over which dot is selected (rounding must first occur).

Using Rows and Columns

When rows are used for positioning, cursor placement is dependent on the current vertical motion index (VMI), which is the number of lines of text per inch. For example, at six lines per inch, one line is 1/6 inch and at eight lines per inch, one line is 1/8 of an inch.

Column positioning is dependent on the current horizontal motion index (HMI), which is the distance between consecutive fixed-pitch characters; this distance changes with the selected font. For fixed pitch fonts, every character is the width of the HMI. For proportional fonts, the HMI is equal to the width of the space character.

Since column positioning is based on the pitch of the currently selected font, it can be an advantage in some applications. Disadvantages of using row/column positioning are that the commands are not useful for complex page formatting, they are difficult to use with proportional font formatting, and they are font dependent.

HP-GL/2 Plotter Units

When in the HP-GL/2 mode, the printer moves in either *plotter units* (plu) or *user units*. There are 1016 plotter units in an inch and 40 in a mm (1 plu = .025mm or .00098 in.). User units can be set up to virtually any size. Both types of HP-GL/2 units are converted into the equivalent number of dots prior to printing.

Saving the Cursor Position

Sometimes it is useful to be able to save the previous cursor location, or several previous cursor locations. The *push/pop cursor position* command allows you to store and recall up to 20 positions.

A common application of this command is saving the cursor position prior to executing, calling, or overlaying a macro and recalling the position after the macro command has been carried out.

When using the *push/pop cursor position* command, the application must carefully manage all push/pop activity to ensure that the correct position is being returned.

Example: Saving/Restoring the Cursor Position

This example demonstrates the use of the *push/pop cursor position* command. (It is assumed that the job has been correctly set up and that a macro with a macro ID of 53 has been previously created.)

$\text{E}_c\&f\emptyset S$

Store (push) the current cursor position.

$\text{E}_c\&f53y3X$

Call macro number 53.

$\text{E}_c\&f1S$

Recall (pop) the position the cursor was in prior to calling the macro.

Positioning the Cursor at the Limits of the Page

As noted earlier, attempting to print too close to the unprintable region can cause clipping. The following examples show how to print as close as possible to the edges of the page without clipping any characters or raster data. (To see the actual size of the area where you can move the cursor and place a dot, see the section titled “The Actual Printable Area” in Chapter 3.)

Note



The PCL 5 LaserJet printers clip only those portions of the character that extend into the unprintable region (dot-level clipping); characters are not clipped entirely unless the whole character extends into the unprintable region. (This differs from the LaserJet series II printer, which performs character cell clipping.)

Example: Placing Graphics at the Top-Most Position

This example demonstrates placing raster graphics data at the top-most printable dot row. (It is assumed that the proper job setup commands have been sent as described in Chapter 2. If this example is sent to the LaserJet 4 printer, it is assumed that the Unit of Measure setting is at 300 PCL Units/inch.)

<code>Ec&l0E</code>	Set top margin to 0.
<code>Ec*p50Y</code>	Move absolute vertical cursor position 50 dots below the top of the physical page (a 50-dot move places the cursor at the top of the printable area).
<code>Ec*t300R</code>	Set the raster graphics resolution to 300 dots per inch.
<code>Ec*r0A</code>	Set the raster graphics margin to the left-most printable position.
<code>Ec*b#Wdata . . .</code>	Transmit raster data.
<code>Ec*rB</code>	End raster graphics.

Example: Placing Text at the Top-Most Position

This example places a character at the top-most printable dot row. (It is assumed that the proper job setup commands have been sent as described in Chapter 2. If this example is sent to the LaserJet 4 printer, it is assumed that the Unit of Measure setting is at 300 PCL Units/inch.)

`Ec&ℓØE`

Set top margin to 0.

`Ec*p8ØY`

Use an absolute vertical cursor position move down to dot 80 (80 dots is equal to the unprintable region distance (50 dots) plus the character ascent distance (30 dots); this value is read from the TFM files [see the explanation below]).

Text . . . Text

Transmit character data.

In this example, an 80-dot move is required to ensure that portions of characters are not clipped. To determine the exact movement required to prevent clipping, data must be available regarding the baseline distance. The baseline (or ascent) distance describes how far the character cell extends above the baseline; this distance varies from one font to another.

Note



The *ascent* distance is stored in the TFM file for the font. Refer to Chapter 6 to find how to access TFM files.

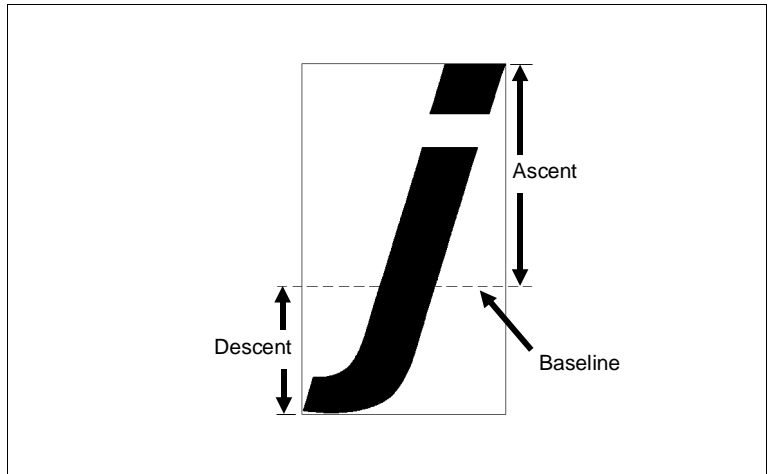


Figure 4-6. The Ascent and Descent Distances

Example: Moving to the Right-Most Printable Position

Background: In order to move to the right-most printable position, the application must use the width of the logical page to calculate the distance that must be moved. The width of the default logical page for letter-size paper is 2400 dots (at 300 units/inch—see the Printable Area Boundaries table in the *PCL 5 Printer Language Technical Reference Manual*). When calculating the number of characters that fit in this 2400-dot width, the procedure varies depending on whether the font is fixed-pitch or proportional:

- For fixed-pitch fonts, divide the logical page width by the pitch of the selected font. For example, for a 10-pitch font, 2400 dots divided by 30 dots per character equals 80 characters (300 dots/inch divided by 10 characters per inch equals 30 dots/character).
- For proportional fonts, add the *horizontal escapement* distances for the characters that will be placed on the line. The sum should not exceed the width of the page (for example, 2400 dots for letter-size paper).

This example positions the cursor at the right-most printable position on a letter-size page. (It is assumed that the proper job setup commands have been sent as described in Chapter 2. If this example is sent to a LaserJet 4 printer,

the Unit of Measure setting must be 300 PCL Units per inch in order to work properly with the values given.)

`Ⓔ*p2399X`

Absolute cursor move to dot number 2399, which is actually the 2400th dot (the first position is dot 0).

Example: Moving the Cursor to the Lowest Page Position

This example shows how to position the cursor at the bottom-most printable position on the page. (This example assumes that we are using letter-size paper and that the proper job setup commands have been sent as described in Chapter 2. For the LaserJet 4 printer, the Unit of Measure setting must be 300 PCL Units/inch in order for this example to work properly with the values given.)

`Ⓔ&t0E`

Set top margin to 0.

`Ⓔ*p3249Y`

Move the cursor with an absolute cursor move command to dot 3249.

The default letter-size logical page size is 3300 dots high (see the *Printable Area Boundaries* table in the *PCL 5 Printer Language Technical Reference Manual*). Subtracting 50 dots for the unprintable region at the bottom of the page leaves 3250 dots. Since the first dot is dot 0, 3249 represents the 3250th dot.

Example: Placing Text at the Page Bottom

This example demonstrates printing text at the bottom-most printable position. (It is assumed that the proper job setup commands have been sent as described in Chapter 2.)

`Ⓔ&lØE`

Set top margin to 0.

`Ⓔ*p3241Y`

Move the cursor with an absolute cursor move to dot number 3241.

As in the previous example, the height of a letter-size page is 3300 dots. Subtracting 50 dots for the unprintable region at the bottom of the page leaves 3250 dots. To print a character, we must also subtract the *descent* distance of the character (that distance from the baseline to the bottom of the character cell). In this instance, the descent distance is 8 dots, leaving a distance of 3242 dots ($3250 - 8 = 3242$). Since the first dot is dot 0, we position the cursor at dot 3241, the 3242nd position.

Note



The *descent* distance is stored in the TFM file for the font. Refer to Chapter 6 for information on accessing the TFM files (see Figure 4-6).

How the Paper Path Affects Cursor Placement

For some applications, such as printing on preprinted forms, a slight skew may be noticed. In other words, lines or text printed on the laser printer may not line up exactly with lines on the preprinted form. This skewing is due to the mechanical tolerances in the printer's paper path, not to inaccuracy of dot placement.

The relationship of any dot to another dot on the page is always maintained. In other words, if the laser printer prints two parallel lines 5 inches apart, the lines will always be parallel to each other (within the resolution of the printer), but the two lines will not necessarily be exactly parallel with the edge of the physical page.

Problems with this type of skewing can be eliminated by using plain paper and printing the form on the page along with the data. A typical example of this type of solution is an application that incorporates several predesigned forms that can be downloaded as macros. When the application is ready to print a certain form, the macro for that form is overlaid and then the data is filled in using cursor placement commands.

Note



The PCL 5 LaserJet printers support macro cartridges, which are an ideal way to support this type of application. (See Chapter 12 for more information about macro cartridges.)

Contents

Introduction	5-1
Selecting Fonts	5-2
Bitmapped Fonts vs. Scalable Typefaces	5-2
Selecting Fonts by Characteristic	5-3
Primary and Secondary Fonts	5-8
General Font Management	5-9
Managing the Download Process	5-9
Downloading Fonts	5-11
Font Auto-Rotation	5-13
Working With Fonts	5-14
Justifying Text When Using Proportional Fonts	5-14
Adjusting Line Spacing for Point Size	5-17
Transparent Print Data and Special Characters	5-18
Underlining Characters	5-19
Font Headers	5-20

Introduction

The PCL 5 LaserJet printers can print using scalable typefaces and bitmapped fonts. To handle scalable typefaces, all PCL 5 printers come equipped with Agfa's Intellifont font scaling technology. The LaserJet 4 printer also includes the TrueType font scaling technology, allowing users to enlarge and reduce type using Intellifont and/or TrueType.

Having premium typefaces resident within the LaserJet printer allows your application to choose from a nearly limitless range of point sizes for producing top-notch documents. Not only are the scalable typefaces available, but the scaling performance is exceptional, too. Combined with the PCL 5 printer's improved I/O, the LaserJet printer font scaling capabilities offer your software application excellent printing performance. (Each PCL 5 printer has a different combination of scalable typefaces and bitmapped fonts—see the *PCL 5 Comparison Guide* for a list of the on-board typefaces and fonts for each printer.)

This chapter explains how to use fonts, from downloading and selecting them, to adjusting line spacing and justifying text. The next three chapters, Chapters 6, 7, and 8, cover supporting fonts in your application. For information on creative effects, such as filling fonts with shading and cross-hatch patterns, see Chapter 11, *The Print Model*. For information on using HP-GL/2 vector graphics to produce special font effects such as mirrored fonts, rotating fonts to any angle, isotropic scaling or anisotropic scaling, see Chapter 10.

Selecting Fonts

Bitmapped Fonts vs. Scalable Typefaces

As previously mentioned, the PCL 5 LaserJet printers can print using scalable typefaces and bitmapped fonts. These fonts can be either cartridge-based, disk-based, internal, or in the case of the LaserJet 4 printer, SIMM-based. Scalable typefaces are provided so that LaserJet printer users can have access to a large number of typefaces in a nearly unlimited number of font sizes.

Scalable typefaces and bitmapped fonts are all selected using the same process. To select a specific font, the application sends a command to the printer. If a bitmapped version of the font already exists in the printer when the command is received, the specified font is selected. If the font needs to be scaled, the font is scaled on a character-by-character basis to the specified point size and then selected; only those characters that will be printed are scaled. The font selection escape sequence actually initiates the font scaling process, eliminating the need for a separate font scaling command (the *point size* command, $\text{^c}(s\#V$, becomes the operator for proportionally spaced fonts; *pitch*, $\text{^c}(s\#H$, is the operator for fixed-spaced fonts).

Note



For most PCL 5 LaserJet printers, scalable typefaces are “bound” to a particular symbol set when downloaded to the printer. In other words, once the typeface is downloaded, only the specified set of characters may be used unless the typeface is downloaded with another symbol set. With the LaserJet 4 and IIIP printers, typefaces may be either “bound” or “unbound.” If an “unbound” scalable typeface is downloaded to the printer, the user may select from any of the many symbol sets contained within the printer. See Chapter 7 and Appendix C for information about “unbound” scalable typefaces.

**Example:
Printing with the
Internal Scalable
Typefaces**

This example shows how to scale an internal typeface and print the resulting font:

37-Point Univers

<code>ⒺcE</code>	Reset the printer.
<code>Ⓔc&lℓO</code>	Set the orientation to portrait.
<code>Ⓔc(8U</code>	Select the Roman-8 symbol set.
<code>Ⓔc(s1p37vℓs3b4148T</code>	Select a bold, upright, 37-point Univers font. No downloading is necessary (this typeface is resident in the PCL 5 printers). This command automatically scales the typeface to a 37-point font and prepares it for printing.
<code>37-Point Univers</code>	Send the desired text to the printer.
<code>ⒺcE</code>	Send a reset at the end of the print job.

Selecting Fonts by Characteristic

Fonts may be selected by sending a font selection command that specifies the desired font characteristics or by selecting a font ID number using the *font selection by ID #* command. Although the *font selection by ID #* command is easier to code, HP recommends instead that software developers select fonts using all the font characteristics as shown on the next page. (Font ID numbers can be easily overwritten, especially in shared or multi-user environments.)

Although the font select escape sequences may be shortened to save code space, the right amount of abbreviation depends on other fonts in the printer as well as the previously selected font. Shortened font select commands can sometimes be abbreviated too much, causing the wrong font to be selected. Using the entire font select escape sequence

eliminates the possibility of leaving out a necessary characteristic while trying to shorten the command.

The following list shows the font selection characteristics in order of priority from highest to lowest.

<u>Priority</u>	<u>Characteristic</u>	<u>Command</u>
Highest	Symbol Set	$E_c(\text{ID})$
	Spacing	$E_c(\text{s}\#\text{P})$
	Pitch	$E_c(\text{s}\#\text{H})$
	Height	$E_c(\text{s}\#\text{V})$
	Style	$E_c(\text{s}\#\text{S})$
	Stroke Weight	$E_c(\text{s}\#\text{B})$
Lowest	Typeface	$E_c(\text{s}\#\text{T})$

As in PCL 4 LaserJet printers, the PCL 5 printers compare the highest-priority font attribute specified with the same attribute of all the fonts accessible to the printer. If only one font contains the matching attribute, the printer selects that font. If more than one exists, the printer compares the next highest priority attribute until only one font remains.

If more than one font still remains, the font location is the next selection criterion. The priority is as follows, from highest to lowest:

- Downloaded bitmapped fonts (lowest ID first)
- Downloaded scalable fonts (lowest ID first)
- Bitmapped cartridge fonts
- Scalable cartridge typefaces
- Bitmapped internal font
- Scalable internal typefaces

If there is more than one bitmapped font in the same location with the same selection criteria, the font with the specified orientation is selected (auto-rotation provides all four orientations for any given font). If only scalable fonts remain and more than one is available, selection is arbitrary.

Note

If you are unfamiliar with selecting fonts by characteristic, the *PCL 5 Printer Language Technical Reference Manual* explains how the printer selects fonts, discussing the font selection parameters and the priority order of those parameters.

**Example:
Selecting Fonts
by Characteristic**

This example demonstrates how to select fonts by characteristic. Notice that the font selection commands are combined into one command, `Ec(sØp1Øh12vØsØb3T`, for efficiency. (It is assumed that all job and page setup commands have been sent to the printer before sending the commands in the example. Job and page setup are discussed in Chapters 2 and 3.)

<code>Ec(8U</code>	Select the Roman-8 symbol set.
<code>Ec(sØp1Øh12vØsØb3T</code>	Select fixed-spaced, 10-pitch, 12-point, upright, medium, Courier.
<code>ABCDEFGHIJL. . .</code>	Print text.
<code>Ec(8U</code>	Select the Roman-8 symbol set.
<code>Ec(s1p14vØs3b4148T</code>	Select a proportionally spaced, 14-point, upright, bold, Univers font.
<code>MNOPQRSTUVWXYZ. . .</code>	Print text.

Note

When selecting fonts by characteristic as shown in this example, the desired font will always be selected as long as it is accessible to the printer. Keeping track of the last selected font is not necessary as is the case when trying to use an abbreviated command.

Font Selection Exceptions

In order to provide for some of the differences between bit-mapped and scalable fonts, the following exceptions apply to selecting scalable fonts and typefaces:

Symbol Set—The PCL 5 printers have many symbol sets from which to choose when creating fonts. See the *PCL 5 Comparison Guide* for a complete list of symbol sets available to each PCL 5 printer. For information on user-defined symbol sets, see Appendix D (the BUILD-SYM utility). See the *PCL 5 Printer Language Technical Reference Manual* for symbol set codes.

Pitch—For a monospaced scalable font or typeface, any specified pitch is available within the range of the PCL printer language. However, the pitch value is converted to a corresponding point size value which is scaled by the Intellifont algorithm in the PCL 5 LaserJet printers. The equation used to convert pitch to height (point size) is:

$$\text{Height} = 1/(\text{Desired Pitch} * (\text{Spacing Value} / 8782) * 0.01383)$$

Note



For the above equation, the TFM files provide the spacing value (TAG 412), the Design Unit value (8782—TAG 408), and the point size in inches (0.01383—TAG 406). For more information about TFM files and TAG values, see Chapter 6, *AutoFont Support*.

The result of the height calculation is converted to the closest point size in the range from .25 to 999.75 point. For example, the pitch in the font header of a CG Times scalable font is 5291. If the requested pitch is 10, then using the above equation, the printer calculates the corresponding point size as follows:

$$1/(10.00 * (5291/8782) * 0.01383) = 12.00 \text{ point}$$

(See the example on the next page concerning selecting a scalable typeface by specifying pitch without point size.)

Height—With scalable fonts and typefaces, you can specify a font height from .25-point to 999.75-point, in increments of one-quarter point.

Style—The style attribute in the current font header is two bytes, allowing the capability to incorporate more styles in addition to upright (0) and italic (1). Refer to the *PCL 5 Printer Language Technical Reference Manual* for more information about the style attribute.

Typeface—The typeface attribute in the current font header occupies two bytes. Scalable typefaces supplied by HP or Agfa may be designated using a two-byte typeface value. This value is calculated by adding 4096 to the typeface value listed in the *HP PCL 5 Comparison Guide*. (Refer to the *PCL 5 Printer Language Technical Reference Manual* for more information on how the typeface number is used.)

Example: Selecting a Scalable Fixed-Pitch Typeface

To select a scalable Courier typeface, select the following attributes:

<code>E_c(s0P</code>	Select fixed spacing.
<code>E_c(s5H</code>	Select a pitch of 5 characters/inch.
<code>E_c(s0S</code>	Select the upright style.
<code>E_c(s0B</code>	Specify medium stroke weight.
<code>E_c(s4099T</code>	Specify the scalable Courier typeface.

In this case, the printer selects the scalable Courier typeface and creates a 25-point font (using the equation above). The height parameter is ignored, since the height of the font is dependent on the pitch requested. Since typeface is the lowest priority attribute, the correct typeface value must be used in the selection command if a specific typeface is to be selected. In this case, the correct typeface value for Courier (from Agfa or HP) is 4099 (4096 + 3), although a value of 3 achieves the same result. (See the *HP PCL 5 Comparison Guide* for the typeface values.)

Primary and Secondary Fonts

In situations where users are frequently alternating between two fonts in a print job, using primary and secondary fonts may improve both printer and system performance. If a secondary font is designated, the software can switch between the primary and secondary fonts using the *Shift In* and *Shift Out* control codes, providing an efficient font selection technique. The example below outlines how secondary fonts may be used.

Example: Using Primary and Secondary Fonts

This example shows the use of primary and secondary fonts for font selection.

<code>Ec(0U</code>	Select the ASCII symbol set as primary.
<code>Ec)0U</code>	Select the ASCII symbol set as secondary.
<code>Ec(s1p10v0s0b4101T</code>	Select a 10-point CG Times font as primary, for use as a text font.
<code>Ec)s1p14v0s3b4148T</code>	Select a bold, 14-point Univers font as secondary, for use as a headline font.
<code><SO></code>	Use the Shift Out control code (ASCII 14) to access the secondary headline font.
This is a Headline	Print a headline.
<code><SI></code>	Use the Shift In control code (ASCII 15) to access the primary text font.
This is text copy	Print a paragraph using the text font.

General Font Management

Managing fonts involves giving the user a choice of fonts, making sure the fonts are accessible (the hardware-based fonts are installed or the soft fonts are downloaded), selecting the fonts when needed, and deleting them when the memory they occupy is needed for another purpose.

This segment of the chapter describes managing the font process so that the user knows which font cartridges must be installed and which soft fonts downloaded. It also discusses downloading fonts to the printer and how auto-rotation and scaling affect font management.

Managing the Download Process

Managing fonts is a difficult task for an application to handle. Users frequently generate pages containing many different fonts. Each font selected by the user must be either an internal printer font, a font cartridge installed in the printer, a LaserJet 4 SIMM-based font, or a bitmapped or scalable font which has been downloaded to the printer.

Managing the font download process includes the task of keeping track of the fonts needed to print the job correctly:

For printers without status readback, the application must keep a record of the fonts used for a particular job. The application can then make the information available to the user, if desired, by means of a menu or a display. If the application is using AutoFont Support, the GLUE.TXT file can be read to obtain the current fonts available to the system and then the required fonts can be downloaded to the printer.

For the LaserJet 4 printer, the application must also keep track of the fonts needed for a particular job. The PCL status readback commands can be used to request the fonts that are currently available in the printer. The application can also read the GLUE.TXT file to see the fonts available to the system. Then the application can download the required fonts that are not already in the printer. If the fonts are not available for some reason, the application can notify the user.

Example: Font Management

This example outlines a way of managing fonts that provides the necessary fonts, only notifying the user if there is some action to be taken:

- 1) As the application is loaded, it determines which fonts are available to the user.
 - The application should be aware of all resident fonts for the device.
 - System-level soft fonts should be available to the user. The application can use the GLUE.TXT file to determine which fonts are available on the system level.
 - For LaserJet 4 printers, the application can use PCL status readback commands to request the fonts that the printer has access to, including cartridge fonts and SIMM-based fonts. After the initial font status check, if the application detects an off-line status, it should recheck the availability of cartridge fonts.
- 2) While creating a document, the user chooses several fonts from the list of available fonts provided by the application.
- 3) The user sends the page for printing.
- 4) The driver downloads the required fonts that are not already in the printer and prints the job.

Note



A list of fonts required to print this job should be attached to the file for future reference. Then the user can recreate the same document later, when specific font information related to the job has been forgotten.

Downloading Fonts

Downloading fonts involves specifying a font ID number, copying the font file to the printer, and then making the font permanent (if desired). All the necessary escape sequences are already embedded in each font file so that a DOS COPY command using the /B [binary] option successfully downloads the font. The following example demonstrates two fonts being downloaded and used.

Example: Downloading Fonts

This example shows how to download bitmapped fonts to the printer as permanent fonts. (This example assumes that the proper job and page setup commands have already been sent to the printer.)

<code>Ⓔc*c200D</code>	Set the font ID number to 200.
<code>COPY /B HV120RPN.USP PRN:</code>	Copy the file containing the 12-point Helvetica medium font to the printer using the /B option (binary copy).
<code>Ⓔc*c200d5F</code>	Specify as permanent the font with ID number 200.
<code>Ⓔc*c201D</code>	Set the font ID number to 201.
<code>COPY /B HV180BPN.USP PRN:</code>	Copy the file containing the 18-point Helvetica bold font to the printer using the /B option (binary copy)
<code>Ⓔc*c201d5F</code>	Specify as permanent the font with ID number 201.
<code>Ⓔc(0U</code>	Select the ASCII symbol set.
<code>Ⓔc(s1ph18vs3b4T</code>	Select the 18-point, Helvetica bold font as the primary font.
<code>HEADLINE TEXT</code>	Print the headline.
<code>Ⓔc(s1ph12vsb4T</code>	Select the 12-point, Helvetica medium font as the primary font for use as a text typeface.
<code>TEXT TEXT TEXT</code>	Print the paragraph text.

Making Fonts Permanent vs. Temporary

Downloaded fonts can be specified as either temporary or permanent. Downloaded fonts are automatically designated as temporary unless they are specifically set to permanent.

Both permanent and temporary fonts are “perishable data.” Permanent fonts are not affected by reset conditions, but are erased any time power is removed or memory is reconfigured, which includes the following situations:

- The page protection status changes (all PCL 5 printers)
- The printer language changes (LaserJet 4/IIISi only)
- The device resolution changes (LaserJet 4 only)

Temporary fonts are also erased when power is removed or memory is reconfigured, plus they are erased when the printer is reset (E_cE, UEL, or RESET key). Essentially, temporary fonts are available for the duration of the PCL job, since they are erased under reset conditions, which happen at the beginning and end of every PCL job.

The decision to make fonts permanent or temporary depends on the number of users using the printer and the applications they are using. A font that will be used many times throughout a print job but would never be used by other jobs would best be designated as a temporary font. Since temporary fonts are erased with a printer reset, the next print job clears the temporary fonts from memory, freeing the space for other data.

Conversely, fonts that will be used by many users or by many print jobs should be created as permanent fonts so that they can be easily accessed without downloading them repeatedly. In many cases, the best option is to allow the user to choose whether the fonts will be downloaded as permanent or temporary.

Font Auto-Rotation

If a font is available in one orientation and a command selects that font while in another orientation, the printer automatically generates a new version of that font in the current orientation. For example, if the printer is printing using *portrait* 12-point ITC Garamond for one page of a print job and then 12-point ITC Garamond is requested on the next page (which happens to be *landscape*), the printer will automatically rotate the necessary characters to create 12-point landscape ITC Garamond if those characters don't already exist in the printer.

There are a few things to keep in mind concerning font auto-rotation.

- Fonts are auto-rotated one character at a time, as needed. Only those characters that will be printed on the page are rotated—not the entire font.
- A reset eliminates fonts that have been auto-rotated.
- Auto-rotated fonts not used on the current page are automatically deleted when the memory they occupy is needed elsewhere.
- The auto-rotation process uses a certain amount of overhead, requiring more user memory than would otherwise be necessary. If the application requires close to the available user memory, the auto-rotation process could use enough extra memory to cause a 20 MEMORY OVERFLOW error message.

Note



These four points also apply to font *scaling*. The printer automatically scales only those characters that are needed, and will delete them if they are not used on the current page when the memory they occupy is needed elsewhere. Scaled fonts are also erased with a printer reset.

Working With Fonts

This segment of the chapter gives examples of some of the more common ways that software developers use fonts within their applications.

Justifying Text When Using Proportional Fonts

Justification of proportionally spaced characters is an essential part of many applications/drivers. To accomplish proportional justification, information is required regarding the space occupied by each character. The horizontal escapement value, which represents the distance from the character reference point of one character to the character reference point of the next one, is essential for justifying text. (For highly precise justification, an application may also use the *left extent* and *right extent* values for the first and last characters in a line.)

Note



The *horizontal escapement* for each character can be read from a TFM file. For scalable fonts, *horizontal escapement* is in Design Units. When converting from Design Units to dots, the “dot” value may be fractional (for example, 24.36 dots). The PCL 5 LaserJet printers round these values to the nearest whole dot. Note that the rounding is not cumulative. In other words, if you are justifying 10 characters that are 10.25 dots each, each character’s width is calculated as 10 dots, not $(10.25 * 10)/10$. (Refer to Chapter 6 for more information on accessing TFM files.)

In most instances, the application’s TFM reader routine has already read the font information it requires from the TFM files and saved it in a font-specific driver file. The following example shows horizontal escapement values that were obtained from the TFM files to correctly space each character in a line.

Example: Justifying Text

This example demonstrates proportional text justification. The words “This is a test, test, test.” are justified in a 2-inch column. To justify text, the horizontal escapement value for each character can be read from the font’s associated TFM file. (This data is for a 10-point Times Roman font.)

ASCII Character	Decimal Equivalent	Horizontal Escapement (In Dots)
T	84	26
h	104	23
i	105	12
s	115	18
<SP>	32	11
i	105	12
s	115	18
<SP>	32	11
a	97	21
<SP>	32	11
t	116	14
e	101	18
s	115	18
t	116	14
,	44	10
<SP>	32	11
t	116	14
e	101	18
s	115	18
t	116	14
,	44	10
<SP>	32	11
t	116	14
e	101	18
s	115	18
t	116	14
.	46	10
		<hr/>
		437

Using the horizontal escapement values above and assuming the column width is two inches (600 dots), send the following commands to the printer:

$\text{E}_c(\text{ØU})$ Select the ASCII symbol set

$\text{E}_c(\text{sØp1ØvØs3b5T})$ Select a 10-point, bold, Times Roman font

$\text{This}\langle\text{SP}\rangle\text{E}_c^*\text{p}+32\text{Xis}\langle\text{SP}\rangle\text{E}_c^*\text{p}+33\text{Xa}\langle\text{SP}\rangle\text{E}_c^*\text{p}+33\text{Xtest},$
 $\langle\text{SP}\rangle\text{E}_c^*\text{p}+33\text{Xtest},\langle\text{SP}\rangle\text{E}_c^*\text{p}+32\text{Xtest}.$

Each word is separated by a space control code, $\langle\text{SP}\rangle$, and a relative cursor move to the next character.

To calculate the total cursor move for the line, the horizontal escapement values are added, totaling 437 dots. With a column width of 600 dots (two inches), there are 163 extra dots remaining to add to the five inter-word spaces (600 minus 437 equals 163 dots). Dividing the number of dots by five (for the five spaces) leaves 32.6 dots per space ($163/5 = 32.6$). Since the PCL 5 LaserJet printers do not accept fractional dot values, an extra dot must be added to three of the spaces ($.6 * 5 = 3$ dots).

Note



To reduce the amount of data and make the software application more efficient, the $\langle\text{SP}\rangle$ control code movement of 11 dots can be incorporated into the cursor positioning commands as follows:

$\text{This}\text{E}_c^*\text{p}+43\text{Xis}\text{E}_c^*\text{p}+44\text{Xa}\text{E}_c^*\text{p}+44\text{Xtest},\text{E}_c^*\text{p}+44\text{Xtest},\text{E}_c^*\text{p}+43$
 $\text{Xtest}.$

Adjusting Line Spacing to Correspond to Point Size

Text leading, the vertical line spacing between lines of text, directly affects readability. As a general rule, leading for body text should be approximately 120% of the point size selected. For example, suitable leading for a 10-point font is 12 points (120% of 10).

Note



Although 120% of the point size is a recommended default spacing value, applications should allow users to adjust leading to suit their needs. Headlines, for example, are generally set with tighter leading than body text, and the exact value depends on the type of application and what the writer/designer is trying to accomplish with the headline.

Example: Adjusting Line Spacing

In this example, two lines of text are printed at the default line spacing, and then line spacing is adjusted using the *vertical motion index* (VMI) command. The third line shows that the line spacing change has taken effect. Note that the line spacing command doesn't take effect until a line feed occurs. (It is assumed that the proper job setup and page setup commands precede the commands in this example [see Chapters 2 and 3].)

DOWNLOAD	Download an ASCII, 12-point, bold Helvetica font.
<code>^c(OU</code>	Select the ASCII symbol set.
<code>^c(s1p12v0s3b4T</code>	Select a proportionally spaced, 12-point, upright, bold Helvetica font.
<code>Text text text<LF></code>	Send the first line of 12-point text.
<code>^c&l20.6C</code>	Set line spacing to 20.6/48 inches per line using the VMI command. (See the following discussion on calculating VMI.)
<code>Text text text<LF></code>	Send another line to see current spacing (default spacing still in effect until line feed <LF> sent).
<code>Text text text</code>	Send the third line of 12-point text.

How to Calculate the VMI Value

Use VMI to designate line spacing:

- 1) Calculate a leading value ($120\% * 12 \text{ point} = 14.4 \text{ point}$)
- 2) Convert the leading value to inches ($14.4 \text{ points per line} / 72 \text{ points per inch} = .2 \text{ inches per line}$)
- 3) Set VMI to .2 lines per inch ($.2 * 48 = 9.6 \text{ VMI}$)

Note



The formula for calculating VMI can be reduced to:
(leading as % of point size * point size * 2/3).
In the example above, $120\% * 12\text{-point} * .6666 = 9.6$.

Using Transparent Print Data to Access Special Characters

In some situations, it may be necessary to print the characters in the “control code range” of the symbol set, in other words, those characters occupying the positions decimal 7 through 15 and 27. Normally, if the printer receives a request to print a character in this range, a control code is executed. In order to print a character that is located in this range, the *transparent print data* command must first be sent to the printer.

The following example shows how to access characters in the control code range.

Example: Accessing Special Characters

This example shows how to access control characters using the transparent print data command:

<code>␣(10U</code>	Select primary symbol set (PC-8).
<code>␣&p1X</code>	Send the transparent print data command. This command designates that the following byte should not be acted upon, but printed instead.
<code>27</code>	Send decimal 27 to the printer. Instead of executing the escape control code, the printer will print the left-pointing arrow symbol.

Underlining Characters

Underlining may be accomplished using the *underline* command or using *rectangular area fill* (rules). Other methods of underlining (such as using raster graphics) should be avoided since they can cause performance problems due to the overhead associated with them.

The PCL 5 LaserJet printers have the capability to print what is called a floating underline (`␣&d3D`), meaning that the underline distance is determined by the greatest underline distance below the baseline of all the fonts printed on the current line of text. This feature is beneficial in that it ensures that all underscore lines on one line of text are at the same level, even if they aren't the same thickness.

Every font has a specific *underscore thickness* and a specific distance from the baseline to the center of the underscore, the *underscore depth*. The *underscore thickness* and *underscore depth* for each font are obtained by reading the TFM file for that font (see Chapter 6 for information about reading the TFM files). The printer automatically uses these values when the floating underline is enabled. When using the rectangular area fill command to print underscore lines, the underscore thickness and underscore depth values can be read from the TFM files and used to determine the desired position and thickness of the underline.

Font Headers

Prior to the introduction of TFM files, the LaserJet font header was used to provide font metric data to software applications, most frequently for font spacing tables. With the introduction of the Tagged Font Metric (TFM) file and a more universal font support approach, reading the font header is no longer the preferred method of font support (see Chapter 6 on “AutoFont Support”). However, the font header for bitmapped fonts continues to contain a useful quantity of font metric information.

Several font header fields that were reserved and/or not recognized by PCL 4 printers are defined for use by PCL 5 printers. For those developers that are creating LaserJet-compatible fonts, the *PCL 5 Printer Language Technical Reference Manual* contains a reference specification for the scalable font header and the bitmapped font header, including information on creating bitmapped fonts with compressed data.

Note



To allow for specific style selection, the style byte is two bytes in length. The typeface byte is also two bytes long to enable differentiation between multiple versions of the same typeface. (With the two-byte value, typefaces can be differentiated by vendor; for example, CG Times vs. Times Roman.) See the Font Selection chapter of the *PCL 5 Printer Language Technical Reference Manual* for more detailed header information.

Contents

Introduction	6-1
TFM File Structure	6-2
Header Structure	6-3
Directory Structure	6-6
Tag Entry Structure	6-7
TFM Usage.	6-10
TFM Tag Descriptions	6-11
Font Identification	6-12
Font Parameters	6-23
Character Parameters	6-27
Kerning Information	6-29
Device Data	6-35
PANOSE Numbers.	6-36
“Glue” File Description and Usage	6-37
Supported Fonts	6-47
Comparing Past Font Support With TFM Support . .	6-49
AutoFont Support.	6-52
Hard-coding TFM Data	6-52
TFM Reader Integration	6-53
Available Tools for TFM Reader Integration	6-55
The TFM Reader Program	6-56
End-User Considerations	6-57
Using the TFM Reader	6-58
TFM Reader Data Flow.	6-59
Modifying the TFM Reader.	6-61
Accessing the TFM Data Structure	6-62
Supplied TFM Files	6-66

File Naming Convention	6-67
Sample TFM Implementation	6-68
Selecting Fonts Using TFM Information	6-73
Symbol Set	6-74
Spacing	6-74
Pitch	6-74
Height (Point Size)	6-75
Style	6-75
Stroke Weight	6-78
Typeface	6-79
Locating the TFM Files	6-79
The Glue File	6-80
Creating TFM Files	6-81
Available Tools	6-82
The TFM Writer	6-82
PANOSE Numbers	6-88
Modifying the TFM Writer	6-88
Compiling the TFM Writer	6-100

Introduction

AutoFont Support provides immediate support for accessory type. It is an industry standard font metric specification that uses the Tagged Font Metric (TFM) file format as its base. AutoFont Support was implemented at the time the LaserJet III printer was introduced, and has been adopted by industry leading software and hardware manufacturers as an efficient and time-saving way to manage type.

On the AutoFont support disk that is shipped with each HP font product, your customers receive an AutoFont Support Installer utility. The AutoFont Support Installer copies each of the TFM files associated with the new typefaces to the user's hard disk, and updates the "glue" file. The glue file serves as a link between the font file and its associated TFM file, keeping track of the typefaces, the associated TFM files, and the locations of these files.

Note



Hewlett-Packard supplies font metric information in the TFM format only. Font spacing tables are not provided for each font product.

To help you update your application for AutoFont Support, HP provides a TFM Reader program module and some related tools. This chapter discusses the TFM file format and how to use the HP-supplied tools to implement font support in your product.

TFM File Structure

Figure 6-1 shows the TFM file format. It consists of three types of structures: (1) the header, (2) one or more directories, and (3) the tagged data.

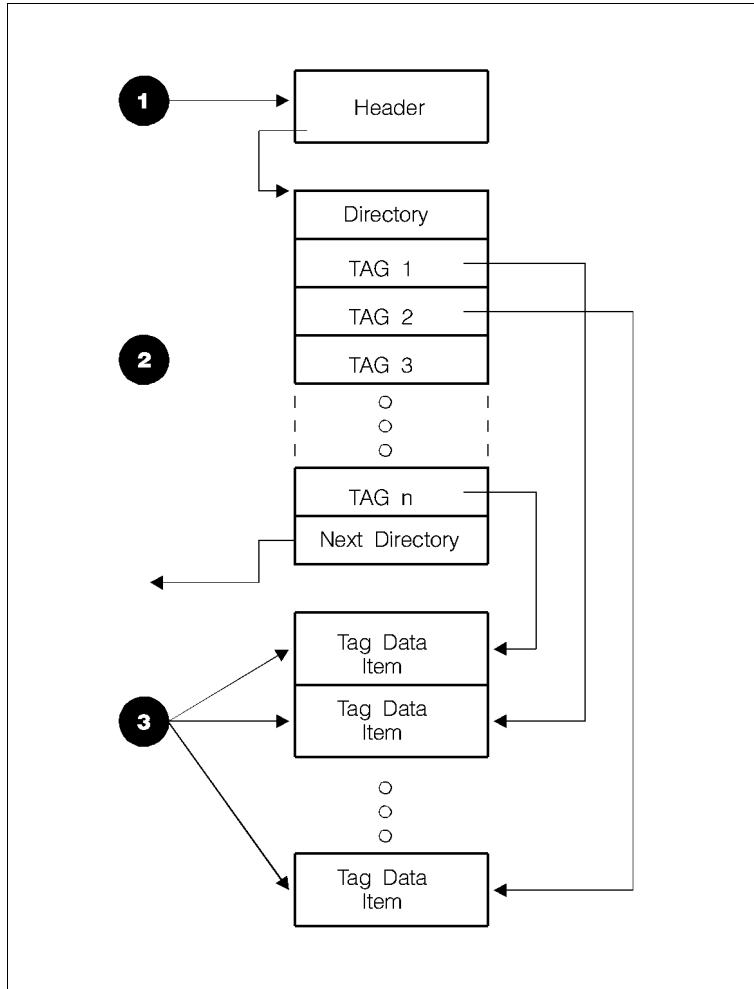


Figure 6-1. TFM File Structure

Virtually all of the information in the file is accessed through 32-bit offsets relative to the beginning of the file rather than through fixed locations.

The file structure is intended for use in two computing environments: the 8086 and the 68000 microprocessor family. The difference between the two processors is the byte order in which integer values are stored. The first data structure in the file indicates what format to expect.

Header Structure

There is only one header structure in the file, and it is the first thing in the file. This structure contains data located in an absolute file position. Figure 6-2 shows the byte structure of the header and the following explanation describes the header fields.

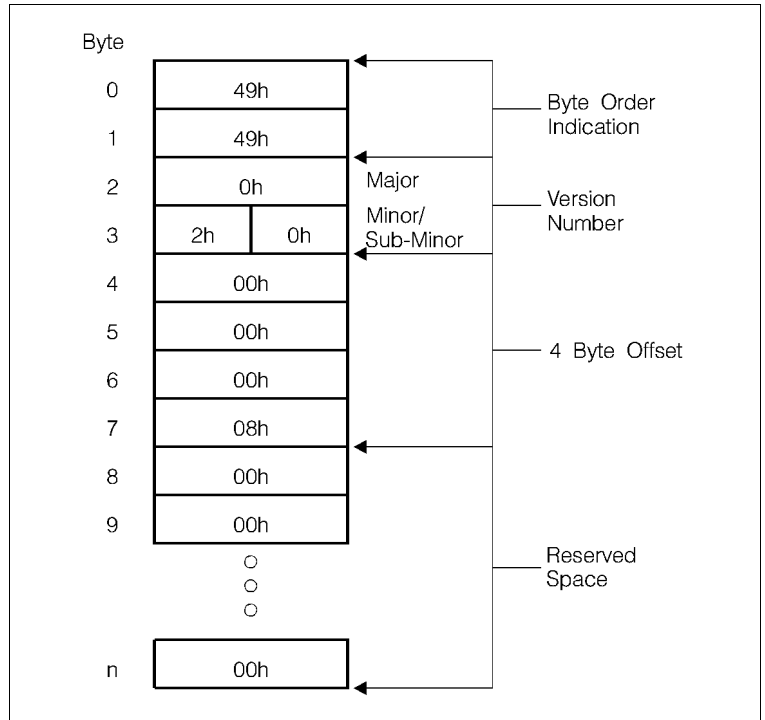


Figure 6-2. TFM Header Structure

- **Bytes 0-1: Byte Order**

The first two bytes of the file indicate the byte order for the rest of the file.

byte 0 = 49h, byte 1 = 49h

or

byte 0 = 4Dh, byte 1 = 4Dh

These two values indicate the byte ordering of data in the file. The character 49h, ASCII character I, indicates that the format is intended for the Intel family of processors where the most significant bytes follow the least significant bytes.

The character 4Dh, ASCII character M, indicates that the format is intended for the Motorola family of processors where the least significant bytes follow the most significant bytes.

- **Byte 2: Major Format Version**

This byte is used to express the major version number of the file format. For example, Version 1.2.0 would be indicated with a 01h in this byte.

If new data types are added the major version number should change. Development versions should always start with the most significant bit set to one. It is up to an application writer to choose whether to accept or ignore a version under development.

- **Byte 3: Minor and Sub-Minor Format Version**

This byte is broken into two equal parts. The upper four bits represent the minor version number and the lower four bits represents the sub-minor version number. For example, version 1.2.0 would be indicated with a 20h in this byte.

If new tag values are defined for the format the minor version number should change. If data or tags are being corrected then the sub-minor version number should change. It should be noted that the sub-minor version number indicates a correction. Version 1.2.0 and 1.2.15 files may show

different outputs but the definitions for the tags and data types in both are the same.

- **Bytes 4-7: Directory Offset**

These four bytes in the header are used as a 32-bit offset from the beginning of the file to the first directory entry in the file. The first byte in the file is at offset zero (0). All entries in the file should be placed on word boundaries. In this way all offset values in the file should be even. An odd value would indicate that an error has occurred.

- **Bytes 8-n: Reserved Space**

This area of the header is reserved for font vendor use only. Reserved space will always terminate on a word boundary. This allows the first directory entry in the file to be on a word boundary. This area is set aside to allow font vendors to place internal information, such as a file version number, into the TFM file.

Directory Structure

Figure 6-3 shows the structure of a Directory Entry. Each directory consists of a Tag Entry Count, one or more tag entries, and an offset to the next directory in the file.

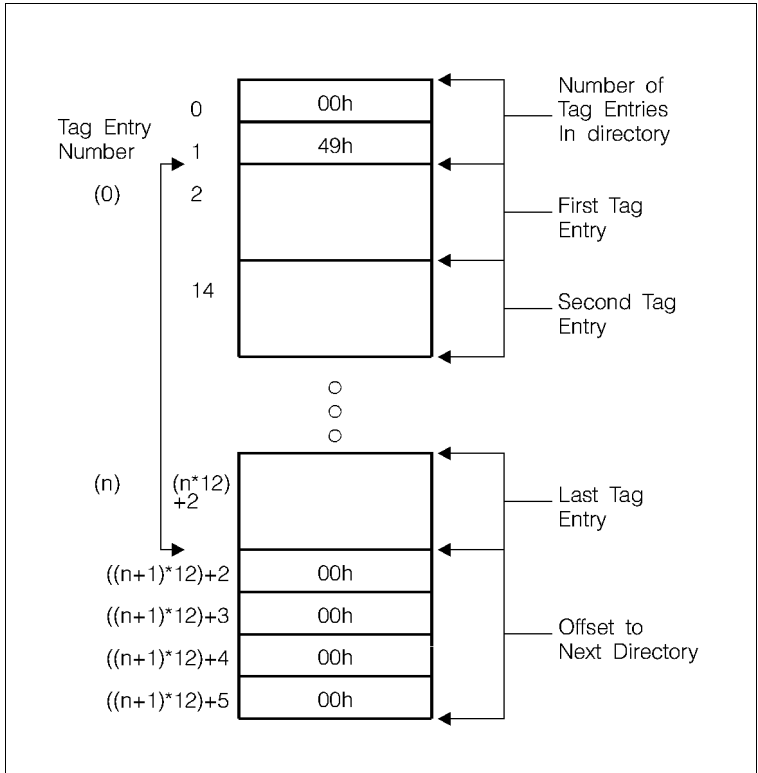


Figure 6-3. TFM Directory Structure

- **Bytes 0-1: Number of Tag entries**

The first 2 bytes in a directory contain a 16-bit integer value indicating the number of Tag Entries to expect in the directory. The Tag Entries immediately follow the first 2 bytes of the directory, and they must be ordered sequentially, based on the tag value. Each tag occupies 12 bytes.

Following the last Tag Entry is a 4-byte file offset indicating the location of the next directory in the file. If the value is zero, then there are no more directories in the file.

The size of the directory structure can be determined with the following equation:

$$m = \text{number of tag entries}$$

$$(m*12)+6 = \text{size in bytes}$$

Tag Entry Structure

Figure 6-4 shows the structure of a Tag Entry. The fields are defined as follows:

- **Bytes 0-1: Tag Value**

These two bytes contain a 16-bit, unsigned integer value specifying the type of data associated with the tag entry, e.g., 423 = Capheight, 425 = Ascent, 411 = Stroke Weight, and so forth.

- **Bytes 2-3: Data Type**

This is another 16-bit unsigned integer value used to specify the type of data to expect in the data field. Current data types are defined in the “Data Types” table on the next page.

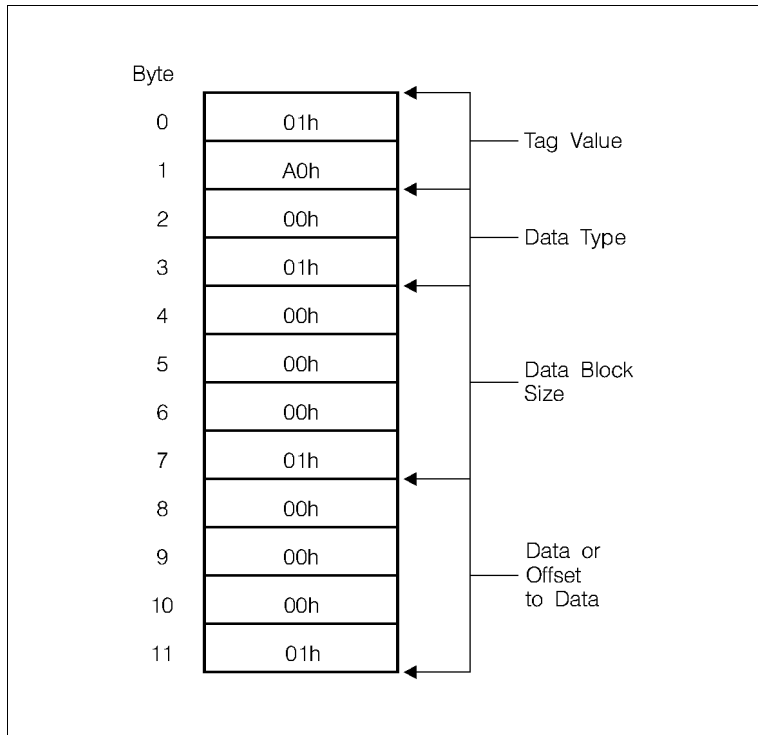


Figure 6-4. Tag Entry Structure

- **Bytes 4-7: Data Block Size**

This is a 32-bit unsigned value indicating the size of the data block associated with the tag. The size of the data block is the units of the data type.

Data Types

Value	Data Type	Description
1	BYTE	8-bit unsigned integer.
2	ASCII	Null terminated array of 8-bit bytes.
3	SHORT	16-bit (2 bytes) unsigned integer.
4	LONG	32-bit (4-bytes) unsigned integer.
5	RATIONAL	Two LONG's (8-bytes) where the first long represents the numerator of a fraction and the second the denominator.
16	SIGNED BYTE	8-bit 2's complement signed integer.
17	SIGNED SHORT	16-bit 2's complement signed integer.
18	SIGNED LONG	32-bit 2's complement signed integer.

For example: if the data type indicated a SHORT integer and the data block size is 8 then the data consumes 16 bytes (8 * the length of a SHORT integer).

The length of the ASCII data type includes the terminating null character. The string length is actually one less than the size indicated.

If the calculated data block size is odd, the block should be padded with a null byte to align the data word boundaries.

- **Bytes 8-11 = Data / Offset**

The last 4-byte segment of the structure is either a file offset to the data in the file or it is the data, if the data will fit within the 4 bytes (e.g. If the data length * sizeof(data type) is less than or equal to 4, then the data is contained here.).

TFM Usage

To successfully read a tagged file format, an application must recognize two things:

- A set of tag values
- Data types supported in the file.

There are three types of tags which may be found in a TFM file:

- Level 1 - Required
- Level 2 - Optional (this information can be derived from level 1 tags)
- Level 3 - Optional (this tag contains information general to the font or TFM file; it has no impact on the metric information within the file)

TFM Tag Descriptions

Numeric values in the following definitions are given as decimal values. Unless otherwise specified all metric units used are Design Units. Design Units are defined by tag 408.

- **Subfile Type (Level 1)**

Tag = 400 (190h)

Type = SHORT

Length = 1

This tag describes how to use the symbol map (Tag 403). For the value zero, the characters are mapped to the Master Symbol List. For the value one, they are mapped to the symbol set they are representing. For the value two, characters are mapped to a Unicode list. The types are listed below:

0 — MSL list

1 — Symbol Set list

2 — Unicode list

- **Copyright information (Level 3)**

Tag = 401 (191h)

Type = ASCII

Length = # of bytes in the ASCII string plus one null byte.

This field contains copyright information, trademark information, terms of use, etc.

- **Comment Information (Level 3)**

Tag = 402 (192h)

Type = ASCII

Length = # of bytes in the ASCII string plus one null byte.

This field contains comment information.

Font Identification

- **Symbol Map (Level 1)**

Tag = 403 (193h)

Type = SHORT

Length = # of Characters contained in the file.

This tag specifies the number of symbols contained in the font file and the array of symbols in the TFM file as mapped by tag 400.

- **Symbol Set Directory (Level 1)**

Tag = 404 (194h)

Type = SHORT

Length = # of directory entries * 14 bytes.

This tag specifies what symbol sets have been predefined and can be derived from the symbols available in the font file. The directory structure can be seen in Figure 6-5. There may be one or more directories in the file with each directory consisting of three offsets and an integer value. Each element of the structure is described as follows:

- **Symbol Set Name Offset (LONG)**. These four bytes are an offset to an ASCII string which identifies the symbol set being defined.
- **Symbol Set Selection String Offset (LONG)**. This is a four-byte offset to a device-specific selection string based on the symbol set. For PCL printers, the string would contain the two unique characters used in symbol set selection.
- **Symbol Set Index Array Offset (LONG)**. This is a four byte offset to an array that contains indices from the symbol set to the symbol map. The array should be the length of the associated symbol set. An array element is used to locate the metrics for the character with the same index in the symbol set.
- **Array Length (SHORT)**. This value indicates the length of the symbol set index array.

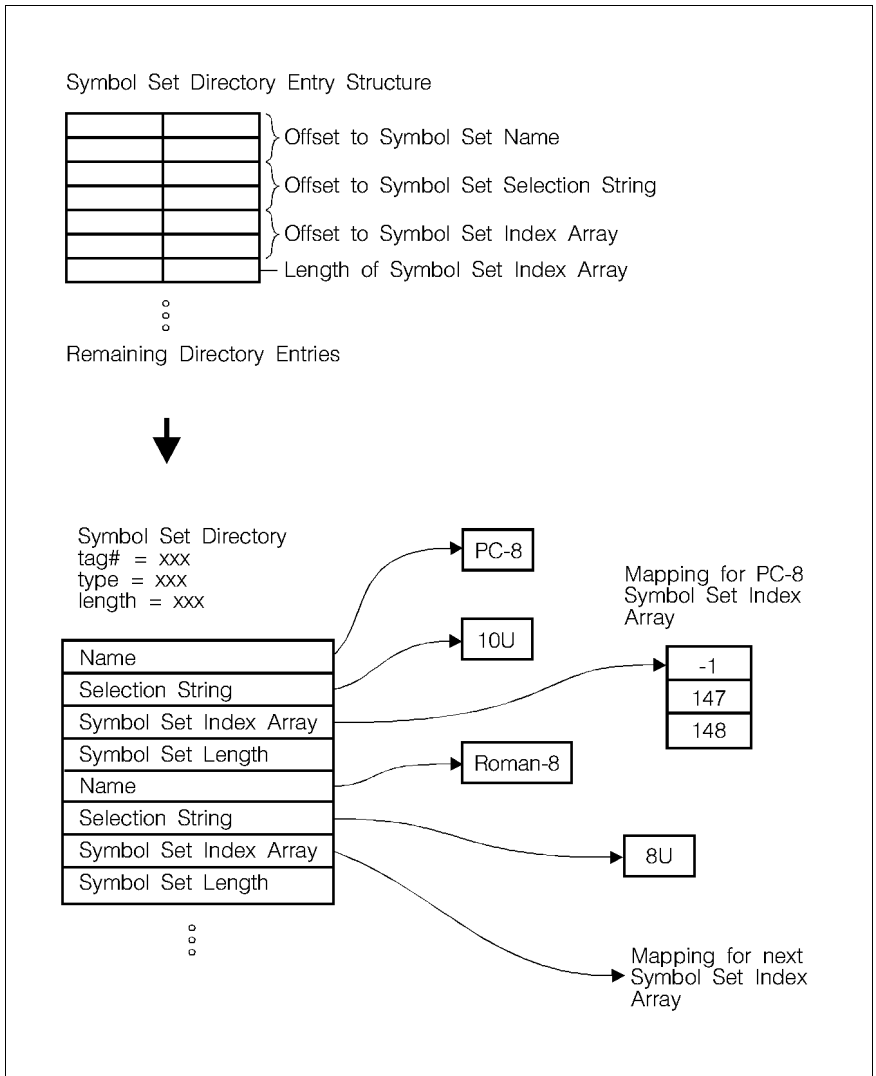


Figure 6-5. Symbol Set Directory Entry Structure

- **Unique Association ID (Level 3)**

Tag = 405 (195h)

Type = ASCII

Length = 13

This tag defines a method for “stamping” a metric file and the associated font file with a unique ID. The format of the stamp is an ASCII string containing the following: the year, month, day, hour, minute, and a two-digit random number. For example, 881018010601 would translate to 1988, October, the eighteenth, at 1:06 AM with a random number of 01.

- **Point (Level 1)**

Tag = 406 (196h)

Type = RATIONAL

Length = 1

This tag is used to specify the “exact” size, in inches, of the point being referred to in this file. Depending on the type designer, a point may be 1/72 inch, 1/72.307 inch, or any number of other sizes.

- **Nominal Point Size (Level 1)**

Tag = 407 (197h)

Type = RATIONAL

Length = 1

This tag is where the optimal display point size for the associated font should be indicated. This is the point size for which the units in the metric file are exact.

- **Design Units (Level 1)**

Tag = 408 (198h)

Type = RATIONAL

Length = 1

The Design Units are the same units used in the remainder of the file for the font metrics. The Design Units are the number of units used per the Nominal Point Size. See “Example: Using TFM Values in Intellifont Calculations” for an example of converting Design Units to physical units.

- **Type Structure (Level 1)**

Tag = 410 (19Ah)

Type = BYTE

Length = 1

This tag is a means for indicating the surface treatment of the typeface.

Structure	Tag Value
Solid	0-7
Outline	8-15
Inline	16-23
Contour	24-31
Solid with Open Shadow	32-39
Open with Solid Shadow	40-47
Inline with Shadow	48-55
Contour with Shadow	56-63
Pattern 1	64-71
Pattern 2	72-79
Pattern 3	80-87
Pattern 4	88-95
Pattern 1 with Shadow	96-103
Pattern 2 with Shadow	104-111
Pattern 3 with Shadow	112-119
Pattern 4 with Shadow	120-127
Inverse	128-135
Inverse w/ Open Border	136-143
Reserved	144-255

“Solid” is by far the most common surface treatment. A solid typeface has a surface of one uniform color. There are no features of the surface to give the typeface any special effects.

The next most common surface treatment is “outline”. The surface of an outlined letter appears identical to the background of the page.

- **Stroke Weight (Level 1)**

Tag = 411 (19Bh)

Type = BYTE

Length = 1

Stroke weight describes the thickness of the strokes that compose the characters in a typeface.

Stroke Weight	Tag Value
Ultra-Thin	0-17
Extra-Thin	18-34
Thin	35-51
Extra-Light	52-68
Light	69-85
Demi-Light	86-102
Semi-Light	103-119
Medium	120-136
Semi-Bold	137-153
Demi-Bold	154-170
Bold	171-187
Extra-Bold	188-204
Black	205-221
Extra-Black	222-238
Ultra-Black	239-255

The values are assigned as a range rather than as a specific number. This allows other systems of weights to be mapped into this range.

- **Spacing (Level 1)**

Tag = 412(19Ch)

Type = SHORT

Length = 1

Whether a font's spacing is fixed or proportional should be indicated here.

If the value is zero then the font is proportionally spaced. Any value other than zero indicates that the font is fixed pitch and the value in this field is the pitch given in Design Units.

- **Slant (Level 1)**

Tag = 413 (19Dh)

Type = SIGNED SHORT

Length = 1

A slant is given in 1/100 degrees. No slant is indicated by a zero value. A positive value is clockwise from the vertical and a negative value is counter-clockwise.

- **Appearance Width (Level 1)**

Tag = 414(19Eh)

Type = BYTE

Length = 1

This tag is a means for indicating that the font is designed as a variation on what would be considered the normal appearance for the typeface. The type style is the same, but the font appears to be either wider or narrower than usual for this style of typeface.

Width	Tag Value
Ultra-Ultra-Condensed	0-20
Ultra-Condensed	21-47
Extra-Condensed	48-74
Condensed	75-101
Semi-Condensed	102-128
Normal	129-155
Semi-Expanded	156-182
Expanded	183-209
Extra-Expanded	210-236
Ultra-Expanded	237-255

- **Serif Style (Level 1)**

Tag = 415(19Fh)

Type = BYTE

Length = 1

This field classifies the serif style.

Serif Style	Value
Sans Serif Square	0-17
Sans Serif Round	18-36
Serif Line	37-55
Serif Triangle	56-74
Serif Swath	75-93
Serif Block	94-112
Serif Bracket	113-131
Rounded Bracket	132-150
Flair Stroke	151-169
Script Non-connecting	170-188
Script Connecting	189-207
Script Calligraphic	208-226
Script Broken Letter	227-255

- **Type Style**

Tag = 416(1A0h)

Tag 416 is obsolete.

- **Typeface (Level 2)**

Tag = 417(1A1h)

Type = ASCII

Length = # of bytes in the ASCII string plus one null byte.

This tag may be used to provide a name that the end user will recognize and use to select a font.

- **Typeface Source (Level 3)**

Tag = 418 (1A2h)

Type = ASCII

Length = # of bytes in the ASCII string plus one null byte.

This tag contains the name of the font designer or supplier.

- **Average Width (Level 2)**

Tag = 419(1A3h)

Type = RATIONAL

Length = 1

Average width is calculated by first multiplying the escapement value of each lowercase character in the typeface by the weights listed in the following table, then taking the sum of all these products and dividing the sum by 1000:

$((\text{escapement value for "a"} * 64) + (\text{escapement value of "b"} * 14) + \dots + (\text{escapement value for "space"} * 166)) / 1000$

Character	Weight	Character	Weight
a	64	o	56
b	14	p	17
c	27	q	4
d	35	r	49
e	100	s	56
f	20	t	71
g	14	u	31
h	42	v	10
i	63	w	18
j	3	x	3
k	6	y	18
l	35	z	2
m	20	space	166
n	56		

These weights are based on the frequency of use in the English language.

- **Maximum Width (Level 2)**

Tag = 420(1A4h)

Type = SHORT

Length = 1

This is the largest Blackwidth of any character in the font (Blackwidth is the distance between the Left Extent and the Right Extent as shown in Figure 6-7).

Font Parameters

Font parameters pertain to the font as a whole rather than individual characters (see Figure 6-6 below).

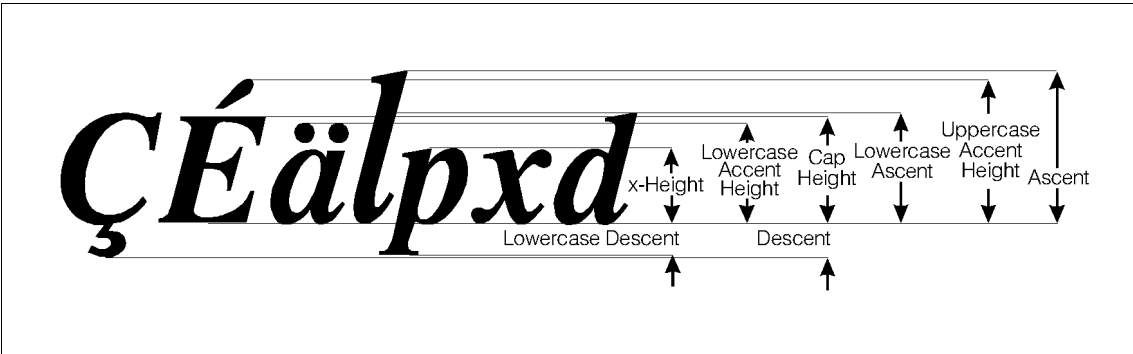


Figure 6-6. Font Metrics

- **Interword Spacing (Level 2)**

Tag = 421 (1A5h)

Type = SHORT

Length = 1

This is the recommended spacing to use between words. Normally this is equivalent to the width of the space character. If a space character is not defined, use a space that is at least equal to the lowercase i, and no greater than the 1 EN which is generally the width of the uppercase N or 1/2 EM.

- **Recommended Line Spacing (Level 2)**

Tag = 422(1A6h)

Type = SHORT

Length = 1

This is the tightest recommended line spacing for the text, baseline to baseline. If this tag is not present, the recommended line spacing for the same text point size is the point size plus 20%.

- **Capheight (Level 2)**

Tag = 423 (1A7h)

Type = SHORT

Length = 1

The height of the uppercase letters in the font. Usually the height of the capital H.

- **x-height (Level 2)**

Tag = 424(1A8h)

Type = SHORT

Length = 1

x-height is the height of the lowercase letters in the font. It is usually specified as the height of the lowercase x.

- **Ascent (Level 2)**

Tag = 425(1A9h)

Type = SHORT

Length = 1

The distance from the baseline to the highest printed mark of any character in the font.

- **Descent (Level 2)**

Tag = 426(1AAh)

Type = SIGNED SHORT

Length = 1

The distance from the baseline to the lowest printed mark of any character in the font.

- **Lowercase Ascent (Level 2)**

Tag = 427 (1ABh)

Type = SHORT

Length = 1

The distance that the ascenders of lowercase letters extend above the baseline. This value is typically specified for a lowercase d.

- **Lowercase Descent (Level 2)**

Tag = 428 (1ACh)

Type = SIGNED SHORT

Length = 1

The distance that the descenders of lowercase letters extend below the baseline. This value is typically specified for a lowercase p.

- **Underscore Depth (Level 1)**

Tag = 429(1ADh)

Type = SIGNED SHORT

Length = 1

This is the distance from the baseline to the center of the underscore. A negative number states the underscore is below the baseline, and a positive number states the underscore is above the baseline.

- **Underscore Thickness (Level 1)**

Tag = 430 (1AEh)

Type = SHORT

Length = 1

This is the thickness of the underscore.

- **Uppercase Accent Height (Level 2)**

Tag = 431 (1AFh)

Type = SHORT

Length = 1

This is the highest point above the baseline at which any printed mark of an uppercase accent mark will appear.

- **Lowercase Accent Height (Level 2)**

Tag = 432 (1B0h)

Type = SHORT

Length = 1

This is the highest point above the baseline at which any printed mark of a lowercase accent mark will appear.

Character Parameters

Refer to Figure 6-7 for the tag values described in this section.

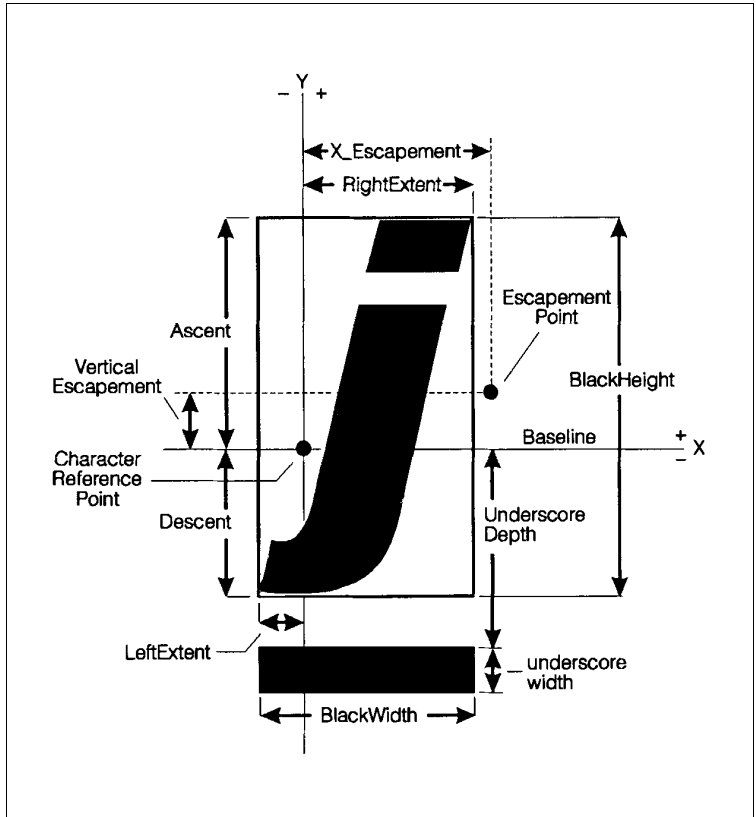


Figure 6-7. Character Metrics

- **Horizontal Escapement (Level 1)**

Tag = 433(1B1h)

Type = SHORT

Length = # of characters

This is the recommended horizontal offset, in Design Units, from the Character Reference Point (CRP), to the horizontal location of the next character in sequence, assuming the same point size and same typeface. A right escapement is a positive value while a left escapement is a negative value. This value is normally used in conjunction with the Vertical Escapement to locate the CRP for the next character.

- **Vertical Escapement (Level 1)**

Tag = 434(1B2h)

Type = SHORT

Length = # of characters

This is the recommended vertical offset, in Design Units, from the CRP, to the vertical location of the next character in sequence, assuming same point size and same typeface. A positive value indicates an upward escapement while a negative value indicates downward escapement. This value is normally used in conjunction with the Horizontal Escapement to locate the CRP for the next character.

- **Left Extent (Level 1)**

Tag = 435 (1B3h)

Type = SIGNED SHORT

Length = # of characters

This is the position of the leftmost extension in the character's design relative to the CRP.

- **Right Extent (Level 1)**

Tag = 436 (1B4h)

Type = SHORT

Length = # of characters

This is the position of the rightmost extension in the character’s design, relative to the CRP.

- **Character Ascent (Level 1)**

Tag = 437 (1B5h)

Type = SHORT

Length = # of characters

This is the position of the topmost extension in the character’s design, relative to the CRP.

- **Character Descent (Level 1)**

Tag = 438 (1B6h)

Type = SIGNED SHORT

Length = # of characters

This is the position of the lowest extension in the character’s design, relative to the CRP.

Kerning Information

This section lists the tags related to kerning. For general information about kerning, see the “Kerning” discussion in Appendix G.

- **Kern Pairs (Level 2)**

Tag = 439(1B7h)

Type = SHORT

Length = # of character pairs*3+1

This tag is used as a structure containing the following data:

- # Pairs (SHORT). The first two bytes form an unsigned integer (SHORT) value indicating the number of kern pairs defined.

The following structures contain the data for each of the kern pairs. It is repeated for each pair.

- Char1 (SHORT). This is the index value for the first character in the pair.
- Char2 (SHORT). This is the index value for the second character in the pair.

- Kern (SIGNED SHORT). This is a signed integer value indicating the recommended amount by which to kern the two characters. The kern value is added to the escapement value of the first character. A positive value means increased spacing and negative means decreased spacing. Figure 6-8 shows the structure of this tag data.

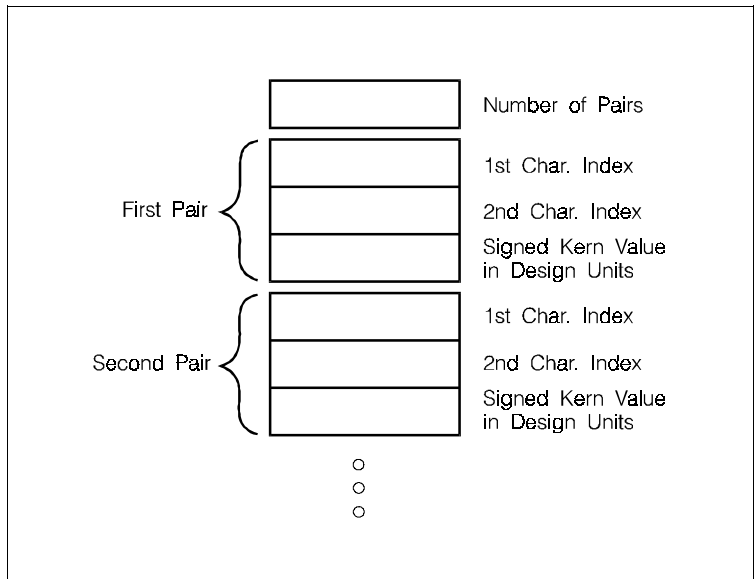


Figure 6-8. Pair Kern Information

- **Sector Kern Information (Level 2)**

Tag = 440(1B8h)

Type = SHORT

Length = # of characters * (number of sectors per character * 2 + 1) + 2

This tag is also used as a structure and is similar to the pair kern structure. This structure contains the following data:

- # Characters (SHORT). The first two bytes form an unsigned integer value indicating the number of characters for which sector kern data is assigned.
- # Sectors (SHORT). An unsigned integer value indicating the number of sectors to expect with each character.

The following structures are repeated for each character that has sector kern data.

- CharIndex (SHORT). An unsigned integer value used as an index to designate the character whose sector information follows.
- SectorLeft (SIGNED SHORT). This is an array of signed integers containing the sector values for the left side of the character. The length of the array is defined by the # Sectors value set above. The first element in the array corresponds to the sector at the top of the character. That value is followed by the next sector and the sequence continues on down through the array. Positive values indicate kerning to the right and negative to the left.
- SectorRight (SIGNED SHORT). The array for the right sector values immediately follows the array for the left side and has exactly the same format.

Figure 6-9 illustrates the Sector Kern structure.

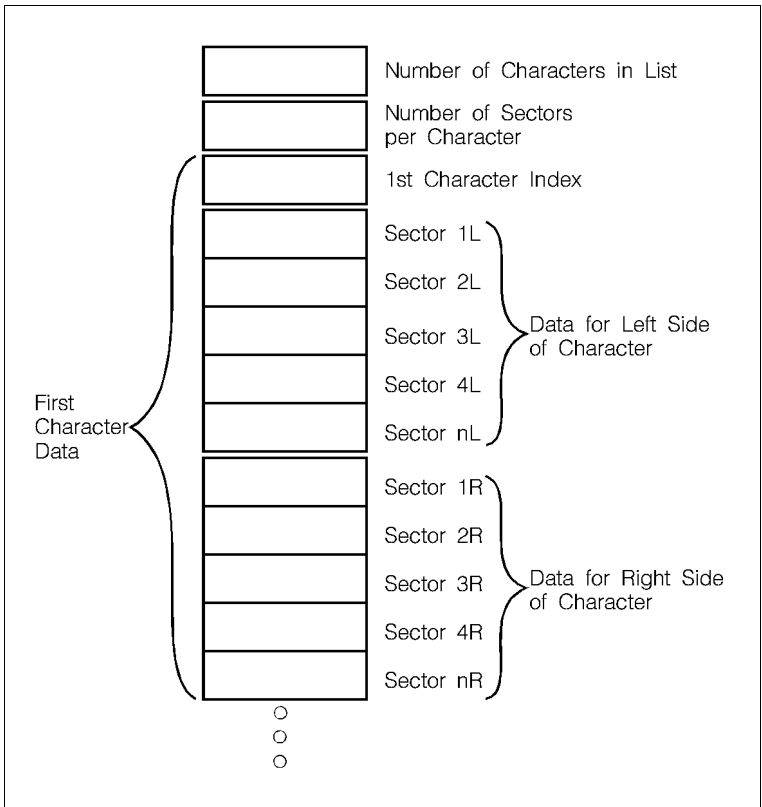


Figure 6-9. Sector Kern Information

- **Track Kern Information (Level 2)**

Tag = 441(1B9h)

Type = SHORT

Length = # of tracks * 5 + 1

This tag is structured as follows:

- #Tracks (SHORT). This is an unsigned integer value indicating the number of tracks which are defined.

The following structures are repeated for each track that is defined. The tracks must be ordered from the tightest to loosest kerning.

- #TValue (SIGNED SHORT). This is a signed integer value indicating the relative tightness of the track. The larger the negative number, the tighter the kerning.
- MaxSize (SHORT). This is an integer value, in Design Units, which indicates the largest type size to which this track kern information is applied.
- MinSize (SHORT). This is an integer value, in Design Units, which indicates the smallest type size to which this track kern information is applied.
- MaxKern (SIGNED SHORT). This is a signed integer value which indicates the maximum extent to which characters may be kerned with this track.
- MinKern (SIGNED SHORT). This is a signed integer value which indicates the minimum extent to which characters may be kerned with this track.

Figure 6-10 illustrates the Track Kern Structure. The result of the track kern algorithm, described under the section about kerning, is added to the escapement value of each character in the string being kerned. Positive values increase the spacing and negative decrease it.

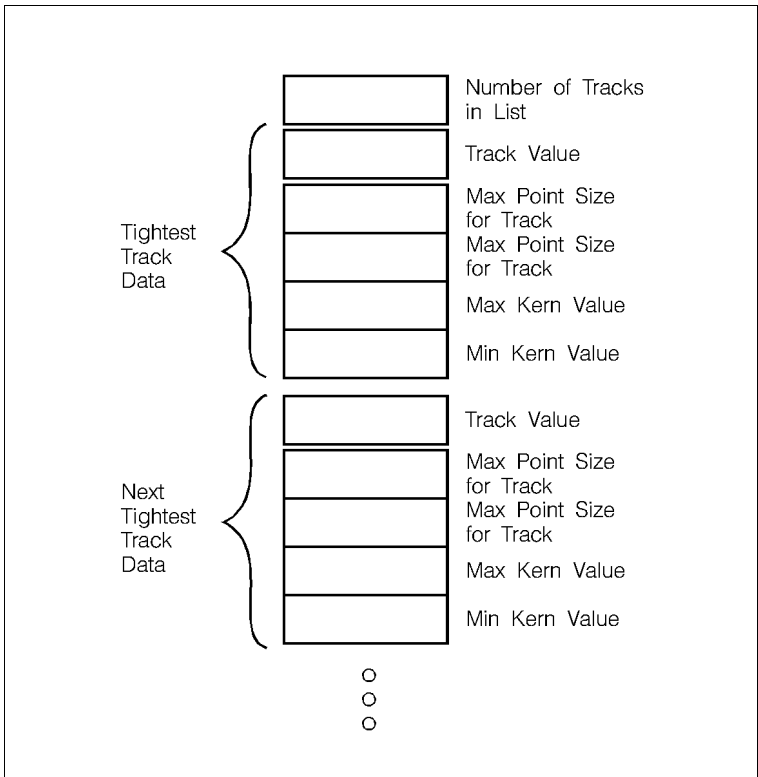


Figure 6-10. Track Kern Information

Device Data • **Typeface Selection String (Level 1)**

Tag = 442 (1BAh)

Type = ASCII

Length = # of bytes in the ASCII string plus one null byte.

This tag defines the value needed to select the correct typeface from a PCL printer. The string matches identically what must be sent to a PCL printer in the escape sequence `?(s#T`. NOTE: Some of the values may be two-byte values. The LaserJet IID, IIP, and later printers all support a two-byte value for typeface selection. A LaserJet series II and previous printers only recognize a one-byte value, which is the lower byte of the two byte value (modulo 256).

PANOSE Numbers

- **PANOSE Numbers (Level 3)**

Tag = 443 (1BBh)

Type = Byte

Length = 10

This tag contains the ten-byte PANOSE Classification number which is used to describe the visual characteristics of a given typeface. These characteristics are used to associate a typeface with other similar typefaces that have different names. The names for these ten bytes are given below. Detailed specifications for these bytes can be obtained in the *PANOSE Classification Guide* from Elseware Corporation (206-632-3300).

Type	Name
BYTE	bFamilyType;
BYTE	bSerifStyle;
BYTE	bWeight;
BYTE	bProportion;
BYTE	bContrast;
BYTE	bStrokeVariation;
BYTE	bArmStyle;
BYTE	bLetterform;
BYTE	bMidline;
BYTE	bXHeight;

“Glue” File Description and Usage

A file called GLUE.TXT is used to form the link between the printer font and its associated TFM file. The “glue” file is an ASCII based file, consisting of lines up to 80 characters in length. Each line is terminated by a CR/LF pair (value 0Dh, 0Ah (13, 10)).

This file should be located in the \AUTOFONT directory.

File Format

In general, data begins in the first byte of the line. The following types of lines are recognized:

- “ ” (Blank lines). Blank lines are ignored.
- “;” (Comments). A semi-colon in any position on a line indicates the rest of the line is a comment.
- “[]” (Category headings). Category headings represent general information about the type of printer and type of printer files. The following categories are used:
 - [Cartridge] - Cartridge fonts
 - [PCL fonts] - PCL soft fonts
 - [Printer] - Internal printer fonts
- “{ }” (Sub-category headings). Sub-category headings provide information about the class of printer files being used. The following sub-categories are used:

Printer Class	Description
A	LaserJet bitmap fonts
B	DeskJet fonts
C	Compressed Bitmap fonts
D	PaintJet XL fonts
E	Dot matrix printer fonts
F	Reserved
O	PCL Encapsulated Outline scalable fonts
P	PostScript fonts

- “XXXXXX = ” (Parameters). Parameters begin with a keyword (case insensitive), followed by a space, equal sign, space, and data. The data consists of one or more ASCII strings. The items are separated by at least one blank and may be scanned in with the C “scanf” command. Each parameter is followed by sub-parameters and ends with the next parameter, category, sub-category, or end of file. The following parameter is used:
 FONT = typeface family name
- “/yyyy = ” (Sub-parameters). Each sub-parameter begins with a slash (/) followed by the keyword, space, equal sign, space, and data. The data consists of one or more ASCII strings, separated by at least one blank. All sub-parameters are part of the parameter they are under until a new parameter is entered.

Font Entries

Each font is represented by an entry in the appropriate sub-category. The entry consists of one parameter and a collection of sub-parameters, each on its own line.

The font entry commences with a “FONT” parameter. All following sub-parameters are considered part of the FONT entry, until the next parameter is reached, or the end of the sub-category, category, or file is reached.

The following are the parameters and sub-parameters found in the font entry. The first parameter, FONT, is required for each entry. The other sub-parameters may or may not be present (Required sub-parameters will vary by font class).

- FONT - Will be followed by “ = ” and the typeface family name.
- /family - The typeface family number.

Typeface Family	Value
SANS SERF ROUND	0000-0039
SANS SERF EVEN PROP	0040-0399
SANS SERF OLD STYLE	0400-0645
SANS SERF DECO	0646-0799
MOD SANS/FLARE SERF	0800-1029
RND/TRIANGLE BRACKT	1030-1099
BRKT SERF OLD STYLE	1100-1145
BRACKET OLD SPARE	1146-1395
BRACKT SERF EVN PROP	1396-1449
BRACKET SERF EVN SPARE	1450-1705
BRACKET SERIF MODERN	1706-1827
SERF MODRN, DECO SPARE	1828-1995
BRACKET SERIF DECO	1996-1999
LINE SERIF	2000-2149
SQUARE SERIF	2150-2299
NON-CONN SCRIPT	2300-2353
CONN SCRIPT	2354-2409
CALLIG SCRIPT	2410-2429
BROKEN LETTER SCRIPT	2430-2459

- /resource - Indicates the format or location of the font.
The first entry after the “ = ” is a single character:

D = downloadable soft font

I = internal font (resident in printer)

C = cartridge

Other data may follow on the same line. For fonts of type D, the next string on the same line is the number of bytes of printer memory required by the font. Cartridge fonts have a unique cartridge identifier as the next string of data followed by a string containing the cartridge name. The following table contains the cartridge identification numbers. Internal fonts have a printer name as the next string of data.

Cartridge ID Number	Cartridge
1	A
2	B
3	D
4	E
5	C
6	L
7	F
8	H
9	G
10	N
11	M
12	Q
13	P
14	J
15	U
16	V
17	W
18	Y

Cartridge ID Number	Cartridge
19	X
20	R
23	Z
24	K
25	W1
26-28	Pro Collection
240	Tax
1200	S1
1201	S2
1202-1203	Polished Worksheets
1204-1205	Persuasive Presentations
1206-1207	Word Perfect
1208	Math Scientific
1209-1210	Forms
1211	Bar codes
1212-1213	Global Text
1214	Great Start
1215	International Collection
1300	Distinctive Documents I/ Compelling Publications I
1301	Brilliant Presentations I/ Compelling Publications II
1302	WordPerfect scalable cartridge

- /font file - This sub-parameter always appears for downloadable soft fonts. It contains the path and filename of the corresponding soft font.
- /symset - Symbol set. The first item is a PCL symbol set selection string code which indicates the mapping of the font. The second item is a number representing the classification of the symbol set. NOTE: This entry can occur multiple times within the entry. This allows an entry to represent a collection of identical fonts except that each uses a different symbol set.

PCL Symbol Set Selection String Code	Value
LATIN_TEXT_SET	1
INTL_TEXT_SET	2
PROPER_QUOTE_SET	4
DOUBLE_QUOTE_SET	8
MATH_SET	16
LEGAL_SET	32
LINE_SET	64
PC_LINE_SET	128
P1_SET	256
DINGBATS_SET	512
OCR_SET	1024
BARCODE_SET	2048
SPECIAL_SET	4096
TAXLINE_SET	8192

Each symbol set is broken up into a subset of the set of items listed above. For example, take a classification number “9”, represented in the /symset sub-parameter of the “glue” file. The 9 breaks up into 8 + 1 (always break the number up into numbers of base 2

origin). The 8 represents the DOUBLE QUOTE SET and the number 1 represents the LATIN TEXT SET. Therefore, the symbol set being represented is a double quote and Latin text symbol set. Another example: the classification number is 4101 = 4096 + 4 + 1 which means a special, proper-quote, Latin text symbol set is being used by the font.

- /orient - Orientation of the font. One or more of the strings “P” = portrait, “L” = landscape, “RP” = reverse portrait, and “RL” = reverse landscape will be included. For downloadable outlines, this subparameter is irrelevant.
- /ptsize - Point size of the font, not the size of the point as is in the point tag (tag 406). This sub-parameter is not used if the font is scalable. NOTE: If the entry is representing a collection of identical fonts except in different point sizes, this sub-parameter would contain the different point sizes, with a space dividing each. For example, if the entry is to represent CG Times in 8 and 10 point, the sub-parameter would be “/ptsize = 8 10”.
- /tfm - The path and filename of the TFM metrics file which pertains to the font referenced by the FONT parameter.
- /weight - Stroke weight, 0-255, as in TFM file. If this sub-parameter is not present, the typeface is assumed to be medium weight.
- /slant - If the value is 0 or not present, the font is upright. If the value is positive, the font is italic. If the value is negative, the font is left italic.
- /typeface - The full typeface name up to 50 characters.
- /type file - The path and filename of the typeface data source file. This sub-parameter is used by applications with font scaling capabilities. The data source file is only useful if the file located at this sub-parameter location is compatible with the font scaling method. For example, if this sub-parameter points to a .TYP file then applications with the Intellifont font scaling method can use this file to create screen fonts or printer fonts.

NOTE: For all scalable font entries, this sub-parameter is provided. For non-scalable font entries, this sub-parameter is provided if the data source file is known.

- /class - Font class. This is usually unnecessary, because soft fonts are grouped in sub-categories by class. However, it may be used in other categories or sub-categories, such as for resident printer fonts or cartridge fonts.

Font Class	Description
A	LaserJet bitmap fonts
B	DeskJet fonts
C	Compressed Bitmap fonts
D	PaintJet XL fonts
E	Dot matrix printer fonts
O	PCL Encapsulated Outline scalable fonts
P	PostScript fonts

The following listing shows a sample glue file:

```
[PCL Fonts]
{A} ;LaserJet bitmap fonts
FONT = CG Palacio
/family = 1142
/resource = D 6588
/weight = 179
/slant = 1
/font file = c:\TD2\FONTS\PAJOGUSA.SFP
/symset = OU 5
/orient = P
/ptsize = 4.00
/type file = c:\TD2\TYPE\92535.TYP
/tfm = C:\AUTOFONT\PAJOOOOS.TFM
{O} ;PCL Encapsulated Outline scalable fonts
FONT = Antique Olv
/family = 430
/resource = D 24082
/font file = C:\TD2\FONTS\ANROOOSO.SFS
/symset = OU 5
/type file = C:\TD2\TYPE\91119.TYP
/tfm = C:\AUTOFONT\ANROOOOS.TFM
[Cartridge]
{1208 Text Equations}
FONT = CG Times
/class = A; LaserJet bitmap fonts
/family = 1100
/resource = C 1208 Text Equations
/symset = OA 16
/symset = 8M 18
/symset = 15U 256
/symset = 8u 5
/orient = P
/ptsize = 8 10
/tfm = C:\AUTOFONT\TRROOOOS.TFM
FONT = Prestige
/class = A; LaserJet bitmap fonts
/family = 3001
/resource = C 1208 Text Equations
```

/symset = 8M 18
/orient = P
/ptsize = 7.00
/tfm = C:\AUTOFONT\PRROSM8A.TFM
{1300 Distinctive Doc I Compelling Publ I}
FONT = Stymie
/class = O; PCL Encapsulated Outline scalable fonts
/family = 2172
/weight = 179
/resource = C 1300 Distinctive Doc I Compelling Publ I
/symset = 7J 5
/symset = ON 9
/symset = 1U 33
/symset = 8M 18
/type file = C:\TD2\TYPE\900067.TYP
/tfm = C:\AUTOFONT\SYSBOOOOS.TFM
[Printer]
{LaserJet III}
FONT = CG Times
/class = O; PCL Encapsulated Outline scalable fonts
/family = 1100
/resource = I LaserJet III
/symset = 9U 9
/symset = 14J 5
/symset = 6M 16
/type file = C:\TD2\TYPE\92500.TYP
/tfm = C:\AUTOFONT\TRROOOOS.TFM
FONT = Line Printer
/class = A; LaserJet bitmap fonts
/family = 2969
/resource = I LaserJet III
/symset = ON 9
/orient = L
/ptsize = 8.50
/tfm = C:\AUTOFONT\LPROYE1A.TFM

Supported Fonts

HP font products can be categorized as either *bitmapped* or *scalable*. The term *bitmapped* describes the way the font information is stored in a file. A bitmapped font has a fixed size that is represented in a file as a pattern of dots describing its shape. *Scalable* fonts are stored as “outlines” and are not limited to a specific size. The *Intellifont*[®] and TrueType scaling algorithms resident in the LaserJet 4 printer (Intellifont is resident in the other PCL 5 printers) scales the font outlines to create fonts of different sizes.

HP supplies font metric data via TFM files for all current font products:

- TFM files are supplied to the end user with the purchase of any HP font product, requiring software to provide dynamic font support for immediate compatibility.
- Font metric files for the LaserJet 4 printer internal fonts are supplied on disk and should either be hard-coded into the application driver or loaded with the software installation and read dynamically.
- All font products are shipped with the AutoFont Support Installer that easily copies the TFM files to the appropriate location on the hard disk.

TFM File Distribution

The following lists differentiate between those font products that are shipped with TFM files and those products that are not.

Font Products Shipped to Users with TFM Files Provided:

Bitmapped Fonts

- The MasterType Library of Font Cartridges—including the ProCollection cartridge, Global Text, TextEquations, Bar Codes & More, Forms Etc., Persuasive Presentations, Polished Worksheets, the Microsoft Cartridge, and the WordPerfect cartridge.

Scalable Typefaces

- All scalable typefaces—includes Type Director typeface products such as Benguiat, CG Times etc.
- All scalable cartridge typefaces.

Other Fonts (TFM Files Supplied to Software Developers):

Bitmapped Fonts

- Internal bitmapped fonts—printer-resident fonts.
- Font cartridge products such as the HP 92286A, B, C, ... Z, S1, and S2.

Scalable Typefaces

- Scalable internal typefaces—scalable typefaces resident within the PCL 5 LaserJet printers.

Note



The TFM files for all PCL 5 LaserJet printer internal typefaces and bitmapped fonts are supplied on disk, since they do not ship with the printers. To support these fonts, the application can load these TFM files into the AUTOFONT directory as part of the software installation process; the TFM data can then be read dynamically using the same procedure as with all the other fonts. (The Type Director 2.x program can also create TFM files for Intellifont scalable typefaces and for PCL bitmapped soft fonts.)

Comparing Past Font Support With TFM Support

Before the introduction of the LaserJet III printer, HP provided software developers with sample font products and with a means of incorporating spacing tables within printer drivers. Each time a new font was introduced, rewriting or creating a new printer driver was required in order to support the new font. Figure 6-11 describes the font support process prior to TFM files.

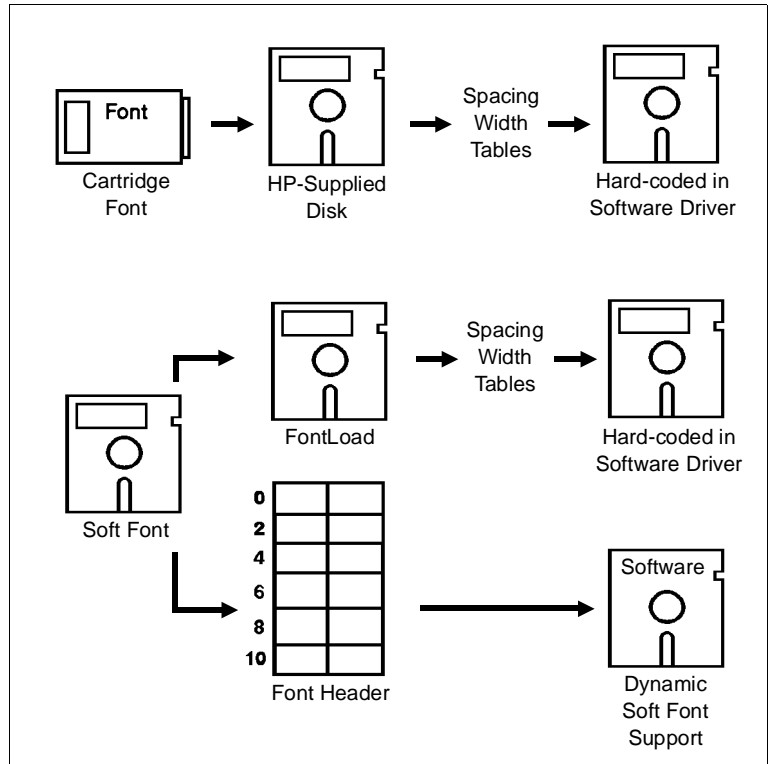


Figure 6-11. Font Metrics Before AutoFont Support

Note



For those developers using the font header for obtaining font metrics, the header for bitmapped fonts has been modified to contain more information. Although the header is backward-compatible, software developers that currently use the font header for obtaining font information should take note of the changes. Support for TFM files eliminates the need to read font headers, but those needing information about the new font header can find it in the *PCL 5 Printer Language Technical Reference Manual*.

The Benefits of Supporting TFM Files

Dynamic TFM support has several benefits:

- A standard method for access to font metrics
- A one-time resource and code investment
- Frees font driver writers to perform other tasks
- Saves code space needed for font driver updates
- Eliminates font driver distribution issues
- Ensures that the user has immediate and efficient use of font products with software, creating a higher level of customer satisfaction
- Provides automatic support of fonts from sources other than HP, if the source supplies TFM files

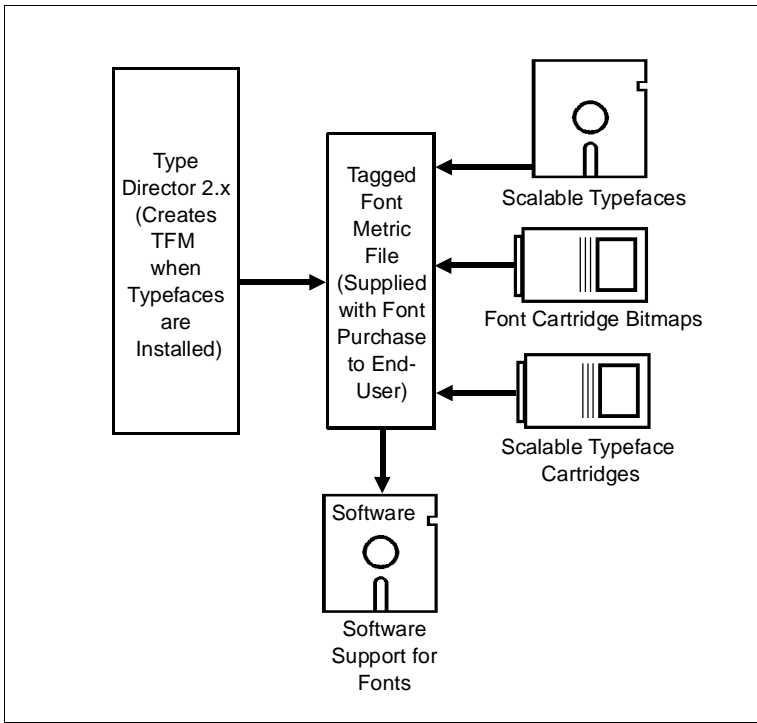


Figure 6-12. Font Metrics With TFM Support

AutoFont Support

HP recognizes that full integration of the AutoFont Support program requires a commitment from the software developer, and that it may take a while to take full advantage of it. Because of the commitment involved, and the fact that each software product has a different development schedule, there are two levels of TFM integration:

- Hard-coding TFM data
- TFM Reader integration

Hard-coding TFM Data

The process of hard-coding TFM data is very similar to the font support process for LaserJet printers introduced before the LaserJet III. If you wish to hard-code font metric data into an application, HP has provided a TFM Reader utility on disk. The TFM Reader includes a program module to print TFM data to the printer, the display screen, or to an ASCII file. Use the TFM Reader as stand-alone executable code to extract the TFM data, and then use the ASCII file containing the data in the same way that font spacing tables are used.

Note



Font metric information for *scalable* fonts can be hard-coded into an application, but the information is provided in Design Units at one given size, and must still be scaled appropriately.

For example, to support the ITC Bookman Light Italic font, you would use the TFM Reader to obtain TFM data and then format the resulting font metric information to include in your ITC Bookman driver. Once the font metric data is incorporated into your driver, your application can access the data when it begins to format pages for printing and then adjust the data depending on the desired point size.

HP discourages hard-coding font metric data as a method of support, since font metric information is not distributed to software vendors prior to introduction of new font products (TFM files are supplied to end-users with their font product purchase). Since font metrics are different for every point size, for bitmapped fonts it is necessary to incorporate the data for every point size you wish to support with your application.

To the user, this level of font support means that new font products won't operate with application software without font-specific drivers; hard-coding font metric data is recommended only as a short-term solution until a driver is written to dynamically read TFM files. (For more information on hard-coding TFM data, refer to the *TFM Reader Integration* portion of this chapter.)

TFM Reader Integration

In contrast to hard-coding metric data, writing a driver that can read TFM data eliminates the need to write another driver for each new font product you wish to support. Supporting TFM files allows you to write one driver that can handle any of HP's font products.

Integrating a TFM reader routine enables a software developer to easily support non-scalable bitmapped fonts as well as scalable fonts, since HP provides TFM files with all current font products.

Note



Software developers need not interface with Type Director for font metrics (as was done by some developers in the past); Type Director 2.x creates TFM files for all disk-based scalable typefaces used with PCL 4 and PCL 5 devices. Type Director creates one TFM file per scalable typeface; the font metrics in that TFM file are linearly scalable as appropriate for other point sizes. (In Type Director 2.5, TFM file creation is optional.)

Following are two scenarios that describes how a customer uses an application with an integrated TFM reader. The first scenario is for cartridge-based fonts; the second is for scalable disk-based fonts.

CARTRIDGE-BASED FONTS

The user installs the TFM files for the new cartridge using HP's AutoFont Support Installer supplied with the cartridge. During the font installation, the TFM files are loaded into the \AUTOFONT directory. When the user specifies that cartridge for printing, the software accesses the \AUTOFONT directory for the appropriate TFM information.

SCALABLE DISK-BASED FONTS

For disk-based fonts, the user also uses the supplied HP AutoFont Support Installer. As with the cartridge fonts, the TFM files for the typefaces are loaded into the \AUTOFONT directory. The user selects a symbol set using the AutoFont Support Installer, and the installer creates a scalable font that is stored in the \TD\FONTS directory. The application software, knowing where the font files are stored, lists the fonts that are available to the user. The user then chooses fonts from the available selection. To format the document for printing, the application uses the dynamically-read TFM data. When ready to print, the application downloads the scalable fonts, allowing the printer to scale them to the requested size.

To help developers that are integrating TFM reading capability into their software, HP provides a TFM Reader routine on one of the disks supplied with this manual, and an example program that shows how to read a TFM file. Refer to the TFM Reader portion of this chapter for information on the available tools and how to use them.

Adding TFM reading capability to your software requires the incorporation of a TFM reader routine—either a customized TFM reader or one similar to the TFM Reader that is supplied on disk with this manual. As its name implies, the

TFM Reader reads font data from that font's associated TFM file and then formats the data as a data structure that can be read by your application.

The TFM Reader is intended to be an integral part of your software application. To implement TFM Reader Integration, the TFM Reader is called from your software application as font metric information is required. The section below describes how to incorporate the TFM Reader into your application.

Available Tools for TFM Reader Integration

The tools available to implement TFM Reader integration are as follows:

TFM Reader Source and Executable Code

HP has included the TFM Reader source code (READER.C) and executable code on a disk that is supplied with this manual. Integration of READER.EXE is the heart of dynamic TFM support. (The TFM Reader may also be modified or used as an example for code that you write.)

TFM Files

With this manual, HP has supplied the needed TFM files to support the internal LaserJet printer fonts.

Example Implementation

All source code for the TFM Reader and the example implementation are included on one of the disks supplied with this manual. Software developers may use or modify the source code for specific use with their software packages.

The TFM Reader Program

Although most applications use metric information for determining letter spacing, each developer has their own way of coding the data into their drivers. As a sample program, the TFM Reader serves as a model for extracting metrics information from a TFM file. This program is written in the C programming language and provides two functions:

- It reads a TFM file and creates a TFM data structure containing font metric information.
- It can print font metric information from the data structure to an ASCII file, printer, or display screen.

The TFM Reader creates a data structure, which is stored in RAM and contains the metrics information read from all the tags in the TFM file. After creating the data structure, the TFM Reader returns the address of that structure to the application program. The TFM Reader may be modified to access only some tags, reducing the size of the resultant data structure (and increasing performance).

The TFM Reader has a program module that takes the information from the TFM data structure and prints it to an ASCII file. The ASCII file is created only as a means of viewing the contents of the TFM file. The TFM Reader should be incorporated as part of the application and used to transfer the desired data into an application-specific metrics file (unless the developer uses the TFM file directly). The TFM Reader is basically a function within the application program.

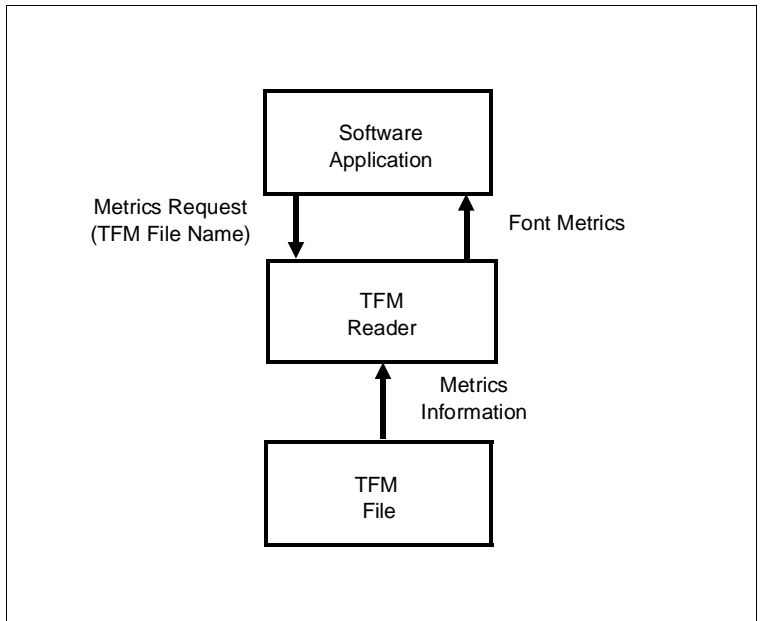


Figure 6-13. The TFM Reader Interface

End-User Considerations

With the integration of the TFM reading capability, HP's AutoFont Support Installer allows end-users to install all TFM files into the \AUTOFONT directory, on the volume of their choice (the HP-recommended default is C:\AUTOFONT). This eliminates the need for the customer to store a TFM file in multiple directories if more than one application is used.

Some applications could use the TFM file as a temporary file which is only used to build an application-specific metric file. Even in this situation, the TFM files should be stored on the hard disk so that other applications have access to the files.

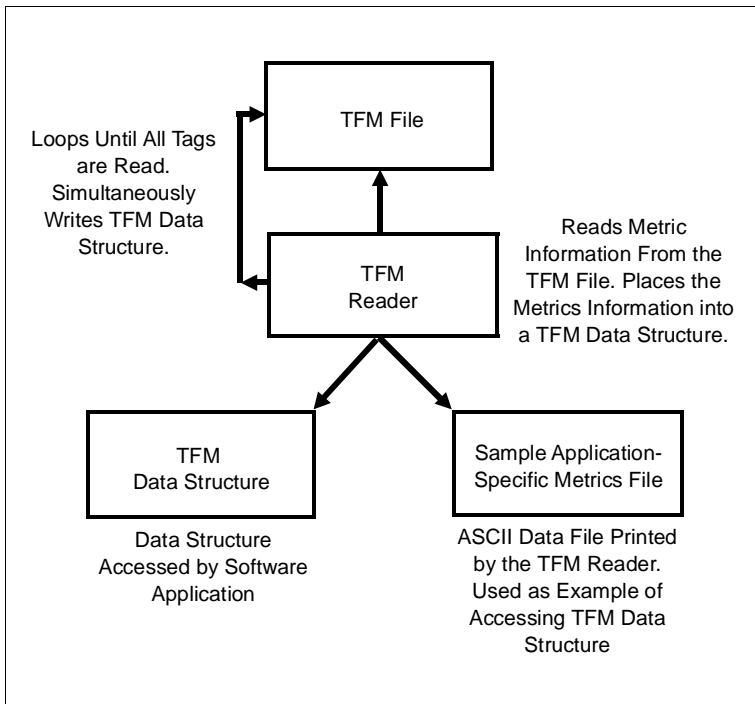


Figure 6-14. The TFM Reader

Using the TFM Reader

The calling program must provide the TFM Reader with the path name and file name of the TFM file to be read. When the TFM Reader has read the file, it passes the address of the TFM data structure it has created to the calling program. The TFM data structure it creates is stored in RAM and contains metric information extracted from the TFM file. The size of the data structure is proportional to the size of the TFM file.

The TFM Reader program should be called from within your application's interface module; it is used as follows (C programming language):

T = TFMREAD(infile);

TFMREAD This function calls the reader routine. The reader routine returns a pointer to the TFM data structure if correctly executed or exits to DOS with an error code if it fails.

T This value is returned from the reader routine (TFMREAD) as a pointer to the TFM structure which was created by TFMREAD. An example declaration is:

Struct TFMTType *T;

infile This parameter passes to the reader routine (TFMREAD) the pathname and filename of the TFM file to be read. An example declaration is:

Char infile[] = "filename"

TFM Reader Data Flow

Following is a description of how the data flows from the application through the TFM Reader and finally to the TFM data structure which is accessed by the application.

- 1) The application makes a request for metric information by supplying the path name of the desired TFM file to the TFM Reader.
- 2) The TFM Reader first retrieves the required header information from the TFM file.
- 3) The TFM Reader then retrieves the requested tags from the TFM file. The tags are ordered in the file by their respective tag number. The TFM Reader reads the tags se-

quentially starting with the first tag. Some tags may contain more data than is stored in the data section of the tag header. In this case, an offset is placed in this location to tell the TFM Reader where to access the tag information. The TFM Reader then interprets the information and retrieves the data.

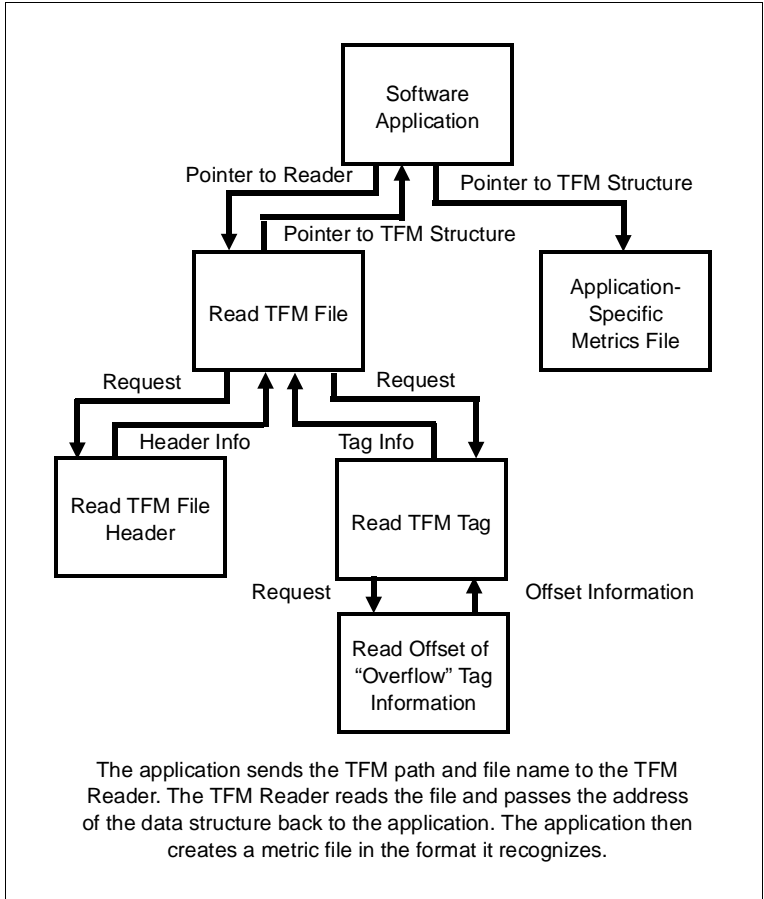


Figure 6-15. TFM Reader Data Flow Diagram

- 4) The TFM data structure is created at the same time the TFM file is being read.

- 5) The TFM Reader returns the address of the TFM structure to the software application.
- 6) The application uses the TFM data structure to create an application-specific metric file in the format needed for the application. (This strategy is sufficient until the TFM file becomes the font metric standard; it is preferred, however, that the TFM file itself take the place of the application-specific structure.)

Modifying the TFM Reader

The TFM Reader code (READER.EXE), located on one of the supplied *PCL 5 Developer's Guide* disks, accesses all tags in the TFM file and creates a TFM data structure which includes information from *all* tags. You may modify the TFM Reader to access only specified tags, with the resulting output being a much smaller TFM data structure.

To modify the TFM Reader to read only certain tags, delete the unused *case* statement. Refer to the TFM Reader source code (READER.C) on one of the HP-supplied disks (included as part of this manual). For example, to prevent the TFM Reader from reading the copyright tag, delete the lines beginning with *case tagCOPYRIGHT:* through *break*. To prevent the TFM Reader from reading the strokeweight tag, delete the lines beginning with *case tagSTROKEWT:* and ending with *break*.

All program lines from the appropriate *case* statement (depending on the tag to be skipped) through the *break* must be deleted. The default *case* is used to compensate when there is no *case* statement for a particular tag. It causes an advance to the next tag in the TFM file without reading the information contained in that tag.

Accessing the TFM Data Structure

A basic map of the TFM data structure created by the TFM Reader program is shown in table 6-1. Within the main data structure is the *typeface* sub-structure, since there may be more than one typeface in a TFM file (although with HP TFM files, this won't be the case). The *typeface* sub-structure contains five sub-structures which are *general*, *typefaceMetrics*, *symbol*, *characterMetrics*, and *kerning*.

Table 6-1. The TFM Data Structure

TFM	Contains general information about the TFM file, such as processor family, version of the file, and number of typefaces contained in the file.		
	typeface	general	Contains general information about a particular typeface, such as typeface name, source, copyright, number of characters in the typeface, and number of pre-defined symbol sets available.
		typefaceMetrics	Contains font metrics about a particular typeface such as point size, stroke weight, cap height, ascent, and descent.
	symbol	symbolMap	Specifies the mapping of symbols in the font file and mapping to HP's MSL and the Unicode list.
		symbolSetDirectory	Sub-structure contains symbol set name, selection string, and mapping index for each pre-defined symbol set.
	characterMetrics	character	Contains metrics information such as horizontal escapement, vertical escapement, left extent, and right extent for each character in the typeface.
	kerning	kernpairs	Contains kern pair information.
		sectorKernChar	Contains sector kerning information.
		trackKern	Contains track kerning information.

Following are some examples that demonstrate accessing information from the TFM data structure.

Each field in the TFM data structure is explained in detail in the “TFM File Structure” section which appears earlier in this chapter. Data types for the TFM Data Structure are as follows:

char	BYTE	/* 8-bit (1 byte) character */
int	WORD	/* 16-bit (2 bytes) integer */
long	LONG	/* 32-bit (4 bytes) integer */
double	RATIONAL	/* 64-bit (8 bytes) floating point */

Once the TFM data structure has been created, font metric data is accessed by giving the complete name of the requested information. The following examples demonstrate how to access the TFM data structure.

Example: Accessing the TFM Data Structure

The following five examples demonstrate how to access data from the data structure created by the TFM Reader:

- 1) This example accesses the version number of the TFM file by using the following pointers:
TFM pointer -> version
- 2) This example accesses the typeface name of the font from the TFM file by using the following pointers:
TFM pointer -> typeface -> general.typeface
- 3) In this example, the recommended line spacing for the font is accessed by using the following pointers:
TFM pointer -> typeface ->
typefaceMetrics.recommendedLineSpacing
- 4) The symbolIndex for a given pre-defined symbol set is accessed by using the following pointers, where *x* is the symbol set index and *y* is the index number of the character index:
TFM pointer -> typeface ->
symbol.symbolSetDirectory [*x*] -> symbolIndex [*y*]

5) This example demonstrates how to access the width of each character for a given pre-defined symbol set by finding the symbol index as shown in the previous example, and then using the following pointer to access *horizontal escapement* (*x* is the character index number):

```
TFM pointer -> typeface ->  
characterMetrics.character[x] -> horizontalEscapement
```

Once TFM data is retrieved from the TFM data structure, the data is frequently used in calculations. For instance, to determine the number of characters in a line of text, the sum of the horizontal escapement values of each character in the line must be subtracted from the desired text width. The following example demonstrates using TFM data to calculate the width of a character.

Example: Using TFM Values in Intellifont Calculations

This example converts the width of a 20-point numeral “2”, which has a horizontal escapement of 4391 Design Units, to PCL Units of Measure. The first example uses 300 PCL Units per inch and the second example uses 600 PCL Units per inch.

The font metric units for scalable fonts are always given in Design Units. Convert Design Units to PCL Units using the following equation:

$$x = \text{round}(b * c * (d/e) * (uom))$$

where:

x = desired metric value in units of measure

b = inches per point

c = point size of font

d = metric value in Design Units

e = number of Design Units in the design EM

uom = PCL Units of Measure

Note



Device dots determine the *resolution* to which the character is rendered. The PCL Units of Measure setting determines the accuracy of the *placement* of the character.

For LaserJet 4 printers, the Unit of Measure command defines how a “dot move” is interpreted. Font metrics should be calculated based on the current PCL Unit of Measure size. The default Unit of Measure is 300 PCL Units per inch. For more information, see the “Units of Movement” discussion in Chapter 4, or refer to the *PCL 5 Printer Language Technical Reference Manual*.

Intellifont Method (20 point at 300 PCL Units per inch)

Assume the following values:

b = 1 inch/72.307 points (Tag 406)

c = 20 points

d = 4391 Design Units (Tag 433)

e = 8782 Design Units (Tag 408)

uom = 300 PCL Units of Measure

Using the above values:

$$x = \text{round}(1/72.307 * 20 * 4391/8782 * 300)$$
$$= \text{round}(41.489)$$
$$= 41 \text{ PCL Units of Measure}$$

Intellifont Method (20 point at 600 PCL Units per inch)

Assume the following values:

b = 1 inch/72.307 points (Tag 406)

c = 20 points

d = 4391 Design Units (Tag 433)

e = 8782 Design Units (Tag 408)

ppi = 600 PCL Units of Measure

Using the above values:

$$\begin{aligned}x &= \text{round}(1/72.307 * 20 * 4391/8782 * 600) \\ &= \text{round}(82.979) \\ &= 83 \text{ PCL Units of Measure}\end{aligned}$$

Supplied TFM Files

Tagged Font Metric Files have been included in the \TFMS directory on one of the supplied *PCL 5 Developer's Guide* disks. TFM files for on-board printer fonts are not supplied to the end user; instead, they are supplied to you. Software developers should either copy the supplied internal-font TFM information to the AUTOFONT directory during their installation procedure or hard code this information in a basic internal font driver for the PCL 5 printers (a "base" driver).

There is one TFM file per scalable typeface. That TFM file has indices for all pre-defined symbol sets. Note that TFM files for scalable fonts are not dependent on symbol set or orientation. TFM files for bitmapped fonts are bound by symbol set *and* orientation.

Some of the TFM files located on the supplied disk are for the LaserJet printer's internal bitmapped fonts, and the remaining files are for the scalable fonts internal to the PCL 5 printers.

File Naming Convention

Using the former TFM file naming convention, there was no way to uniquely identify very similar typefaces or versions of typefaces based solely on the TFM filename. Therefore, Hewlett-Packard is using a new file naming convention. Since there are not enough filename characters available to correctly identify a typeface, the new convention will not attempt to relate font information to the filename in any way. For accessory type with AutoFont installation programs, information about the font should be obtained from the TFM file itself or from the glue file.

Former filename convention	New filename convention
ttrxxssa.tfm	vvv _ _ _ _ .tfm
tt: typeface family designator r: treatment designator xx: point size ss: symbol set a: format designator tfm: filename extension	vvv: HP-assigned vendor ID number (base-36) _ _ _ _ : random number tfm: filename extension

There is no need to change older, existing TFM filenames unless there is a specific problem with TFM filenames overwriting each other. The new convention should be applied to all new TFM files created.

In current TFM releases, Hewlett-Packard is using the new filename convention, using the characters “9NB” for the “vvv” field. The characters “9NB” are followed by a 5-digit random number, and a “.TFM” extension.

Sample TFM Implementation

The file “AG.C” is the source code for an “ASCII Generator” program that accesses all fields in the TFM Data Structure and prints the information in ASCII format either to a file, to the display screen, or to a printer. The source code for AG.C is located on one of the supplied disks. The executable code is part of READER.EXE (the TFM Reader executable code).

As mentioned, the ASCII Generator is a part of the TFM Reader, and is called by the TFM Reader. It can also be accessed from the command line. To create an ASCII file from the TFM data structure, run the supplied READER.EXE file by typing READER at the DOS prompt. The TFM Reader shell will prompt you for the name of the TFM file to be read. Enter the TFM file name. Once the TFM Reader has read the TFM file and created the RAM-resident TFM data structure, the TFM Reader prompts for the location to send the output ASCII file. The selections are as follows:

- S — to the SCREEN
- P — to the PRINTER
- F — to a FILE
- Q — to QUIT

If “Q” is selected, the TFM Reader program is exited and the TFM data structure is no longer accessible.

Software developers may use a program similar to the ASCII Generator to access the TFM data structure and then create an application-specific metrics file. The ASCII Generator contains functions which read the sub-structures within the main TFM Data Structure. For more information, see the *Accessing the TFM Data Structure* discussion earlier in this chapter.

Following is a list of the ASCII Generator’s functions:

printTFMType

This function reads the TFM “type” information from the TFM structure, formats the information, and prints it to an ASCII file. (Refer to the AG.C code on the supplied disks.) The function is called only once for each TFM file. The *printTFMType* function reads the following data from the TFM header and directory:

- Processor Family
- Version Number
- Number of Typefaces

Note



A TFM file may include information about several typefaces. This is not likely, but if there is more than one typeface, the “Number of Typefaces” field indicates a number greater than 1. If this happens, the TFM Reader reads the entire typeface before continuing to the next typeface.

printGeneralInfo

This function reads the “general typeface” information from the TFM data structure, formats it, and writes it to a file in ASCII format. (Refer to the AG.C code on the supplied disks.) This function is called as many times as there are typefaces in the TFM file. The *printGeneralInfo* function reads the following information from the *general* sub-structure of the TFM data structure:

- TFM Type (Tag 400)
- Unique Association ID (Tag 405)
- Typeface Name (Tag 417)
- Typeface Source (Tag 418)
- Copyright (Tag 401)
- Comment (Tag 402)
- Typeface Selection String (Tag 442)

- Number of Characters (refer to Tag 403)
- Number of Symbol Sets (refer to Tag 404)
- PANOSE Numbers (Tag 443)

printTypefaceMetrics

This function reads the typeface metrics information from the TFM structure, formats it, and writes it to a file in ASCII format. (Refer to the AG.C source code on the supplied disks.) This function is called as many times as there are typefaces in the TFM file. The *printTypefaceMetrics* function reads the following information from the *typefaceMetrics* sub-structure of the TFM data structure:

- Point Size in Inches (Tag 406)
- Nominal Point Size (Tag 407)
- Design Units (Tag 408)
- Stroke Weight (Tag 411)
- Appearance Width (Tag 414)
- Serif Style (Tag 415)
- Type Structure (Tag 410)
- Spacing (Tag 412)
- Slant (Tag 413)
- Average Width (Tag 419)
- Maximum Width (Tag 420)
- Inter-word Spacing (Tag 421)
- Recommended Line Spacing (Tag 422)
- Capheight (Tag 423)
- Xheight (Tag 424)
- Ascent (Tag 425)
- Descent (Tag 426)
- Lower Case Ascent (Tag 427)

- Lower Case Descent (Tag 428)
- Underscore Descent (Tag 429)
- Underscore Thickness (Tag 430)
- Uppercase Accent Height (Tag 431)
- Lowercase Accent Height (Tag 432)

printSymbolMSLInfo

This function finds the total number of symbols in the typeface and maps each symbol to HP's Master Symbol List. The function then formats the information and writes it to an output file in ASCII format. This function is called as many times as there are symbols in the typeface. (Refer to the AG.C code on the supplied disks.)

`printSymbolMSLInfo` reads the following information from the *symbol* sub-structure of the TFM data structure:

- Symbol Map (Tag 403)

printSymSetInfo

This function reads the symbol set information from the TFM structure, formats it, and writes the information to an ASCII file. The function is called as many times as there are symbol sets in the typeface. (Refer to the AG.C source code which is located on the supplied disks.) The following information is accessed from the *symbolsetDirectory* sub-structure of the TFM data structure.

- Symbol Set Name (refer to Tag 404)
- Symbol Set Selection String (refer to Tag 404)
- Symbol Set Index Array (refer to Tag 404)
- Array Length (refer to Tag 404)

printCharacterMetrics

This function reads the character metrics information from the TFM data structure, formats the information, and writes it to an output file in ASCII format. This function is called as many times as there are characters in the type-

face. (Refer to the AG.C code on the supplied disks.) The *printCharacterMetrics* function reads the following information from the *characterMetrics* sub-structure in the TFM data structure:

- Horizontal Escapement (Tag 433)
- Vertical Escapement (Tag 434)
- Left Extent (Tag 435)
- Right Extent (Tag 436)
- Character Ascent (Tag 437)
- Character Descent (Tag 438)

printKerningInfo

This function reads the kerning information from the TFM data structure and writes it to the ASCII output file. This function reads three types of kerning information: sector, track, and kern pairs. The function is called as many times as there are typefaces in the TFM file. (Refer to the AG.C source code which is located on the supplied disks.)

printKerningInfo reads the following information:

From the *kerning* sub-structure of the TFM data structure:

- Number of Kern Pairs (refer to Tag 439)
- Number of Sector Kern Characters (refer to Tag 440)
- Number of Sectors per Character (refer to Tag 440)
- Number of Tracks (refer to Tag 441)

From the *kernPairs* sub-structure (refer to Tag 439):

- First Character Index
- Second Character Index
- Kern Value

From the *sectorKernChar* sub-structure (refer to Tag 440):

- Character Index
- Left Side Sector Values
- Right Side Sector Values

From the *trackKern* sub-structure (refer to Tag 441):

- Track Value
- Maximum Point Size
- Minimum Point Size
- Maximum Kern Value
- Minimum Kern Value

Note



The ASCII Generator only prints tags that have been read by `READER.EXE`. It is not necessary to modify the ASCII Generator if the TFM Reader is modified to access only certain tags.

Selecting Fonts Using TFM Information

Adding the TFM Reader, or a modified version of the reader, allows you to use font metric information for more than just character spacing. A very good use of TFM data is for obtaining the proper PCL selection command for a particular font.

To select a printer font, seven parameters are used:

- Symbol Set
- Spacing
- Pitch
- Height (Point Size)
- Style
- Stroke Weight
- Typeface

The values for these seven parameters may be read from a TFM file as described in the following paragraphs. The values obtained from the TFM files may then be used in the font select command to select that particular font.

Note

When selecting fonts for the LaserJet series II printer, the orientation of the font must also be specified, because the font must be available in the orientation you wish to print. (The LaserJet 2000, IID, IIP and all PCL 5 LaserJet printers automatically rotate fonts to match the current orientation.)

Symbol Set

The value needed to select a given symbol set is provided by the symbol set selection string, located in the font's TFM file. The byte offset to the selection string is located in the 5th through 8th bytes of Tag 404 (symbol set directory). The symbol set selection string is used in the symbol set command, $\text{\textcircled{E}}\text{c(ID)}$.

Spacing

The spacing value for font selection is obtained from Tag 412 (spacing). A value of zero indicates that the font is proportional. Any other value indicates that the font is fixed-spaced. The application must translate any non-zero Tag value to 1 for proportional spacing, $\text{\textcircled{E}}\text{c(s1P)}$, and use a value of 0 for fixed spacing, $\text{\textcircled{E}}\text{c(s0P)}$.

Pitch

The value needed to select the correct font pitch is also provided using Tag 412 (spacing). If the value read is zero, the font is proportional. If the Tag value is any other number, the value given is the pitch of the font in Design Units.

For example, the spacing value of 600 Design Units is given in the TFM file for a Courier 12-point, 10 cpi bitmapped font. Given that the font was designed using 72 points per inch, and the number of Design Units in the design EM is 1000, the following equation can be used:

$$(1 \text{ in./72 points}) * 12 \text{ points} * (600 \text{ du}/1000 \text{ du}) = 0.1 \text{ inches}$$

The pitch was calculated to be 0.1 inch/character, and the inverse is 10 characters/inch. Ten is the value that would be sent to the LaserJet printer in the command $\text{\textcircled{E}}\text{c(s\#H)}$. To convert the pitch to dots, the following equation can be used:

$$(0.1 \text{ inch}) * (300 \text{ dots/inch}) = 30 \text{ dots}$$

Height (Point Size)

The command value needed to select the correct point size is provided by Tag 407 (nominal point size). If the font is a bitmapped font, the value read is the value to be sent to the printer in the height command $\text{e}_c(\text{s}\#V$. For scalable fonts, the point size is input by the user and then sent to the printer using the same command. In this case, the printer uses the height value as an operator to indicate the desired point size for scaling the font.

Style

The style value is obtained using the Tag values read from three different tags: Tag 413 (Slant), Tag 414 (Appearance width), and Tag 410 (Type structure). The PCL values for these three tags are combined into a style word as follows:

Bits 0 - 1: Posture (slant)

Bits 2 - 4: Width

Bits 5 - 9: Type Structure

Bits 10 - 15: Reserved

Note



All LaserJet printers look for an exact match for style, although only the 0 and 1 values were defined for the LaserJet series II, IID, IIP, and earlier printers.

Posture

To obtain the correct posture value, the value is read from Tag 413 (slant). If the value read is other than zero the font is italic and the value indicates the degree of slant. Valid values for the PCL Posture (bits 0 and 1 of the style byte) are as follows:

PCL Value	Posture
0	Upright
1	Italic
2	Alternate Italic
3	Reserved

Width

The width value is read from Tag 414 (appearance width). The valid range for TFM values and corresponding PCL values is shown in the following table.

Width	Tag Value	PCL Value*
Ultra-Compressed	0-20	4
Extra-Compressed	21-47	3
Extra-Condensed	48-74	2
Condensed	75-101	1
Semi-Condensed	102-128	0
Normal	129-155	0
Semi-Expanded	156-182	0
Expanded	183-209	6
Extra-Expanded	210-236	7
Ultra-Expanded	237-255	7

* Multiply PCL Value by 4 for Style Word Partial Sum

Structure

To obtain the correct value for structure, read the value of Tag 410. The TFM values and their corresponding PCL values are indicated in the following table.

Structure	Tag Value	PCL Value*
Solid	0-7	0
Outline	8-15	1
Inline	16-23	2
Contour	24-31	3
Solid with Open Shadow	32-39	4
Open with Solid Shadow	40-47	5
Inline with Shadow	48-55	6
Contour with Shadow	56-63	7
Pattern 1**	64-71	8
Pattern 2**	72-79	9
Pattern 3**	80-87	10
Pattern 4**	88-95	11
Pattern 1 with Shadow**	96-103	12
Pattern 2 with Shadow**	104-111	13
Pattern 3 with Shadow**	112-119	14
Pattern 4 with Shadow**	120-127	15
Inverse**	128-135	16
Inverse w/ Open Border**	136-143	17
Reserved**	144-255	18-31

* Multiply PCL Value by 32 for Style Word Partial Sum

**These values are not supported on the LaserJet series II, IID, and IIP printers.

To derive a parameter value for the style command, the Tag values for posture, width, and structure are converted to

PCL values using the tables above. Then the PCL values are added to get the style command value: $E_c(s\#S)$.

The example below shows how the desired style command parameter value is obtained from a sample font:

Assume that the Tag values are as follows: Posture value (Tag 413) = 1; appearance width (Tag 414) = 75; structure value (Tag 410) = 32.

- A posture value of 1 (italic) equals a PCL value of 1.
- An appearance width of 75 (condensed) equates to a PCL value of 4 ($1 \times 4 = 4$).
- A structure value of 8 (outline) equates to a PCL value of 32 ($1 \times 32 = 32$).

Totaling the PCL values gives a parameter value of 37 = ($1 + 4 + 32 = 37$). Therefore, the correct style command for selecting this font is $E_c(s37S)$.

Stroke Weight

The correct stroke weight parameter value is provided by Tag 411 (stroke weight). The Tag values and their corresponding PCL values are listed in the following table:

Stroke Weight	Tag Value	PCL Value
Ultra-Thin	0-17	-7
Extra-Thin	18-34	-6
Thin	35-51	-5
Extra-Light	52-68	-4
Light	69-85	-3
Demi-Light	86-102	-2
Semi-Light	103-119	-1
Medium	120-136	0
Semi-Bold	137-153	1
Demi-Bold	154-170	2
Bold	171-187	3

Stroke Weight	Tag Value	PCL Value
Extra-Bold	188-204	4
Black	205-221	5
Extra-Black	222-238	6
Ultra-Black	239-255	7

The PCL Value is the value used in the stroke weight command. For example, if the value of Tag 411 is 137 (semi-bold), the corresponding PCL value is 1 and the resulting stroke weight command is $\text{e}_c(s1B)$.

Typeface

Tag 442 provides the font's typeface value. The Tag value is the value that is used in the typeface command, $\text{e}_c(s\#T)$, to select that font.

Note



Some typeface values are two-byte values. The LaserJet IID, IIP, and PCL 5 printers all support two-byte typeface selection parameters. The LaserJet series II and previous printers only recognize a one-byte value, which is the lower byte of the two-byte value (modulo 256).

Locating the TFM Files

Hewlett-Packard recommends that all TFM files be placed in the \AUTOFONT directory of the root directory. HP's AutoFont Support Installer is used as the mechanism for copying TFM files onto the hard disk. Using the utility, the user is able to select the volume, but not the directory, where the TFM files will be located.

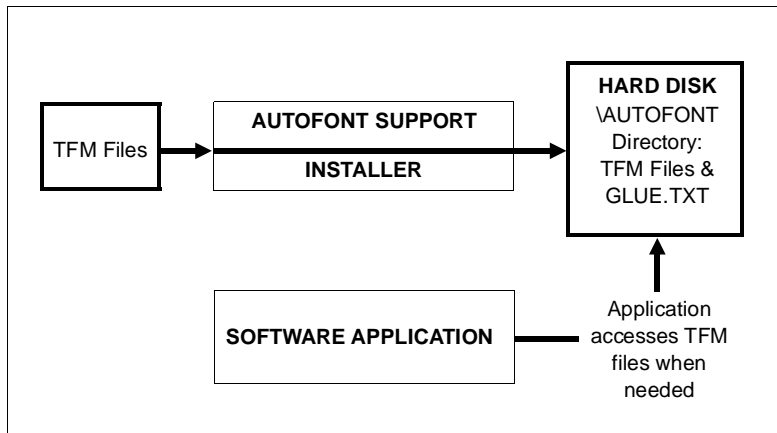


Figure 6-16. Locating the TFM Files

The Glue File

When the AutoFont Support Installer loads the TFM files onto the hard disk, an ASCII file named `GLUE.TXT` is created and is placed in the same directory as the TFM files (`\AUTOFONT`). This file serves as a link between the font file and its associated TFM file. The `GLUE.TXT` file is a tool that software developers can use to get font information that is not obtainable from the font metric files, namely:

- The volume/directory location of the TFM files
- The location of the font file associated with the TFM files
- Whether the TFM file represents information about a downloadable soft font, internal printer font, or a cartridge font.
- The symbol set associated with the font

Note



Detailed information about the `GLUE.TXT` file is described earlier in this chapter.

Creating TFM Files from Intellifont Files or PCL Bitmaps

This section is for font vendors wishing to create TFM files for their fonts. It discusses the following items:

- Tools available to create TFM files
- How to use the TFM Writer
- How data flows through the TFM Writer
- The errors that can be returned by the TFM Writer
- How to modify the TFM Writer
- How to access the TFM Writer data structures

The TFM Writer was created by Hewlett-Packard to create TFM files from Agfa FAIS files, Agfa Library files, or PCL bitmaps with either short (26 bytes) or long (64 bytes) PCL font headers. Most recently, the ability to create a TFM file from a TrueType typeface file has been added.

Note



Chapter 8 discusses TFM Writer support for TrueType files.

This writer is composed of four programs: one for Agfa FAIS files, one for Agfa Library files, one for PCL bitmapped files, and one for TrueType files. Each program is compiled from the same source code, but with the use of Microsoft C #IFDEF statements, different parts of the source code can be included or excluded. To protect the compatibility of all TFM files, font vendors may not modify the TFM Writer to produce new tags. However, portions of this code may be modified to create TFM files from other source metric or font files. Refer to the “Modifying the TFM Writer” discussion for information on how to modify the TFM Writer to read different input metric files.

Note



When referring to the TFM Writer throughout this section, the reference is actually to the four different programs associated with the TFM Writer.

Available Tools

The tools available to implement TFM Writer are as follows:

- TFM Writer Source and Executable Code

TFM Writer source and executable code has been included in the WRITER subdirectory on one of the disks supplied with this manual.

- TFM Specifications

The Tagged Font Metric Specifications found earlier in this chapter provide definitions of each TFM tag.

The TFM Writer

Using the TFM Writer

The TFM Writer, provided on disk, is invoked by spawning from an interface program, INTRFACE.EXE. This interface program prompts the user for all needed information to create a TFM file from a PCL bitmapped font file, a FAIS font file, a Library font file, or a TrueType file. The interface program provides the TFM Writer with the following information through an ASCII file named PARAMET.ERS.

When you run the TFM Writer program you will be asked the following questions. Each question corresponds to a line entry in the file PARAMET.ERS:

1. The type of file being used.

The TFM Writer knows of four different files it can read: FAIS, Library, PCL bitmaps, or TrueType files. In this location, place an “f” for FAIS files, “l” for Library files, “b” for PCL bitmap files, or “n” for bitmap files with subfile type = 1 (Tag 400).

2. The file name of the source file which the TFM Writer will read to obtain metric information.
3. The file name of a second file (only if needed).

For example, FAIS files contain two files associated with each typeface, the attribute file and the descriptor file. The attribute file (an odd numbered file such as F0001.FF) would be listed in step #2 of this list and the descriptor file (an even numbered file such as F0002.FF)

would be listed in step #3 of this list. If a second file is not used, place a “-1” in this location.

4. The path of the files to be read.
5. The path of the *.SYM files.

The *.SYM files are ASCII files which contain the different symbol sets to be placed in the TFM file. Each contains general information about the symbol set (that is, the PCL selection string, symbol set name, etc.) and a list containing the decimal value and corresponding Agfa CG character number of a character.

6. The path and file name of the kern pair list.

The HP version of TFM Writer uses the ASCII file TD.KRN which contains a list of over 1000 pairs of different character combinations. The format of this file contains the 1st Agfa CG character number and the 2nd Agfa CG character number separated by a space. Each entry is on a separate line.

7. The path and file name of the CG to MSL conversion list.

The HP version of the TFM Writer reads FAIS and Library files that contain Agfa CG numbers for the character numbers. The file, CG-MSL.EXH, contains a list of Agfa CG numbers and equivalent MSL numbers. This list is then used to convert the CG character numbers in the metric files to their equivalent MSL numbers. The format of this file contains the CG character number and the corresponding MSL character number, separated by a space. Each entry is on a separate line. Font vendors should replace this list with one that converts their character numbers to the equivalent Master Symbol List numbers.

8. The path and file name of the limited sensitivity and universal character metrics file.

HP uses a file (PLUGIN.TYQ) which contains characters common to all typefaces or groups of typefaces. This file is used to append those common characters to the list specific for the typeface. If this file is not needed, place a “-1” in this location.

9. The path and file name of the temporary ASCII TFM file.

A temporary ASCII file is used to initially generate the various tags. When all tags have been developed, the WT-TFM.EXM module is called which reads this temporary file and converts it to binary. The default path and file name used is TEMP\RESULTS.TAG

10. The destination path of the new TFM file.

After the interface program has spawned to the correct program of the TFM Writer (FAIS.EXM—executable program for reading FAIS files, LIBRARY.EXM—executable program for reading Library files, and BITMAP.EXM—executable program for reading Bitmap files), the TFM Writer reads the parameters listed above via the PARAMETERS file. Using the parameters, TFM Writer finds as much information about the typeface as possible. With this information, tags are developed and placed in the temporary ASCII file. After all tags applicable for the typeface have been produced, a new TFM file name is placed at the end of the file. Control is then given back to the interface program. The interface program spawns to WT-TFM.EXM. This program first reads the PARAMETERS file to find the temporary ASCII file name and destination path. It then reads the last line of the temporary ASCII file for the new TFM file name. Using this new path and file name, WT-TFM.EXM converts the temporary file from ASCII to binary, placing the information in the order specified by the Tagged Font Metric specifications (outlined earlier in this chapter).

Data Flow

The following process describes how metric information is placed in the TFM file with the TFM Writer and WT-TFM.EXM. Figure 6-17 describes how data flows through the TFM Writer from a source metric file containing information about the typeface to the binary TFM file which is used by the software developer.

Internal Process for Creating a TFM File:

1. The File Reader is entered.
2. The File Reader places the CG-MSL conversion list into a global data structure. This structure is used frequently throughout the TFM generation process to convert a character number to its equivalent MSL number. Font vendors should supply a conversion list similar in format to the CG-MSL.EXH file to convert character numbers to Master Symbol List numbers.
- 3a. The File Reader retrieves the typeface metrics from the file(s) requested through the PARAMET.ERS file. If font vendors want to read metric files in a format other than FAIS, Library, or PCL, the File Reader should be modified to include new code that reads the metric file.
- 3b. If the File Reader is retrieving metrics from a bitmap font, the symbol set file (.SYM file) using the same symbol set as the bitmap font is found. For example, if the bitmapped symbol set is Roman-8, the .SYM file for Roman-8 is found. General information about the symbol set is taken from the .SYM file and used later when developing the Symbol Set Directory tag.
4. The File Reader adds universal and limited sensitivity characters to the already accumulated list of characters. These characters are only added if they are required for a given typeface product, like an FAIS or Library file.
5. Once all possible information has been obtained, control is given to the Tag Developer which develops the tags and places them in the temporary ASCII file. For developing the Symbol Set Directory tag for scalable typefaces, all

.SYM files in the directory specified in the PARAMET.ERS file are used. Information is taken from these files and placed in the Symbol Set Directory tag. For developing the Pair Kern tag, the kern pair list (TD.KRN) is used. It contains a list of all the pairs to be placed in the Pair Kern tag.

6. After all possible tags have been developed, the Tag Developer returns control to the interface program. The interface program then spawns to the ASCII to Binary Converter, WT-TFM.EXM.
7. WT-TFM.EXM converts the tags in the temporary ASCII file to binary and places them in the new binary TFM file. The name of this file is read from the last line of the temporary ASCII file.
8. WT-TFM.EXM returns control to the interface program and the TFM generation process is done.

Note



When control is returned to the interface program, a status number is passed back. If the status number is zero, no errors occurred and the process is successful. If the status number is not zero, an error occurred and a TFM file was not successfully created. The textual representation of the error number is found in TFM.ERR.

If font vendors wish to modify the TFM Writer to read other source metric files, the File Reader portion of the TFM Writer is the only portion that should be modified. All other portions of the TFM Writer should NOT be modified, to ensure consistency for all TFM files. Refer to the following “Modifying the TFM Writer” section for details on how to modify the writer for different metric files or on how to get new tags placed into the TFM file.

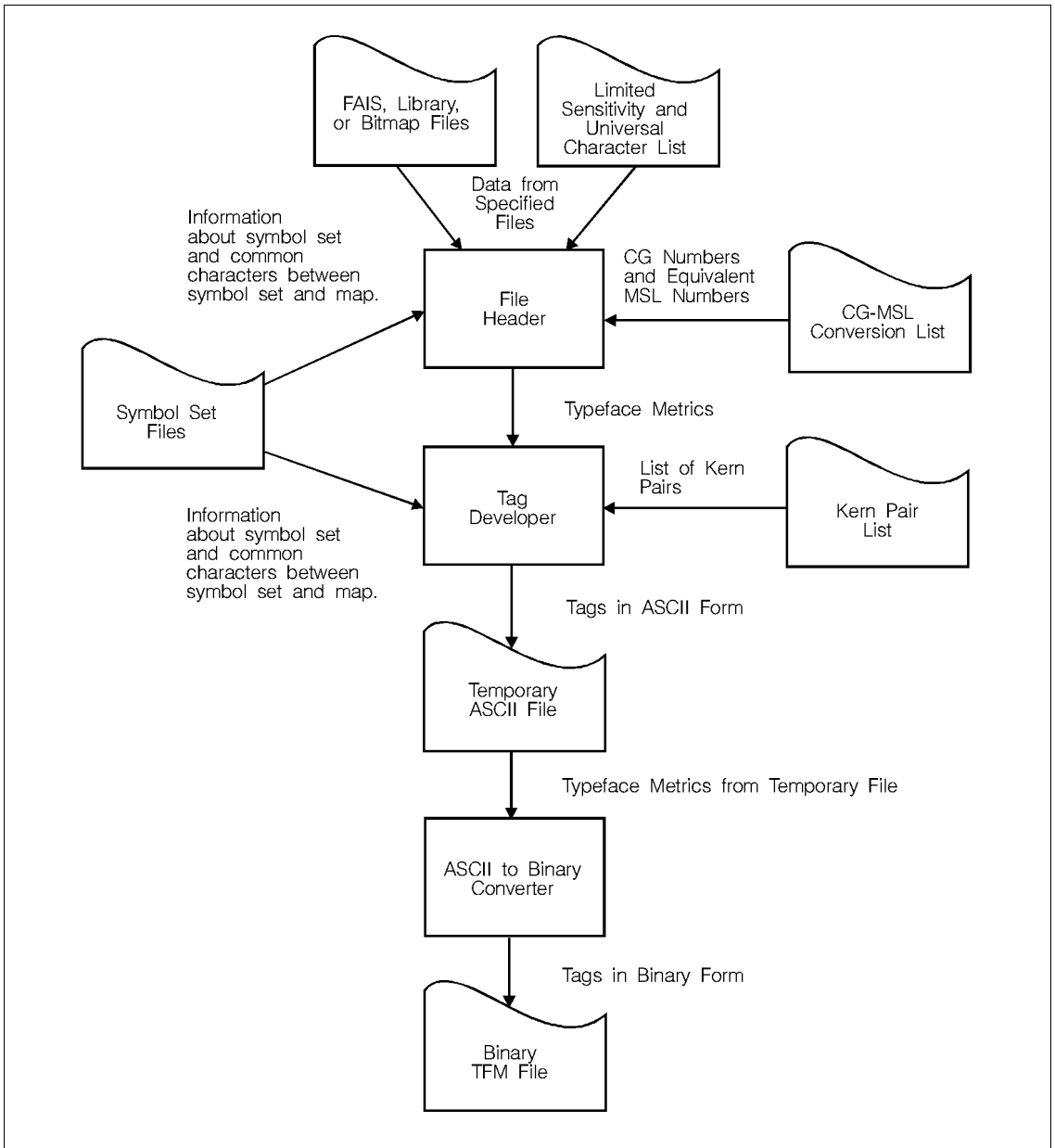


Figure 6-17. TFM Writer General Data Flow Diagram

PANOSE Numbers

PANOSE numbers (Tag 443) may be built into an Intellifont TFM file with the Library file option of the TFM Writer. The following procedure should be followed:

1. Create an ASCII file called “PANOSE.IF” and place it in the same directory as LIBRARY.EXM. Each line in the PANOSE.IF file contains a typeface number followed by 10 PANOSE numbers, separated by spaces. The file format is shown below.
2. Create a TFM file using the Library option. The TFM Writer will check for the presence of the PANOSE.IF file. If it finds the file, it will check for the typeface number and associated PANOSE information. If the information is found, Tag 443 will be built into the TFM file.

PANOSE.IF										
91118	2	11	8	3	2	2	4	3	2	4
90270	2	4	6	4	4	5	5	2	2	4
90249	3	3	5	2	4	4	6	7	6	5

Modifying the TFM Writer

The TFM Writer reads from an Agfa FAIS file, Agfa Library file, PCL bitmap file, or TrueType file. It creates all level 1 tags, plus level 2 and level 3 tags that are applicable for a given typeface.

Modifying the File Reader Portion of the TFM Writer

Modifying the file reader portion of the TFM Writer to read different source metric formats is relatively easy. First, generate code to read the new input file format. Second, place the appropriate data into the appropriate set of data structures provided in the “Accessing the TFM Writer Data Structures” section that follows this. The actual structures are located in header files, TFMFAIS.H, TFMLIBR.H, and TFMBITM.H, which are on the supplied disks. For example, if this new file reader reads scalable files, place the data into either the FAIS or Library data structures. If the file reader reads non-scalable files, place the data into the bitmapped data structures.

When choosing the most suitable data structures, note that some structures are used interactively with different files. For example, the copyright data structure is used by the file reader when reading FAIS, Library, or bitmapped files. This feature is not common for all of the data structures. Some structures are specifically for only one type of file. The tables in the “Accessing the TFM Writer Data Structures” section list the type of file(s) that each structure can be used with.

Whenever a modified metric file reader is used, fill the existing data structures with the metric data from the new source file. When the Tag Developer, module DEV1, is called, a variable specifying the type of data structure is sent with the metric information. Font vendors using a modified metric file reader should specify in this variable the type of existing data structure used, not the type of file read. For example, if the bitmap data structures are filled in, the variable sent to the DEV1 module should be a “b”. If Library data structures are used, the variable sent to the DEV1 module should be an “l”. If FAIS data structures are used, the variable sent should be an “f”.

Note that the file type value specified in PARAMETERS is used for determining what type of file the File Reader is reading. This could be “f” for FAIS files, “b” for Bitmap files, “l” for Library files, or some other letter for another font vendor’s metric file. The value sent to the DEV1 module is used to determine what type of data structures the Tag Developer is using. This could be “f” for FAIS data structures, “b” for Bitmap data structures, or “l” for Library data structures. Metric information from another font vendor’s metric file should be placed in one of these data structures, and the same variables should be used.

Adding New Tags to the TFM File

When software developers read TFM files from different font vendors, they must be guaranteed all tag numbers are consistent. They must also be guaranteed that certain tags are always present. For example, character metrics should

always be present in a TFM file. Since every software developer requires different information about fonts, the font vendor should provide at least all level 1 tags. (These tags are defined earlier in this chapter.) To add tags or information to a TFM file, the steps listed below should be followed:

Note



Hewlett-Packard is responsible for adding any new tags to the Tag Developer portion of the TFM Writer and updating the TFM specifications outline in this chapter.

To add new tags to the TFM File:

1. Request a new tag. To do this, contact your HP LaserJet printer support liaison.
2. Hewlett-Packard evaluates the request.
3. If the requested information is not present in the existing TFM file, Hewlett-Packard relays the request to other software developers and font vendors.
4. If a favorable reply is received from other software developers and font vendors, the tag is placed into the TFM specifications and included in the Tag Developer portion of the TFM Writer.
5. This new TFM Writer and specification is distributed. New TFM files can be produced with the new tag.

Accessing the TFM Writer Data Structures

The TFM Writer uses several different data structures. They are composed of several main global variables used for general typeface descriptions, data structures used for different types of kerning data, and a link list for character metrics. Some, such as the link list, are common for all types of files whether bitmapped or scalable, while others are dependent on a specific type of file being read. These structures reside in memory as pointers. If a structure is needed, the structure must first be assigned memory for a specific size, usually the size of the structure. After it is allocated memory, it can be used.

Basic maps are provided below for the common data structures used to place information into the tags. The actual data structures used are located in `TFMFAIS.H`, `TFMLIBR.H`, and `TFMBITM.H`. Examples on how to access different structures are shown on the following pages.

Note



In the data structures listed below, the (*) markers indicate which tag is produced by that field. Those marked by a (**) are used for other calculations. Those fields not marked are not used.

The structures described below are used for reading the font metric information from a bitmapped font file.

Bitmapped Font File Data Structures		
*bitFontHeader	fhSize	Font descriptor size (**)
	fhB0	Reserved
	fhFontType	Font type
	fhW0	Reserved
	fhBaseline	Baseline distance
	fhCellWidth	Cell width
	fhCellHeight	Cell height
	fhOrientation	Orientation (**)
	fhSpacing	Spacing (*412)
	fhSymbolSet	Symbol set (*404)
	fhPitch	Pitch (default HMI) (*412,419,421)
	fhHeight	Height (*407)
	fhXHeight	xHeight (*424)
	fhWidthType	Width type (*414)
	fhStyle	Style (*413)
	fhStrokeWeight	Stroke weight (*411)
	fhTypeFace	Typeface (*422)
fhSlant	Slant	
fhSerifStyle	Serif style (*415)	

Bitmapped Font File Data Structures

*bitFontHeader	fhQuality	Quality
	fhW1	Reserved
	fhUnderlineDistance	Underline distance (*429)
	fhUnderlineHeight	Underline height (*429,430)
	fhTextHeight	Text height
	fhTextWidth	Text width
	fhFirstCode	First code
	fhLastCode	Last code
	fhPitchExt	Pitch extended (*412,419,421)
	fhHeightExt	Height extended (*407)
	fhW2	Reserved
	fhFaceSource	Typeface source
	fhHiFaceNumber	Hi-byte typeface number
	fhLoFaceNumber	Lo-byte typeface number
	fhFontName[17]	Font name (*417,418)
*charHeader	ASCII_num	Index number (**)
	chFormat	Format
	chContinuation	Continuation
	chSize	Descriptor size
	chClass	Class
	chOrientation	Orientation
	chB0	Reserved
	chLeftOffset	Left offset (**)
	chTopOffset	Top offset (**)
	chCharWidth	Character width (**)
	chCharHeight	Character height (**)
	chDeltaX	Delta x (**)

The structures described below are used for reading the font metric information from either FAIS or Library files.

Common Data Structures for FAIS and Library Files		
fontHeader_copyright[62]		Copyright (*401) (used for bitmaps also)
fontHeader_fontDescription[100]		Font description (*402)
fontHeader_timeStamp[6]		Time stamp (*405)
fontHeader_NCHAR		Number of characters (**)
attrHeader_isFixedPitch		Type of pitch (*412)
attrHeader_scaleFactor		Scale factor (*408)
attrHeader_ascender		Ascent (*425)
attrHeader_descender		Descent (*426)
attrHeader_capHeight		Cap height (*423)
attrHeader_xHeight		Xheight (*424)
attrHeader_lcAccentHeight		Lowercase accent ht. (*432)
attrHeader_ucAccentHeight		Uppercase accent ht. (*431)
attrHeader_uscoreDepth		Underscore depth (*429)
attrHeader_uscoreThickness		Underscore thickness (*430,429)
attrHeader_spaceBand		Space width (*412,419,421)
displayHeader_NCHAR		Number of characters (*440)
displayHeader_italicAngle		Italic angle (*413)
*typefaceHeader	NFACES	Number of typefaces
	typeFaceName[50]	Typeface name (*402,417,418)
	familyName[20]	Family name
	weight[20]	Weight of typeface (*411)
*identifier	identifier	Typeface number (*442)

Common Data Structures for FAIS and Library Files

*descriptorSet	stemStyle	Style of the stem (*415)
	stemMod	Mode of the stem (*411)
	stemWeight	Weight of the stem (*415)
	slantStyle	Description of slant style
	horizStyle	Description of horizontal style (*414,415)
	vertXHeight	Description of vertical x Height
	videoStyle	Normal/Reverse characters
	copyUsage	Type of characters
hpfat	NFATS	Number of font alias tables (**)
	aliasTableName[20]	Font alias table name
	NXREF	Number of cross references
	typefaceNumber	Typeface number
	PCLalfa[2]	Typeface abbreviation (**)
	treatment	Typeface treatment
	fixPitch	Fixed/Proportional
	PCLwidth	PCL width (*414)
	structure	Structure of the typeface
	face_width	Appearance Width
	posture	Upright/Italic
	stem_wt	Stroke weight (*411)
	family	TypeFamily code
	serif_style	Type of serif (*415)
	serif_flag	Serif/Non-Serif
	compress	Compressed/Non-Compressed
	font_name[17]	Font name (*417,418)
	FAIS_file_num	Typeface number

The structure described below is used for reading the text kern information from FAIS files.

FAIS File Text Kern Data Structures			
*textKern	kernSign	Sector Kerning adjust to character escapement (*440)	
	kernUnit	Type of units (*440)	
	NSECT	Number of sectors (*440)	
	*character (for each character)	left[8]	Left side of character (*440)
		right[8]	Right side of character (*440)

The structure described below is used for reading the text kern information from a Library file.

Library File Text Kern Data Structure		
*libTextKern	kernSign	Sector Kerning adjust to character escapement (*440)
	kernUnit	Type of Units (*440)
	NSECT	Number of sectors (*440)
	*character (for each character)	sector[4]

The structure described below is used for reading the track kern information from a FAIS file.

FAIS File Track Kern Data Structure			
*trackKern	NTRACK		Number of tracks (*441)
	track[10] (for each track)	kernDegree	Degree of tightness (*441)
		minKernPtSize	Minimum point size (10ths) (*441)
		maxKernPtSize	Maximum point size (10ths) (*441)
		minKernAmt	Minimum adjustment (*441)
		maxKernAmt	Maximum adjustment (*441)

The structure described below is used for reading the kern pair information from a FAIS or Library file.

FAIS and Library Kern Pair Data Structure		
*pairKern	char1	character 1 (*439)
	char2	character 2 (*439)
	value	kern amount in Design Units (*439)

The structure described below is used for accumulating character metric information from a FAIS, Library, or Bit-mapped font file. Each structure is a link for one character with the list being all the characters.

Character Metrics Data Structure		
*character_metrics	cgNum	Character number (*403)
	cgKern	1=has kerning, 0=no kerning (*439,440)
	cgKernIndex	Index in kerning array (*439,440)
	horizontalEsc	Horizontal escapement (*419,430)
	verticalEsc	Vertical escapement (*434)
	rightExtent	Right extent (*420,436)
	leftExtent	Left extent (*420,436)
	ascent	Ascent (*423,425,427,437)
	descent	Descent (*426,428,438)
	*next	Pointer to next link (**)

The structure described below is used for converting CG numbers to their equivalent MSL numbers.

CG to MSL Conversion Data Structure		
*cgmsl	CG	CG number
	MSL	MSL number

Once one of these TFM data structures is used, metric information is accessed by giving the complete name of the requested information. The Tag Developer of the TFM Writer

automatically does this. Reference the TFM data structures (header files) earlier in this chapter when using the following examples. These examples demonstrate how to access the structures to create tags.

Example: Accessing the Typeface Name

This example accesses the typeface name retrieved from a FAIS file by using the following pointers:

```
typefaceHeader —> typeFaceName
```

or if a font alias table is present:

```
hpfat.font_name
```

Example: Accessing Horizontal Escapement Data

Access a character's horizontal escapement retrieved from any of the files by using the following steps:

1. Assign a temporary pointer (temp) to pointer character_metrics (character_metrics always points to the first link in the list).

```
temp = character_metrics
```

2. Using the temp —> cgNum variable, search through the character_metric's linked list for the link containing the character in question.

```
for ( ; temp —> cgNum < character; temp=temp —> next);
```

Note



By assigning temp=temp —> next, you will be able to parse through the linked list, one line at a time.

3. When the character is found, the variable for finding the horizontal escapement is temp → horizontalEsc.

horizontal escapement = temp → horizontalEsc

Example: CG to MSL Conversion

Access the 100th CG number and its MSL equivalent in the CG-MSL conversion list by using the following pointer:

CG number = cgmsl[100].CG

Equivalent MSL number = cgmsl[100].MSL

Example: Accessing Copyright Information

Access the copyright statement retrieved from either a FAIS or Library file by using the following pointer:

fontHeader_copyright

Compiling the TFM Writer

The following source code (.C) files and header (.H) files create the TFM Writer executables on the supplied disks:

TFMWRITE.C	File format reader source code
DEVELOP.C	Tag Developer source code
WT-TFM.C	ASCII to binary source code
IOBYTES.C	Input and output byte functions
TFMBITM.H	Bitmap file format structures
TFMDEF.H	Definitions
TFMERROR.H	Error conditions
TFMFAIS.H	FAIS file format structures (FAIS and Library)
TFMLIBR.H	Library file format structures (Library only)
TFMTAG.H	Tag structures

When referring to the TFM Writer, we refer to each program that reads a different file format.

FAIS.EXM	Executable TFM Writer for reading FAIS files
LIBRARY.EXM	Executable TFM Writer for reading Library files
BITMAP.EXM	Executable TFM Writer for reading Bitmap files

The .EXM file is a .EXE file renamed with the new extension. These executable files are modules spawned by the interface program, INTRFACE.EXM.

The source code, TFMWRITE.C, DEVELOP.C, and .H files, are used to create the above .EXM files. To create each executable .EXM file, a #define variable is changed in TFMDEF.H to represent a different format.

To create the executable files with the above files, follow the steps below:

1. Get into the TFMDEF.H file and go to line 14. Notice the #define statement (two capital letters). Change these letters to the correct type described in lines 8-10 (FS = FAIS, LB = Library, BM = Bitmap). If the subfile type = 1 (Tag 400) for a bitmapped font, also define *sym-set_not_used* by removing the comment markers. Then save the file with the change, and return to the DOS prompt.

2. Compiling the source code is relatively simple. The compiler used for the TFM Writer is Microsoft C, version 7.0. Each executable module is compiled in Large Model format. To compile code, type:

```
cl /AL TFMWRITE.C DEVELOP.C
```

Typing the line above compiles and links the source code, creating an executable file called TFMWRITE.EXE.

3. Once the executable file is generated, it should be renamed to an .EXM file with the first eight characters being the name of the file format type to be read. For example, if the source is compiled for FAIS files (#define FS in TFMDEF.H), the executable file is named FAIS.EXM. It is good practice to pack the executable to save space. This and the renaming process described above can be done by using Microsoft's exepack function. To use the exepack function, type the following:

```
exepack TFMWRITE.EXE (file format name).EXM
```

4. Repeat steps 1 to 3 for the different formats.
5. Each of the above .EXM files makes a temporary ASCII file which contains the different tags developed. To convert the ASCII file to the binary TFM file, the WT-TFM.EXM executable module is used. Compiling this module is relatively simple. Again, Microsoft C, version 7.0 is used. To compile, type the following:

```
cl /AL WT-TFM.C IOBYTES.C
```

Typing the above line compiles and links the source code, creating an executable file, WT-TFM.EXE. Note that the same .H files are used in WT-TFM.EXE as in TFMWRITE.EXE.

6. Once the executable file is generated, it should be renamed to an .EXM file. The same procedure is followed as in step 3 except the new module is called WT-TFM.EXM. To use the exepack function, type the following:

```
exepack WT-TFM.EXE WT-TFM.EXM
```

Once all the steps are completed, the various modules used to create TFM files have been generated. To use the modules, perform the following steps:

1. Create the PARAMETERS file.
2. Spawn to the correct file format .EXM module.
3. Spawn to the WT-TFM.EXM module.

Note



All of these steps are performed automatically when using the INTRFACE.EXE program on the disks provided with this manual.

When the steps have been completed, a TFM binary file will be located in the destination directory. Using the READER.EXE program, you will be able to proof the data in the file.

Contents

Why Integrate Intellifont?	7-1
Font File Formats	7-1
File Sizes	7-4
Adding Intellifont to Your Application Software.	7-5
Requirements for Adding Intellifont	7-5
AutoFont Support Installer.	7-6
Intellifont with AutoFont Support	7-7
Intellifont Without AutoFont Support	7-7
Resources for Adding Intellifont	7-10
Screen Fonts	7-13
Internal LaserJet Printer Fonts	7-13
Bitmapped Font Cartridge Products	7-13
Scalable Typeface Products	7-14

Why Integrate Intellifont® ?

Intellifont is a set of high-performance program routines developed by Agfa to quickly and accurately scale typefaces. Intellifont is used to:

- Create screen fonts that exactly match printer fonts
- Allow users to use scalable type for those LaserJet printers that do not include Intellifont in their firmware (LaserJet series II, IIP, IID, 2000)
- Provide compatibility with other printers that accept Intellifont-compatible typefaces

Intellifont is used to create screen fonts and matching printer fonts that are compatible with PCL 4 *and* PCL 5 printers. Since the PCL 5 LaserJet printers contain Intellifont in their firmware, software applications can download scalable fonts and/or bitmap fonts to it. The LaserJet printers using PCL 4 do not accept downloaded scalable fonts – only bitmap fonts.

Intellifont is resident within HP's Type Director utility, but it is also available under license from Agfa for integration within your software. Intellifont provides an excellent solution for matching screen and printer fonts and for scaling fonts “on the fly” during the printing process.

The following discussion of file formats explains the process of converting scalable typeface files to the file formats needed by Intellifont and the PCL 5 LaserJet printers.

Font File Formats

When an end-user purchases one of HP's scalable typeface products, the typeface files are in the Intellifont FAIS format. Intellifont FAIS is an acronym for the Font Access and Interchange Standard file format developed by Agfa. Intellifont FAIS files contain font data that describes the contour (or “outline”) of each character in the font, along with some metric data describing the typeface. The FAIS files contain this contour data for every character in the

typeface *complement*, which is a superset of the various LaserJet symbol sets (ASCII, Roman-8, etc.).

There are two FAIS files for every typeface on the media: a *Font Attribute file* and a *Font Display file*. In order to be used by Intellifont, the two FAIS files must be processed through a *Loader* routine, which creates a file called a *Library file*. (The Loader routine is part of Type Director and also the AutoFont Support Installer [discussed later in this chapter], or it can be integrated into your application.)

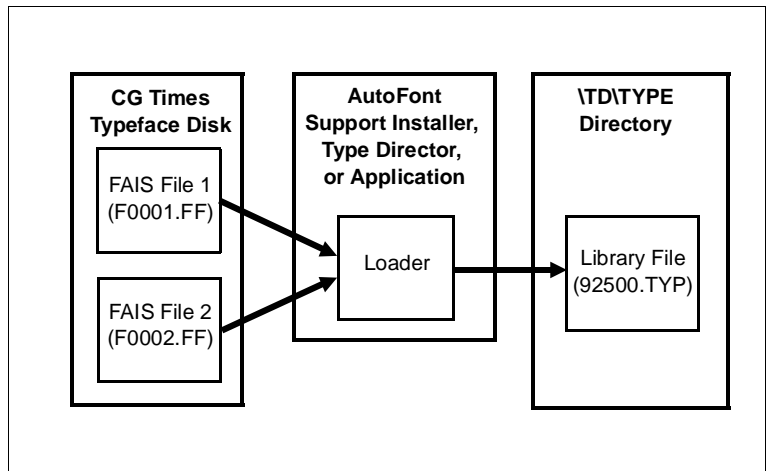


Figure 7-1. FAIS to Library File Conversion

The Loader converts the FAIS files to a Library file by generating a different header and stripping any unnecessary data from the two files. The resulting Library file contains the same character contour data as the FAIS data file it was derived from, and it contains only the information that Intellifont requires.

Note



The typefaces are shipped in the FAIS format for future compatibility, allowing Intellifont to be further improved while still being able to use the same typeface files. (Changing Intellifont requires only a Loader code change instead of obsoleting the entire typeface library.) Another reason for the FAIS format is to provide compatibility with Agfa image-setters without the purchase of new typefaces.

Creating Downloadable Fonts

Once the Library file is generated, Intellifont uses it as input to create bitmap fonts for a printer or the display screen. Although the PCL 5 LaserJet printers have Intellifont incorporated within their firmware, a Library file cannot be downloaded directly to the printer. Before a scalable font can be downloaded to a PCL 5 printer, it must have:

- PCL font and character descriptor information
- Symbol set information (if the scalable font will be bound to a particular symbol set). If a symbol set is requested, only characters for the requested symbol set are downloaded.

Downloadable scalable fonts for the PCL 5 LaserJet printers are made using a program module called “Rambo”, which extracts from a Library file the contour data for only those characters in the requested symbol set, and encapsulates the file with a PCL header. The resulting scalable font file can then be downloaded to a PCL 5 LaserJet printer and scaled to any size for printing (see Figure 7-2).

As mentioned earlier, to send the font to a PCL 4 LaserJet printer, the Library file must first be scaled and converted to a bitmap using Intellifont. A user can do this using either the Type Director utility or using your application (with Intellifont integrated into it). From an end-user’s perspective, incorporating Intellifont is the most desirable solution, allowing WYSIWYG screen and printer fonts; your application can use quality screen fonts and then scale matching printer fonts as part of the print preparation process. The user doesn’t have to use another external utility, such as Type Director.

Building bitmapped fonts with Type Director requires that the user either create the needed point sizes ahead of time, or exit the application, load Type Director, and then create them. Compared to an application incorporating Intellifont, the Type Director solution is a separate process, uses more disk space, and is less user-friendly.

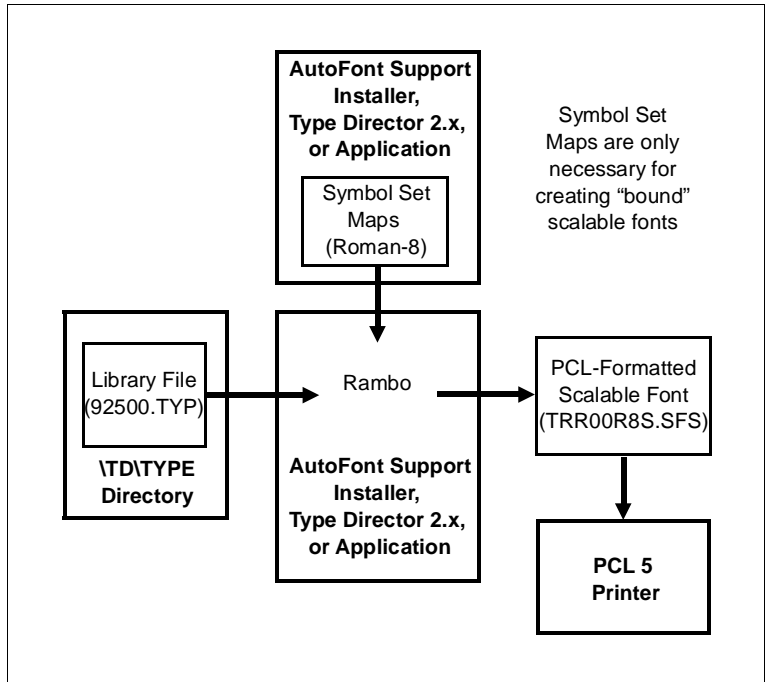


Figure 7-2. Library to PCL Format Conversion

File Sizes

The list below contains approximate sizes for the different types of files just discussed:

- Library files (~70 – 80 Kb [typeface-dependent])
- TFM files (~25 Kb for scalable typefaces; ~3 Kb for bitmap fonts [these are symbol set dependent])

- Bound scalable font files (typeface and symbol set dependent: CG Times: ASCII [35 Kb]; Roman-8 [62 Kb]; PC-8 [83 Kb])
- Unbound scalable font files (approximately three times larger than bound fonts, depending on the number of symbols—for Rambo, they are about three times larger)

Adding Intellifont to Your Application Software

Integrating the *Intellifont* font scaling technology into your software application allows you to create screen fonts to match printer fonts for PCL 4 printers, PCL 5 printers, or any other printer that accepts Intellifont fonts. Intellifont font scaling capability adds value to your software, making it easier for your customers to use fonts.

Requirements for Adding Intellifont

Incorporating Intellifont means that, besides adding the Intellifont program code to your software, you will also need access to these font file formats:

- Library files for making screen bitmaps or printer bitmaps (provided by the Loader program module)
- Symbol set maps to make the scalable font files or bitmaps (symbol set maps are provided with the BUILDSYM utility and Type Director 2.x)
- Scalable font files for downloading to the PCL 5 LaserJet printers (Rambo creates these from Library files)
- Scaled bitmap fonts for downloading to a LaserJet series II or other PCL 4 printer (Intellifont creates these from Library files)
- TFM files for font metrics (these are AutoFont Support files provided with every HP accessory font product)

AutoFont Support Installer

One way to access all of these file formats is to incorporate Intellifont and let HP's AutoFont Support Installer provide you with the other necessary resources. The AutoFont Support Installer utility ships with every HP font product sold. For both bitmapped and scalable cartridge products, it loads the TFM resources on the hard disk. For disk-based scalable products, the AutoFont Support Installer loads the TFM resources on the hard disk, creates Library files and stores them on the hard disk, and creates scalable font files for downloading to the PCL 5 LaserJet printers. In essence, the AutoFont Support Installer performs the functions of the Loader and Rambo programs and also loads TFM files.

Note



The AutoFont Support Installer differs from Type Director in that it only installs and converts fonts; it does not scale fonts as Type Director does. Both utilities use the same directory structure for storing files.

Another solution is to integrate all of the required program modules into your application. This requires integrating Intellifont, the Loader, Rambo, and symbol set files. The table below briefly describes the two Intellifont integration solutions. Both solutions are discussed later in more detail.

<p>Solution 1: Integrating Intellifont Using HP's AutoFont Support Installer:</p>
<ul style="list-style-type: none">• Integrate Intellifont code (25K)• Rely on AutoFont Support Installer to load TFM files, create Library files and scalable font files.
<p>Solution 2: Integrating Intellifont <i>Without</i> Relying on HP's AutoFont Support Installer:</p>
<ul style="list-style-type: none">• Integrate Intellifont Code (25K)• Integrate Loader (33Kb) for creating Library files• Integrate Rambo (33Kb) for creating scalable fonts• Integrate Symbol Set Maps (40Kb for all maps) for use with Rambo (all symbol maps are required)

Integrating Intellifont Using HP's AutoFont Support Installer

As shown in Figure 7-3, HP's AutoFont Support Installer is used to create Library files, which are then scaled by Intellifont and converted into bitmapped images compatible with any printer that accepts LaserJet-format fonts. The Library files are also used to create screen fonts for the computer display screen. Since the AutoFont Support Installer is shipped with all HP font products, there is no need to integrate the Rambo program module or the Loader.

The benefits of relying on HP's AutoFont Support Installer are:

- You have less code to integrate into your software.
- It provides a standard directory structure where various types of files are always located.
- End-user font installation is more of a standard process, eliminating the need for users to learn a new font loading procedure for every software product.

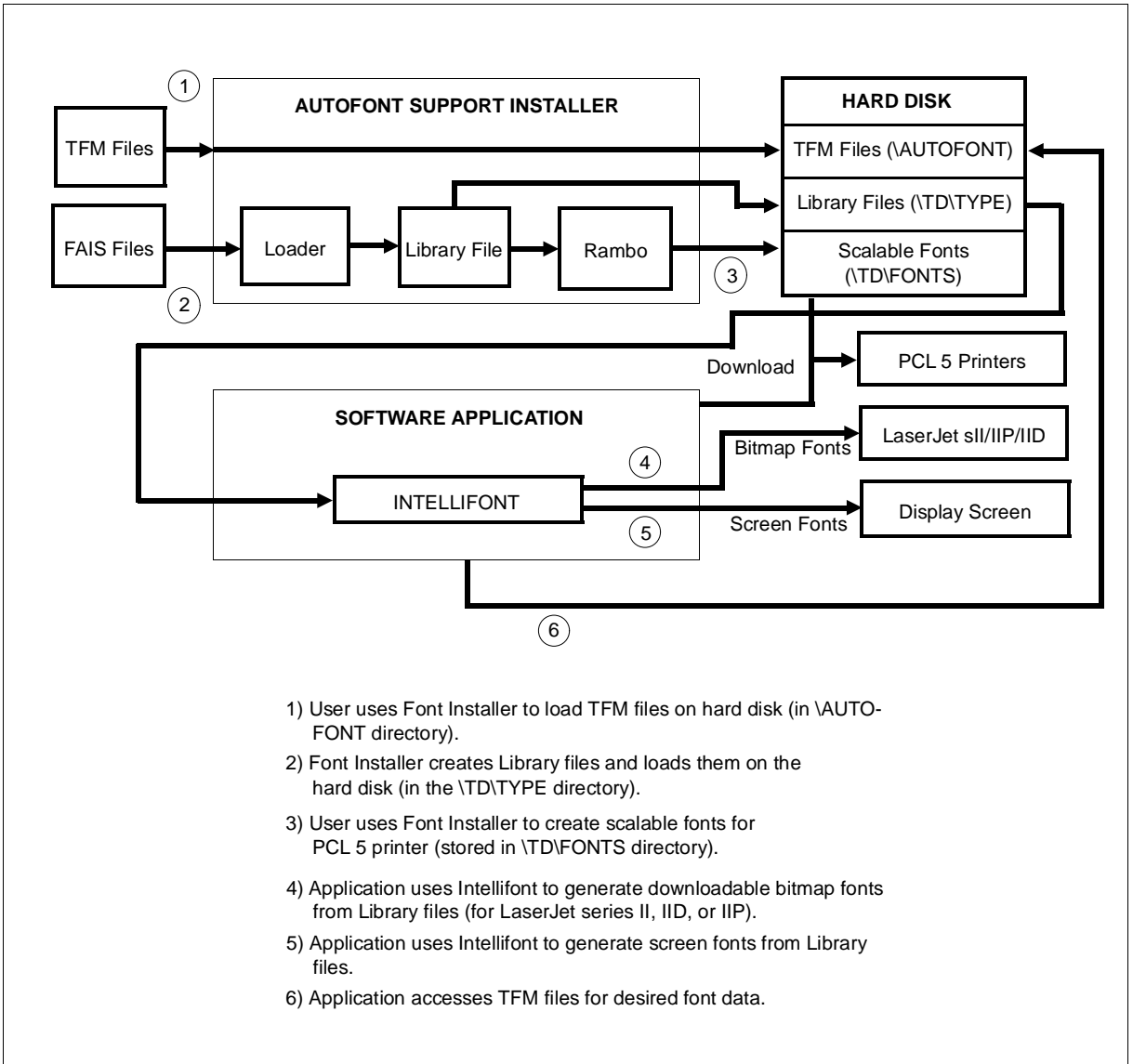
Integrating Intellifont Without Relying on HP's AutoFont Support Installer

The second scenario for incorporating Intellifont involves adding the Loader and Rambo program modules to your application, bypassing the use of HP's AutoFont Support Installer utility. This solution is illustrated in Figure 7-4.

As shown in the illustration, this solution involves incorporating the following tools:

- Intellifont code
- Loader program module
- Rambo program module
- Symbol set maps (for those symbol sets you want for screen fonts and PCL 4-compatible fonts; these maps are not necessary for PCL 5 LaserJet printer fonts)

The benefit of incorporating the Loader and Rambo into the application is that you have a higher degree of control over font operations. However, this solution requires more development time and occupies more code space than using the AutoFont Support Installer.



- 1) User uses Font Installer to load TFM files on hard disk (in \AUTO-FONT directory).
- 2) Font Installer creates Library files and loads them on the hard disk (in the \TD\TYPE directory).
- 3) User uses Font Installer to create scalable fonts for PCL 5 printer (stored in \TD\FONTS directory).
- 4) Application uses Intellifont to generate downloadable bitmap fonts from Library files (for LaserJet series II, IID, or IIP).
- 5) Application uses Intellifont to generate screen fonts from Library files.
- 6) Application accesses TFM files for desired font data.

Figure 7- 3. Intellifont Integration Using AutoFont Support

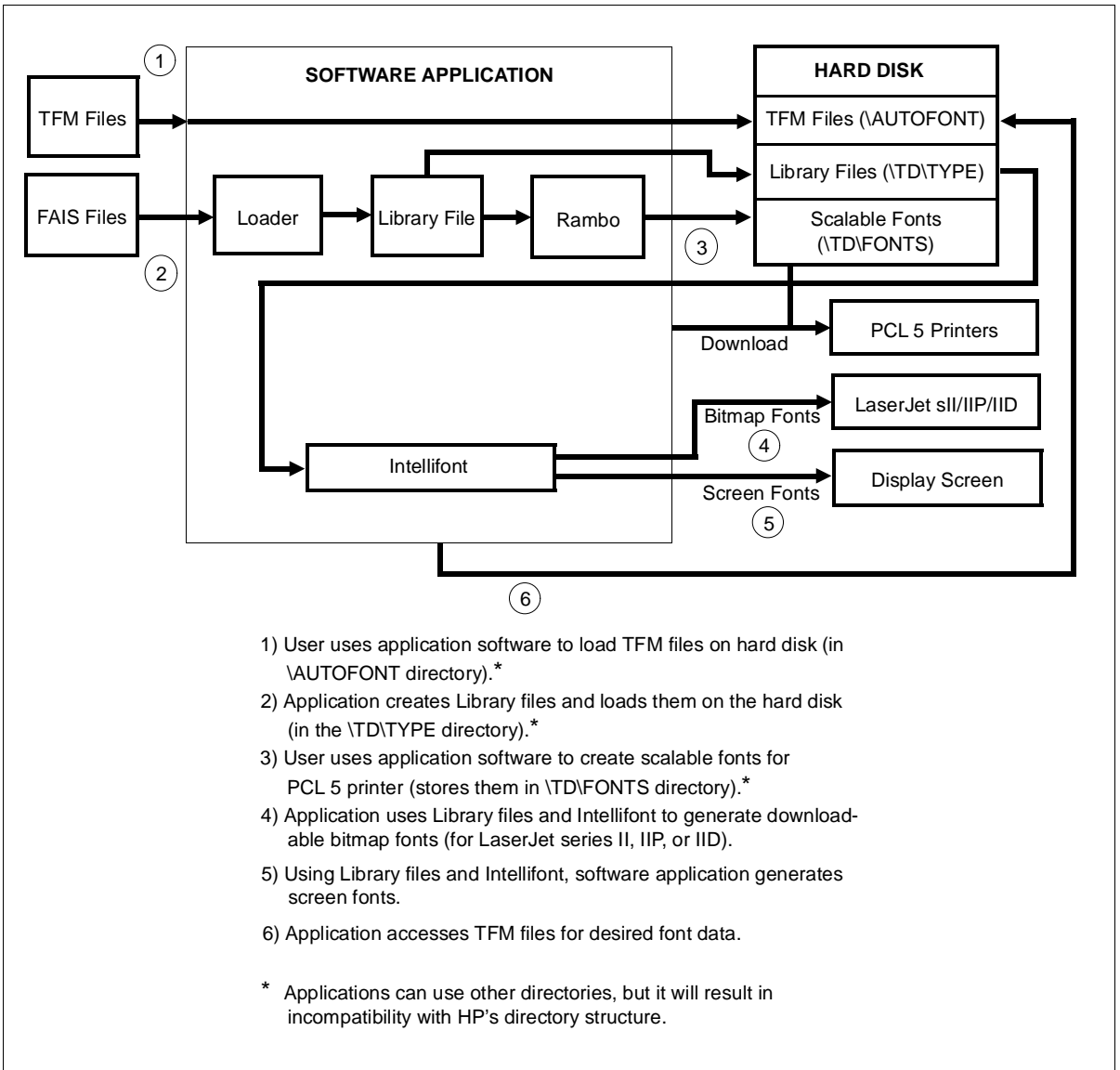


Figure 7- 4. Intellifont Integration w/o AutoFont Support

Resources for Adding Intellifont

This section describes the tools just mentioned for integrating Intellifont into your application software.

AutoFont Support Installer

The *AutoFont Support Installer* is shipped with all HP font products and performs several functions:

- It copies TFM files to the \AUTOFONT directory.
- It converts FAIS files to Library files and stores them on disk in the \TD\TYPE directory.
- It converts Library Files to scalable font files in the requested symbol set and stores them on disk in the \TD\FONTS directory.

As previously mentioned, since the AutoFont Support Installer performs the functions of the Loader and Rambo code, developers have a choice of either letting end-users use the AutoFont Support Installer for the necessary resources or integrating the Loader and Rambo into their application.

Intellifont

Intellifont is a font scaling subsystem capable of generating high-quality character images in a variety of output and graphic formats. The Intellifont subsystem is a group of callable routines implemented in the C programming language to facilitate its portability. Its function is to provide definitions of character forms and metrics in various formats to support multiple technologies. A user of the sub-system directs its operation by passing it a set of defined parameters. The subsystem executes and provides the requested character image and metric definition. The Intellifont subsystem is not meant to be a free-standing program and does not position or image characters. These functions are the responsibility of the calling application.

The Intellifont subsystem that can be used for integrating into application software is called Bullet, a small (~25 Kb) and fast version of Intellifont. The Bullet product is available from and licensed by Agfa. For more information on in-

tegrating Intellifont, call Agfa at (508) 658-5600 and ask for the OEM Technical Support Manager.

The Loader

The *Loader* program converts FAIS typeface files to a Library file. Refer to the beginning of this chapter for more information about the FAIS to Library file conversion process. The Loader is obtained from Agfa with the Intellifont code and is approximately 33 Kb in size.

The Rambo Program Code

The Rambo program converts a Library file to a PCL scalable font file which may be downloaded to a PCL 5 device (LaserJet III, IIID, IIIP, IIISi, 4). The Rambo program and source code are provided on disk by HP. The size of Rambo is approximately 33 Kb, and is dependent on the compilation method. Implementing the Rambo code is described in detail in *Integrating Rambo* (Appendix C).

Symbol Set Files

HP supplies the symbol set files that Rambo uses (if “bound” symbol sets are desired) to create scalable fonts using specified symbol sets. The symbol set files are included with Rambo, Type Director 2.x, and the AutoFont Installer. The symbol set files occupy approximately 40 Kb. Refer to the *Symbol Set Maps* discussion (later in this chapter) for more information on using the symbol set files. (The *PCL 5 Printer Language Technical Reference Manual* includes symbol set maps for all Type Director symbol sets.)

Note



The *application downloader* and *screen font formatter* are provided by the software developer. The *application downloader* performs a binary file copy to download bitmap fonts and scalable typefaces to the printer. The *application screen formatter* formats screen fonts to meet the needs of the application.

Figure 7-5 illustrates how Intellifont can be integrated into a software application. The shaded areas indicate the tools provided by Hewlett-Packard and Agfa.

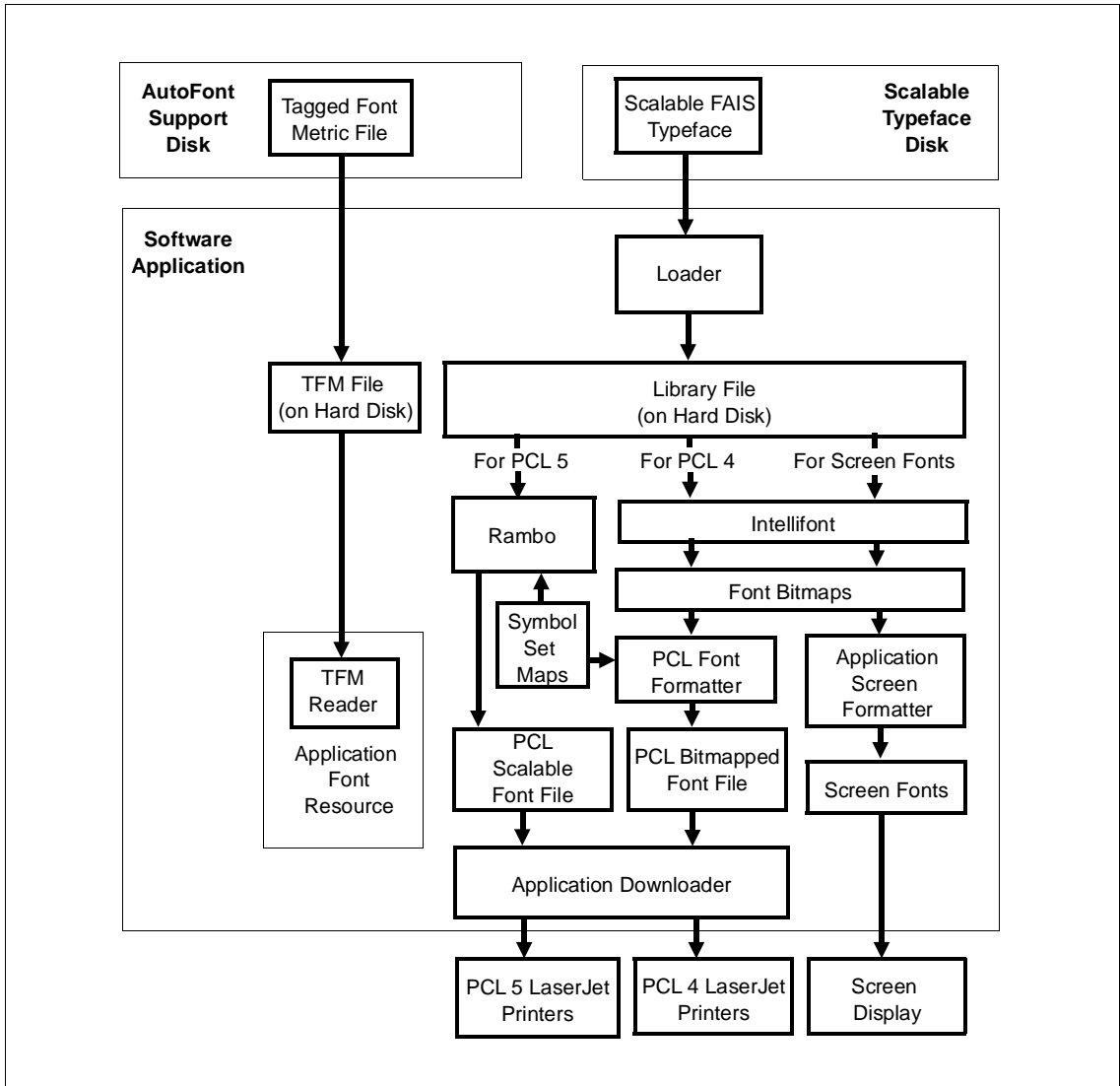


Figure 7- 5. Intellifont Integration

Screen Fonts

For many software applications, it is highly desirable to use screen fonts that match the printer fonts the user has selected. As indicated previously, screen fonts can be made using Intellifont (if it has been integrated into the application), using Type Director 2.x, or using Intellifont for Windows. The available screen fonts for HP font products are as follows:

Internal LaserJet Printer Fonts

HP's Type Director 2.x utility creates discrete sizes of screen font bitmaps in Microsoft Windows (FON) format, and in Xerox Ventura Publisher format. Software developers can make their own screen fonts in discrete sizes using Type Director 2.x; the utility ships with 4 treatments each of the CG Times and Univers typefaces. These screen fonts may be hard-coded into the LaserJet printer driver. If Intellifont is integrated into the software, Library files can be converted to screen fonts without the use of Type Director.

Note



The use of the Type Director screen fonts requires that the software developer sign a license agreement with Agfa (free of charge).

Bitmapped Font Cartridge Products

Matching screen fonts are not supplied for bitmapped font cartridge products. When using HP cartridge products, software may simulate actual printer fonts by substituting CG Times screen fonts for all serified fonts and Univers screen fonts for all sans-serif cartridge fonts. For example, for a Triumvirate font cartridge, the software could use the Univers screen font while using the Triumvirate font metrics (supplied to the end-user on the AutoFont Support disk included with their font purchase). Since Univers and Triumvirate are both sans serif fonts, the end user would see a screen font that approximates the design of the printer font.

Scalable Typeface Products

Matching screen fonts for disk-based scalable typeface products are obtained using Intellifont (for those applications that have integrated it), through the use of the Type Director program, or through Intellifont for Windows. Type Director outputs screen font bitmaps in the Windows (FON) and Ventura formats in discrete point sizes as specified by the end-user. For scalable cartridge products, the disk-based typefaces are also shipped in the same package to allow access to screen fonts in the same manner.

Note



For more screen font information and for free licensing rights for screen fonts, contact the Agfa OEM Technical Support Manager at 508-658-5600, x2218.

Contents

Introduction.	8-1
TrueType Information for TFM tags	8-1
Examples: Using TFM Values in Calculations	8-8
TrueType Method: 11 point at 300 dpi, 600 ppi . . .	8-9
TrueType Method: 11 point at 600 dpi, 600 ppi . .	8-10
The TFM Writer Development Environment	8-10
Compiler Version	8-10
Description of Files	8-11
Compiling the TFM Writer	8-12
Generating a TrueType TFM.	8-12

Introduction

Chapter 6 discusses AutoFont support for Intellifont FAIS files, Intellifont Library files, and PCL Bitmap files. In addition to supporting these files, TFM generation code has been enhanced so that it is capable of generating TFM files from TrueType typeface files. This chapter describes each tag contained in a TFM file created from a TrueType typeface, as well as describing the development environment.

This chapter builds on the TFM tag information supplied in Chapter 6.

TrueType Information for TFM tags

This section describes each TFM tag as it relates to TrueType typefaces.

Note



For more information about each tag, see Chapter 6.

- **Subfile Type (Level 1)**

Tag = 400

For TrueType typefaces, symbol set codes are mapped to Unicode values, since it is not practical to map the Unicode value to a Master Symbol List (MSL) value. This means that it is necessary to add a Subfile Type 2 to indicate that we are mapping a character to a Unicode number. Applications which currently depend on MSL numbers must be modified to interpret a Subfile Type of 2 and a Symbol Map with Unicode numbers.

- **Copyright Information (Level 3)**

Tag = 401

Copyright information is found in the “name” table of the TrueType file. The Name ID for the copyright information is zero.

- **Comment Information (Level 3)**

Tag = 402

Comment information may be found in the “name” table of the TrueType file. Name ID 4 is used to form the comment information.

- **Symbol Map (Level 1)**

Tag = 403

This map is filled with Unicode values, since a conversion from Unicode to MSL numbers is not practical. This is indicated by a new Subfile Type of 2 (Tag 400).

- **Symbol Set Directory (Level 1)**

Tag = 404

For TrueType typefaces, special .SYM files are used to map symbol set codes to Unicode values instead of CG character numbers. Unicode numbers are mapped to internal glyph indices using the Format 4 Subtable of the “cmap” table. The correct character metrics are retrieved from the “glyf” table and associated with the Unicode number in the TFM Symbol Map.

- **Unique Association ID (Level 3)**

Tag = 405

A date stamp for the TrueType typeface is created from the date and time of the TrueType typeface file.

- **Point (Level 1)**

Tag = 406

The point size used for TrueType TFM files is 1/72 inch per point.

- **Nominal Point Size (Level 1)**

Tag = 407

This tag contains the number 250.

- **Design Units (Level 1)**

Tag = 408

The Design Units tag is filled with the TrueType field *head.unitsPerEm*.

- **Type Structure (Level 1)**

Tag = 410

The Type Structure tag is filled with the field *PCLT.Style* bits 5-9.

- **Stroke Weight (Level 1)**

Tag = 411

The Stroke Weight tag is filled with the field *PCLT.StrokeWeight*.

- **Spacing (Level 1)**

Tag = 412

The Spacing tag is filled using the field *PCLT.Pitch* for fixed pitched fonts, and zero for proportionally spaced fonts.

- **Slant (Level 1)**

Tag = 413

The Slant tag is filled using the field *post.italicAngle*.

- **Appearance Width (Level 1)**

Tag = 414

The Appearance Width tag is filled using the field *PCLT.Style*, bits 2-4.

- **Serif Style (Level 1)**

Tag = 415

The Serif Style tag is filled using the field *PCLT.SerifStyle*, bits 0-5.

- **Typeface (Level 2)**

Tag = 417

The Typeface tag is filled using the field *PCLT.TypeFace*.

- **Typeface Source (Level 3)**

Tag = 418

The Typeface Source tag is filled using the unique ID string from the “name” table. This string has a Name ID of 3.

- **Average Width (Level 2)**

Tag = 419

The Average Width field is filled using the field *OS/2.xAveCharWidth*.

- **Maximum Width (Level 2)**

Tag = 420

The Maximum Width tag is filled using the maximum Blackwidth (*rightExtent* – *leftExtent*) of all the characters in the typeface.

- **Inter-word Spacing (Level 2)**

Tag = 421

The Inter-word Spacing tag is filled using the field *PCLT.Pitch*.

- **Recommended Line Spacing (Level 2)**

Tag = 422

The Recommended Line Spacing tag is filled by using the computation *head.unitsPerEm* * 1.2.

- **Capheight (Level 2)**

Tag = 423

The Capheight tag is filled with the field *PCLT.CapHeight*.

- **x-height (Level 2)**

Tag = 424

The x-height tag is filled with the field *PCLT.xHeight*.

- **Ascent (Level 2)**

Tag = 425

The Ascent tag is filled using the field *OS/2.sTypoAscender*.

- **Descent (Level 2)**

Tag = 426

The Descent tag is filled using the field *OS/2.sTypoDescender*.

- **Lowercase Ascent (Level 2)**

Tag = 427

The Lowercase Ascent tag is filled using the highest ascent for the lowercase letters a – z.

- **Lowercase Descent (Level 2)**

Tag = 428

The Lowercase Descent tag is filled using the lowest descent for the lowercase letters a – z.

- **Underscore Depth (Level 1)**

Tag = 429

The Underscore Depth tag is filled with the value – (*head.unitsPerEm* / 5).

- **Underscore Thickness (Level 1)**

Tag = 430

The Underscore Thickness tag is filled with the value *head.unitsPerEm* / 20.

- **Uppercase Accent Height (Level 2)**

Tag = 431

The Uppercase Accent Height is set to the highest ascent value for the uppercase accented characters in the Unicode range 0x00c0 thru 0x00de.

- **Lowercase Accent Height (Level 2)**

Tag = 432

The Lowercase Accent Height is set to the highest ascent value for the lowercase accented characters in the Unicode range 0x00e0 thru 0x00ff.

- **Horizontal Escapement (Level 1)**

Tag = 433

The Horizontal Escapement tag is filled using the field *hmtx.hMetrics[i].advanceWidth*.

- **Vertical Escapement (Level 1)**

Tag = 434

The Vertical Escapement tag is set to zero.

- **Left Extent (Level 1)**

Tag = 435

The Left Extent tag is filled using the field *hmtx.hMetrics[i].lsb*.

- **Right Extent (Level 1)**

Tag = 436

The Right Extent tag is filled using the computation *hmtx.hMetrics[i].lsb + (glyph[i].xMax - glyph[i].xMin)*.

- **Character Ascent (Level 1)**

Tag = 437

The Character Ascent tag is filled using the field *glyph[i].yMax*.

- **Character Descent (Level 1)**

Tag = 438

The Character Descent tag is filled using the field *glyph[i].yMin*.

- **Kern Pairs (Level 2)**

Tag = 439

The Kern Pairs tag is filled using the “kern” table in the TrueType file. The Format 0 “kern” table is used.

- **Sector Kern Information (Level 2)**

Tag = 440

Sector Kern information is not available from a TrueType file.

- **Track Kern Information (Level 2)**

Tag = 441

Track Kern information is not available from a TrueType file.

- **Typeface Selection String (Level 1)**

Tag = 442

The typeface selection number is found in the field *PCLT.TypeFamily*.

- **PANOSE Information (Level 3)**

Tag = 443

The PANOSE Information is found in the structure *OS/2.panose*. It is composed of 10 unsigned bytes.

Examples: Using TFM Values in TrueType Calculations

Note



These two examples calculate E_{size} for a printer with resolution set at 600 dpi and the Unit of Measure setting at both 300 (default) and 600 PCL Units per inch.

Device dots determine the *resolution* to which the character is rendered. The PCL Units of Measure setting determines the accuracy of the *placement* of the character.

For LaserJet 4 printers, the Unit of Measure command defines how a “dot move” is interpreted. Font metrics should be calculated based on the current Unit of Measure. The default Unit of Measure is 300 PCL Units per inch. For more information, see the “Units of Movement” discussion in Chapter 4, or refer to the *PCL 5 Printer Language Technical Reference Manual*.

TrueType TFM data must be calculated by **rounding twice** to find the desired metric value.

$$E_{\text{size}} = \text{round}(c * b * \text{dpi})$$

$$x = \text{round}(d/e * E_{\text{size}} * \text{uom}/\text{dpi})$$

where:

E_{size} = size of EM in device resolution dots

c = point size of font

b = inches per point

dpi = device resolution

x = desired metric value in units of measure

d = metric value in Design Units

e = number of Design Units in the design EM

uom = PCL Units of Measure

**TrueType Method:
11 point at
600 dpi, 300 uom**

Assume the following values:

b = 1 inch/72 points (Tag 406)

c = 11 points

d = 1451 Design Units (Tag 433)

e = 2048 Design Units (Tag 408)

dpi = 600 dots per inch

uom = 300 PCL Units of Measure

EMsize = round($11 * 1/72 * 600$)

= round(91.66)

= 92 dots

x = round($1451/2048 * 92 * 300/600$)

= round(32.591)

= 33 PCL Units of Measure

**TrueType Method:
11 point at
600 dpi, 600 uom**

Assume the following values:

$b = 1 \text{ inch}/72 \text{ points (Tag 406)}$

$c = 11 \text{ points}$

$d = 1451 \text{ Design Units (Tag 433)}$

$e = 2048 \text{ Design Units (Tag 408)}$

$\text{dpi} = 600 \text{ dots per inch}$

$\text{uom} = 600 \text{ PCL Units of Measure}$

$\text{EMsize} = \text{round}(11 * 1/72 * 600)$

$= \text{round}(91.66)$

$= 92 \text{ dots}$

$x = \text{round}(1451/2048 * 92 * 600/600)$

$= \text{round}(65.181)$

$= 65 \text{ PCL Units of Measure}$

The Development Environment for the TFM Writer

In this section, the development environment for the TrueType TFM Writer source code is described.

Compiler Version

The TFM Writer code was written in C and compiled using Microsoft C version 7.0. You will need the following Microsoft Utilities to compile the TFM Writer:

- Microsoft C Compiler version 7.0
- LINK Segmented-Executable Linker version 5.10
- EXEPACK Executable File Compression Utility version 4.06
- NMAKE Program Maintenance Utility version 1.11

Description of Files The TrueType TFM Writer source code involves the following source files:

File Name	Description
DEVELOP.C	Code which creates an intermediate ASCII version of the TFM file from the information which is passed from the code in tfmwrite.c. The code in wt-tfm.c takes this ASCII file and converts it into a binary TFM file.
FONTALIA.TT	An ASCII file which contains HP Font Alias information for those TrueType files without a PCLT table.
INTRFACE.C	Code which provides an interface around the executables for the different typeface file formats. The executable modules bitmap.exm, nsbitmap.exm, fais.exm, library.exm, truetype.exm, and wt-tfm.exm are called by this code.
IOBYTES.C	Code which provides file reading and writing routines for both Intel and Motorola byte ordering. This code is used by wt-tfm.c and the TrueType compilation of tfmwrite.c.
TFM.ERR	Strings describing error conditions which may occur during TFM generation.
TFMDEF.H	Header file containing general TFM-related definitions.
TFMERROR.H	Header file containing the definitions of error codes which may be returned by functions during TFM generation.
TFMGLOB.H	Header file containing global variable definitions for the TFM generator.
TFMTAG.H	Header file containing definitions of the TFM tag numbers.
TFMWRITE.C	Code which extracts necessary information for TFM generation from the typeface file. This information is passed to code in develop.c.
TIMEG.TTF	A TrueType typeface file containing the Universal glyphs for the TrueType typefaces.
TTDEFS.H	Header file containing structure definitions for the TrueType version of the TFM generator.
TTMAK	An NMAKE make file for the TrueType version of the TFM generator. To compile the TrueType version of the TFM generator, change the “#define” line at the beginning of the tfmdef.h file to “#define TT”. Then execute “NMAKE /F TTMAK”.
WT-TFM.C	Code which creates the binary TFM file from the information in the ASCII file, which was created by the code in develop.c.

Compiling the TFM Writer

For your convenience, there exists an NMAKE make file for the TrueType version of the TFM Writer. You will need the programming utilities described in the section above on “Compiler Version”. To compile the TrueType version of the TFM generator, change the “#define” line at the beginning of the tfmdef.h file to “#define TT”. Then execute “NMAKE /F TTMAK”.

Generating a TrueType TFM

Before running the TFM Writer, check the TTSYM subdirectory to ensure that the correct symbol set description files are present. The TFM Writer generates symbol set information for each SYM file found in the TTSYM subdirectory.

By default, TFM files are created in the FILES subdirectory. Ensure that the FILES subdirectory exists.

To create a TFM file, run the INTRFACE.EXE shell by typing INTRFACE at the DOS prompt. The first prompt in the INTRFACE shell asks you from which typeface file format you wish to generate a TFM file. Type “T” to indicate a TrueType typeface.

Next, the shell prompts you for the path to your directory containing the typeface for which you wish to generate a TFM. For example, the subdirectory TTF could be created and used to store TrueType typeface files. Type TTF to indicate the TTF subdirectory. You will then be prompted for the filename of the typeface file.

You will then receive a series of prompts about path and file names. The first prompt will ask you for a path to the directory containing the symbol set description files. Press return to accept the default subdirectory of TTSYM. The second prompt will ask you for a path and file name for the Universal glyph file. Press return to accept the default file name of timeg.ttf. The third prompt will ask you for a path and file name in which to store an intermediate ASCII file containing TFM information. Press return to accept the default file name of RESULTS.TAG. The fourth and final prompt asks for a path name to a subdirectory in which to

create the TFM file. Accept the default subdirectory of FILES.

Following the process just described, a TFM file is created in the FILES subdirectory. You are then prompted whether you wish to create another TFM file. Type N to indicate no, which returns you to the DOS prompt.

Contents

Introduction	9-1
The Raster Graphics System	9-1
Using Raster Graphics	9-3
Raster Compression Modes	9-5
Simple Binary Transfer (Mode 0)	9-5
Run-Length Encoding (Mode 1)	9-6
TIFF Encoding (Mode 2)	9-7
Delta Row Compression (Mode 3)	9-8
Adaptive Compression (Mode 5)	9-12
Compression Mode Performance	9-15
End Raster Graphics	9-16
Positioning the Cursor for Raster Graphics	9-17
Merging Text With Raster Graphics	9-17
Auto-Rotation of Raster Images	9-20

Introduction

The PCL 5 LaserJet printers provide significant enhancements over PCL 4 printers. Many of these enhancements are graphics-related. First of all, the PCL 5 LaserJet printers have an improved raster graphics feature set providing a considerable performance increase over previous raster graphics implementations. Next, a set of features called the Print Model allows users to fill images, rectangles, and even fonts with shades and patterns, and to control the interaction between the overlapping images. Finally, to support many business graphics applications and to provide high-performance graphics capabilities, the PCL 5 LaserJet printers support HP-GL/2 vector graphics commands.

The subject of PCL 5 graphics is covered in three chapters. This chapter discusses using raster graphics, Chapter 10 explains the HP-GL/2 graphics capabilities, and Chapter 11 discusses the Print Model.

The Raster Graphics System

The PCL 5 LaserJet printers print raster images using what is called the *Raster Graphics System*. The Raster Graphics System provides the necessary tools to use raster graphics as an organized system. (HP-GL/2 is used much the same way—as a coherent graphics system within the PCL printer language.) The Raster Graphics System is the term which encompasses the printer's raster graphics printing capabilities, and includes the combination of PCL 4 raster functionality with the addition of some more powerful features.

Using the Raster Graphics System, images can be defined using the concept of a *Raster Graphics Picture*. The image area of the picture is defined using the *raster height* and *raster width* commands.

By defining the image area, images can be formed without having to transmit raster data rows that are all the same

length. Using the image area as a boundary, the printer clips data rows that are too long and zero-fills rows up to the height and width boundaries of the image (on the right and bottom). In other words, unspecified raster data is automatically zero-filled where rows are shorter than the specified width and where rows are not sent to the full height (length) of the picture.

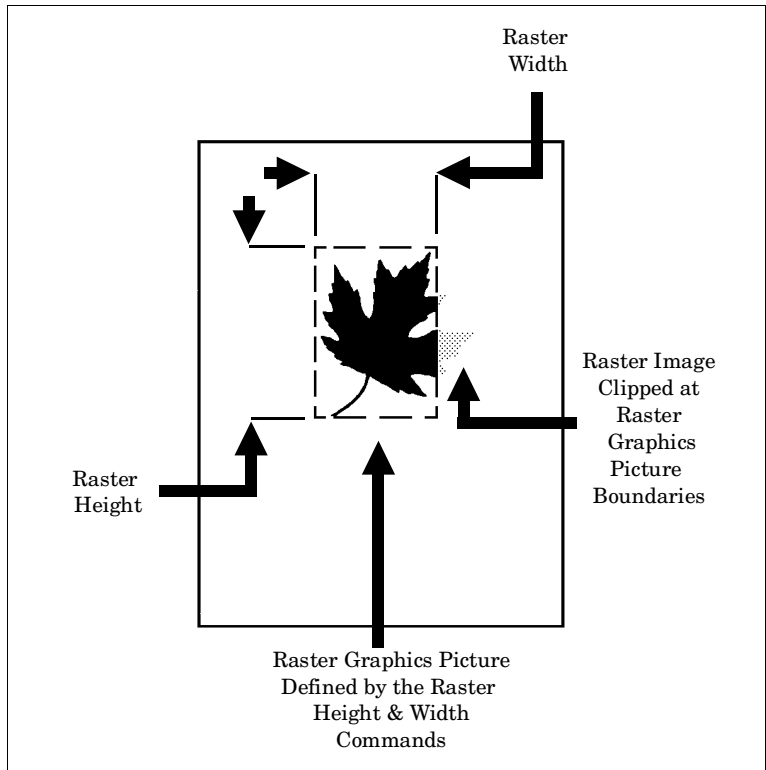


Figure 9-1. The Raster Graphics Picture

The ability to define a raster image area is augmented by four modes of raster data compression. The compression modes reduce the amount of memory needed to store raster images on the host device and increase the speed of printing those images. These added features provide significant performance benefits for raster graphics printing.

Another related performance enhancement, the Y offset command, allows the user to skip a specified number of dot rows to the starting point of the next raster line, reducing the amount of 0 byte rows needed to position a raster image.

For improved raster graphics compression, Hewlett-Packard has developed a utility called FASST which uses the best encoding method for each row, depending on graphics data and the compression modes available to the particular printer. (FASST is discussed in Appendix E.)

In total, the raster graphics capabilities of the PCL 5 LaserJet printers provide a powerful system that reduces the amount of data and time needed to print a raster image. Not only is the Raster Graphics System a powerful extension to the raster graphics capabilities of PCL 4 LaserJet printers, but it also provides opportunities for future functionality that depend on the well-defined raster area that the system employs.

Using Raster Graphics

The Raster Graphics System is made up of ten PCL commands: *start raster graphics*, *end raster graphics* (two of them), *transfer raster data*, *raster compression*, *raster presentation*, *raster resolution*, *raster height*, *raster width*, and *Y offset*. The normal execution sequence of these commands is listed below to demonstrate the Raster Graphic Picture concept:

- Raster Presentation
- Raster Resolution
- Raster Height
- Raster Width
- Start Graphics
- Raster Compression
- Transfer Raster Data Row #1

- Transfer Raster Data Row #2
- Y Offset
- Transfer Raster Data Row #3
- Raster Compression
- Transfer Raster Data Row #n
- End Graphics

Notice that the raster presentation mode, resolution, and raster area dimensions are all set outside the `start..data..end` sequence of commands and that the entire picture is sent during the `start..data..end` sequence, choosing the most effective compression mode for each raster row.

After the `start raster graphics` command, most commands other than *transfer raster data*, *data compression* commands, and the *Y offset* command imply an *end raster graphics* command. The commands for *raster presentation*, *start raster graphics*, *raster resolution*, *raster width*, and *raster height* are ignored in raster graphics mode (that is, between the *start raster graphics* and *end raster graphics* commands) and do not imply an end raster graphics command.

Raster presentation and raster resolution are true modes in the sense that, once specified, they are in that mode unless explicitly changed or reset to their default.

Note

Only raster data appearing within the intersection of the logical page, the printable area, and the raster margins are printed. If the raster width and/or the raster height have not been specified, then the intersection of the logical page and the printable area determine where the raster image appears. In any case, data outside the intersection is clipped.

Raster Compression Modes

Raster data is always sent to the LaserJet printers using the *transfer raster data* command, $\text{E}_c*\mathbf{b}\#W[\mathbf{raster\ data\ bytes}]$. The PCL 5 printers have five ways of interpreting the raster data, four of which are data compression modes:

- Uncompressed (simple binary transfer)—Mode 0
- Run-length compression—Mode 1
- Tagged Image File Format (TIFF) version 4.0 compression (PackBits)—Mode 2
- Delta row compression—Mode 3
- Adaptive compression—Mode 5

The following list shows the graphics compression modes supported by the different LaserJet printers:

Printer	Modes
LaserJet II, IID and earlier	0
LaserJet IIP	0, 1, 2
LaserJet IIIP, 4	0, 1, 2, 3, 5
LaserJet III, IIID, IIISi	0, 1, 2, 3

The compression mode command ($\text{E}_c*\mathbf{b}\#M$) specifies which raster compression mode will be used for the raster data that follows it. The selected compression mode stays in effect until it is explicitly changed, an *end raster graphics* command ($\text{E}_c*\mathbf{rB}$ or $?\mathbf{rC}$)* is received, or until the printer is reset. More than one compression mode may be used for a raster image, providing the capability to optimize the performance of raster graphics files.

* Use $\text{E}_c*\mathbf{rB}$ for all printers except LaserJet IIP, IIISi and 4.

Simple Binary Transfer (Mode 0)

Using the unencoded method, each bit describes a single dot. The most significant bit of the first byte, bit 7, corresponds to the first dot within that row; the least significant

Example: Run-Length Encoding

This example demonstrates run-length encoding in comparison with unencoded raster data:

	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Binary	01010101	01010101	01010101	01010101	01000001	01010100	01010100
ASCII	U	U	U	U	A	T	T

The data shown above would be sent to the printer as shown below if sent unencoded (simple binary transfer):

```
Ec*b0m7WUUUUATT
```

Using the run-length encoding scheme, the data would be sent as follows (the parentheses are shown for clarity—they would not actually be sent in the command string—also, the number in parentheses isn't the ASCII code for 3, 0, or 1, but the byte values 3, 0, and 1 respectively):

```
Ec*b1m6W(3)U(0)A(1)T
```

Tagged Image File Format (TIFF or PackBits) Encoding (Mode 2)

The TIFF compression mode is a combination of simple binary transfer and run-length encoding. Blocks of repeated bytes and blocks of non-repeated bytes are differentiated by using bit 7 of the control byte. If the value of the control byte is negative (-1 to -127), the byte following the control byte is to be repeated the absolute value of “control byte” times (a “replicate run”). If the control byte is positive (0 to 127), it is a “literal run” (that is, the control byte is followed by [control byte + 1] data bytes).

Note



A control byte of -128 is ignored; the next byte is then treated as a control byte. (The control byte is represented by two's complement.)

When using the TIFF compression mode, it is best to encode a two-byte replicate run as a repeated byte. However, when a two-byte replicate run is preceded and followed by a literal run, it is best to merge the three into one literal run. Three-byte repeats should always be encoded as replicate runs.

Note



If the byte count of the value field is met before the literal or replicate run count is met, the byte count has precedence. Once the byte count is met, if the last byte encountered is a control byte, it will be ignored.

Example: TIFF (PackBits) Encoding

This example shows the use of TIFF v. 4.0 encoding to compress raster data:

	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Binary	01010101	01010101	01010101	01010101	01000001	01010100	01010100
ASCII	U	U	U	U	A	T	T

The data shown above would be sent as follows using a simple binary transfer (unencoded):

```
Ec*b0m7WUUUUATT
```

Using TIFF v. 4.0 encoding, the data would be sent as follows:

```
Ec*b2m6W(-3)U(0)A(-1)T or ?*b2m6W(-3)U(2)ATT
```

Delta Row Compression (Mode 3)

Delta row compression is quite different than the previous two compression modes. As its name implies, delta row compression is accomplished by encoding the changes in one raster row compared to the previous row of data (the *seed* row). Delta row compression generally offers the largest degree of performance gain compared to unencoded data, but requires a slightly more detailed explanation to grasp the concept.

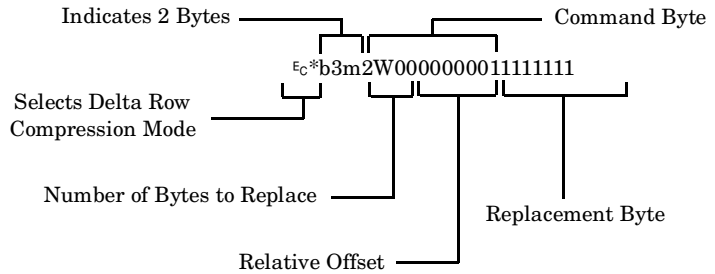
For delta row compression, the raster data is segmented into:

- Command byte(s)
- Replacement byte(s).

The command byte is segmented as follows: the first three bits (most significant bits) indicate the number of consecutive bytes to replace and can range from 0 to 7 (0 means replace 1 byte, and 7 means replace 8 bytes). The lower five

bits (least significant bits) contain the relative offset to the first byte to replace. The replacement byte(s) follow the command byte(s).

The illustration below shows how the transfer raster data command is structured for delta row compression:



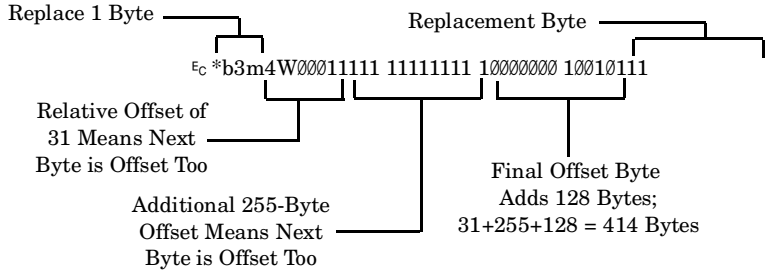
As mentioned, the relative offset is contained in the lower five bits of the command byte, allowing an offset value from 0 to 31 that indicates the distance to the next byte that needs replacing. The compression mode allows for offsets larger than 31 bytes as follows:

A value of 0 - 30 indicates the relative offset. A zero value means that the next untreated byte should be replaced, while a 30 indicates that the 31st byte from the present byte should be replaced.

A value of 31 indicates that the byte following the command byte is also interpreted as an offset. If the offset is too large to be indicated by the five bits of the command byte and the next whole byte, the offset byte can be set to 255, indicating that the following byte is also an offset, etc.

For example, consider the following *transfer raster data* command:

$\epsilon_c * b3m4W$ 00011111 11111111 10000000 10010111



Since the least-significant five bits of the command byte are all 1's, the following byte is also an offset byte; likewise, since all the the offset bits are also set to 1, the byte after that is an offset byte too. In total, the offset is 414 bytes (31 bytes + 255 bytes + 128 bytes = 414 bytes).

How The Raster Y Offset Command Affects The Seed Row

The seed row is affected by the Y offset. A *raster Y offset* ($\epsilon_c * b\#Y$) skips raster rows, leaving them blank (white). It also initializes the seed row to zeros.

For example, while in the *delta row compression* mode (and all other compression modes), the following commands have the described effects:

- $\epsilon_c * b0W$ Replicate the previous row. The seed row is unchanged.
- $\epsilon_c * b1Y$ Move down one raster row. The seed row is set to zeros.

The seed row is affected by every raster graphics transfer, regardless of the compression mode used. For example, an $\epsilon_c * b0W$, while using mode 1 compression (run length encoding), sets the seed row to all zeros. This allows delta row compression to be mixed with other modes in order to achieve better compression performance. Other cursor posi-

tion moves cause the seed row to be set to zeros. (Remember, non-graphic cursor moves have the same effect as an *end graphics* command.)

Note 

If the byte count of the *transfer raster data* command value field is less than the literal number of bytes that can be replaced, the byte count has precedence. Also, if the last byte is a control byte, it is ignored. Therefore, E_c*b1W does not affect the seed row, but causes the previous row to be replicated (in mode 3 only).

**Example:
Delta Row
Compression**

The following example demonstrates how to compress the following data using the delta row method of compression. (The bytes highlighted in *italic type* indicate those bytes needing replacement—that is, those bytes that are different from the previous row, the seed row.)

Byte No.	1	2	3	4	5
Row 1	00000000	<i>11111111</i>	00000000	00000000	00000000
Row 2	00000000	11111111	<i>11110000</i>	00000000	00000000
Row 3	<i>00001111</i>	11111111	11110000	<i>10101010</i>	<i>10101010</i>

E_c*r1A — The *start raster graphics* command initializes the seed row to all zeros.

Row 1 — $E_c*b3m2W(00000001)(11111111)$

The **3m** selects the delta row compression mode and the **2 W** indicates that 2 bytes of data will follow. The first three bits of the first data byte, the command byte, signify a single byte replacement (all three bits are 0). The next five bits indicate an offset of 1 byte from the current position. The replacement byte follows and contains **11111111**.

Row 2 — $E_c*b2W(00000010)(11110000)$

The first three bits of the command byte indicate that one byte will be replaced and the next five bits indicate a relative offset of 2, so the replacement will occur 2 bytes from the current position. The replacement byte follows (11110000).

Row 3 — $E_C * b5W(00000000)(00001111)(00100010)$
(10101010)(10101010)

As in the other rows, the first three bits of the command byte are zero, indicating a single byte replacement. The five offset bytes indicate a relative offset of zero bytes. The replacement byte follows and is 00001111. The third byte is another command byte and the first three bits signify the replacement of two bytes (the top three bits are 001). The offset bits indicate an offset of two bytes from the current position. The fourth and fifth bytes are the two replacement bytes.

Adaptive Compression (Mode 5)

The LaserJet IIIP and 4 printers have another compression mode called adaptive compression, or mode 5 compression. Adaptive compression enables the combined use of any of the four previous modes (0 through 3), plus it includes the ability to print empty rows (all zeros) or to duplicate rows. This compression method also allows for the transfer of raster data in a block format rather than row by row.

The adaptive compression mode minimizes the amount of data needed to define a raster image, resulting in faster I/O and minimized disk storage requirements. This compression method also reduces the amount of PCL overhead, since a PCL command is not needed for every raster row.

Using adaptive compression, the printer can usually print more raster graphics with the standard memory because it stores the raster image in its compressed form, and then decompresses it at print time. In contrast, for the other compression methods, the printer decompresses the raster data as it is received.

Minimizing Memory Requirements for the LaserJet IIIP

For the LaserJet IIIP printer, if the following guidelines are followed, an entire letter-size page of 300 dpi graphics can often be printed with the printer's base memory. (This isn't necessary for the LaserJet 4 printer, since it uses automatic data compression in "memory low" conditions.)

- The packaged mode 5 image must fit in available memory.
- The raster image must be printed in either portrait orientation, or with Presentation mode 3.
- The raster image resolution must be 300 dpi.
- The source and pattern transparency modes must both be set to transparent.
- No raster fill types other than black can be applied.
- The raster image cannot be nested in a macro.
- The raster block must begin in the printable area.

Using Adaptive Compression

Adaptive compression interprets the raster image rows as a block of raster data rather than as individual lines. The result of this interpretation is that the Transfer Raster Data ($E_c * b \# W$) command is sent only once at the beginning of a raster data transfer block. (There is an upper limit of 32 Kbytes in any block.) The Transfer Raster Data command identifies the number of bytes in the block.

For adaptive compression, the raster row format is:

<Command byte><# of bytes/row or rows affected> <Raster bytes>

The *command byte* is a single 8-bit byte.

The command byte designates either the type of compression mode, an empty row, or row duplication. Values for the command byte are shown below:

Command byte values	
0	No compression
1	Run-length compression
2	TIFF compression (version 4.0)
3	Delta Row compression
4	Empty Row
5	Duplicate Row

- The *# of bytes/row or rows affected* field is a two-byte field, with the high-order byte first. For command byte values 0 - 3 (in the table above), the # of bytes/row specifies the number of bytes for the row. For command byte values of 4 and 5, this byte identifies the number of empty or duplicate rows.
- The *raster bytes* field contains the actual data bytes that follow. No *raster bytes* fields are sent for the empty row and duplicate row command-byte values.

Empty Row

The *empty row* command-byte value (4) causes a row of zeros to be printed. The number of empty rows printed depends on the value contained in the *# of bytes/row or rows affected* field following the command byte. A range of 0 to 32767 empty rows can be printed per command. The *empty row* command byte resets the seed row to zero and updates the cursor position. No raster bytes field is sent.

Duplicate Row

The *duplicate row* command-byte value (5) causes the previous row (seed row) to be printed. The row can be duplicated the number of times indicated by the value contained in the *# of bytes/row or rows affected* field following the command byte. A range of from 0 to 32767 row duplications can be performed per command. *Duplicate row* updates the cursor position but does not change the seed row. No raster bytes field is sent.

Example: Adaptive Compression

This example shows the use of adaptive compression to encode the following four rows of raster data.

Byte No.	1	2	3	4	5
Row 1	00000000	00000000	00000000	00000000	00000000
Row 2	00000000	00000000	00000000	00000000	00000000
Row 3	01010101	01010101	01010101	01010101	01010101
Row 4	01010101	01010101	01010101	01010101	01010101

Row 1 — $\text{E}_c^*b5m11W(00000100)(00000000)(00000010)$

The 5m selects the adaptive compression mode and the 11W indicates there are 11 bytes in this raster data block. The command byte follows (value of 4), which indicates there are a number of empty rows. The following two bytes indicate the number of empty rows (2).

Row 3 — $(00000010)(00000000)(000000101)(11111100)$
 (01010101)

This row has a command byte of 2, indicating TIFF 4.0 compression, followed by two bytes indicating the number of bytes in the row (5), followed by the TIFF data (11111100) (01010101) indicating four repetitions of the following byte (11111100 indicates the 4 repetitions, and is represented as the two's complement of 4).

Row 4 — $(00000101)(00000000)(000000001)$

This row has a command byte of 5, indicating *duplicate row*, followed by two bytes indicating the number of duplicate rows (1).

Notice in this example that there is only one *transfer raster graphics* command ($\text{E}_c^*b\#W$), indicating the start of the graphics block, instead of one command at the start of every row (as is necessary with all the other compression modes). Also notice that the example uses only mode 2 (TIFF) compression, but any of the other compression modes may be used as well, as long as only one compression mode is used per raster row.

Compression Mode Performance

The choice of the best compression mode for the highest performance is dependent on several factors. In the LaserJet IIIP and LaserJet 4 printers, mode 5 (adaptive) compression is the most efficient. For the remaining printers, the best compression method depends on the particular image. For optimum raster graphics compression, Hewlett-Packard

recommends that developer's incorporate the FASST utility, which looks at each raster row and determines which compression method is the best for that row. The FASST utility is discussed in detail in Appendix E.

Note



“Performance” refers to the ability to throughput data. Compression always reduces I/O time, but may or may not reduce the printer memory requirements for a particular image. For all PCL 5 printers, when using compression modes 0 through 3, printer memory required for compressed images is the same as for unencoded images.

End Raster Graphics

The end raster graphics commands signify the end of a raster graphic data transfer. There are two raster graphics commands, E_c^*rC and E_c^*rB , and the command used is dependent on the particular LaserJet printer.

- For the LaserJet IIIP, IIISi, and LaserJet 4 printers, Hewlett-Packard recommends using the E_c^*rC end raster graphics command. (All three printers support both end raster graphics commands, but Hewlett-Packard recommends using E_c^*rC .)
- For all other LaserJet printers, use the E_c^*rB end raster graphics command.

Positioning the Cursor for Raster Graphics

When printing raster graphics, the current active cursor position (CAP) automatically moves down one dot row after receiving the final byte in the row, and then returns to the left graphics margin. Consequently, after printing a raster image, the cursor is at the left graphics margin and one dot row below the image.

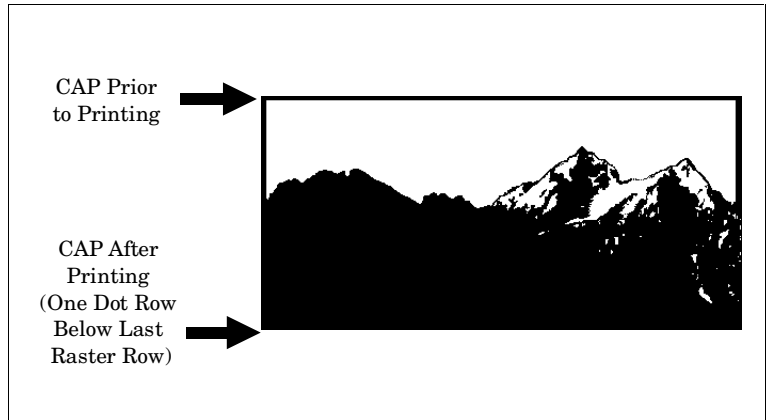


Figure 9-2. Raster Graphics Cursor Position

Note



Cursor positioning for rectangular area fill graphics is different than for raster graphics. For rectangular area fill graphics, the cursor position is that of the starting position after it prints the filled graphic.

Merging Text With Raster Graphics

Merging text and graphics involves knowing the text and graphics boundaries so that the cursor may be moved appropriately. Following the printing of text, the cursor rests at the *character reference point* of the next character.

For *fixed-pitch* fonts, the cursor moves the HMI distance for every character. For *proportional* fonts, the cursor moves the *horizontal escapement* distance; the *horizontal escapement* is one of the font metrics obtained from each

font's TFM file. (See the "TFM Reader" section in Chapter 6 for information on retrieving TFM data.)

Example: Merging Fixed-Pitch Text With Graphics

The following example illustrates merging fixed-pitch text with graphics.

ABCDEFGHIJ
KLMNOPQRST
UVWXYZABCD
EFGHIJKLMN
OPQRSTUVWXYZ



$\text{\textcircled{E}}_c(\text{s}\text{\textcircled{o}}\text{p}1\text{\textcircled{0}}\text{h}12\text{\textcircled{v}}\text{\textcircled{0}}\text{s}\text{\textcircled{0}}\text{b}3\text{T}$

Select a 10-pitch, 12-point, upright, medium Courier font.

ABCDEFGHIJ ...

The printed characters "ABCDEFGHIJ" would occupy one inch (10 characters at 10 characters/inch).

$\text{\textcircled{E}}_c*\text{p}+75\text{x}\text{\textcircled{0}}\text{Y}$

Move the cursor 75 dots (1/4 inch) to the right, relative to the cursor position after printing the text, and to the top margin ($\text{\textcircled{E}}_c*\text{p}\text{\textcircled{0}}\text{Y}$).

$\text{\textcircled{E}}_c*\text{r}\text{\textcircled{0}}\text{F}$

Set the raster presentation mode so that the image prints in the orientation of the logical page.

$\text{\textcircled{E}}_c*\text{t}3\text{\textcircled{0}}\text{R}$

Set graphics resolution to 300 dpi.

$\text{\textcircled{E}}_c*\text{r}45\text{\textcircled{0}}\text{s}225\text{T}$

Define the raster image size (1.5 inches wide by .75 inches high, which equates to 450 by 225 dots).

$\text{\textcircled{E}}_c*\text{r}1\text{A}$

Start raster graphics at the current cursor position.

$\text{\textcircled{E}}_c*\text{b}\text{\textcircled{0}}\text{M}$

Set raster compression mode for a simple binary transfer (unencoded).

$\text{\textcircled{E}}_c*\text{b}57\text{Wdata}\dots$

Transfer raster graphics data.

$\text{\textcircled{E}}_c*\text{r}\text{C}$

End raster graphics. (Use $\text{\textcircled{E}}_c*\text{r}\text{B}$ if using LaserJet III or IIID.)

Example: Merging Proportional Text With Graphics

This example demonstrates a method of merging proportional text with raster graphics:

**This is a test, test, test, test.
This is a test, test, test, test.
This is a test, test, test, test.
This is a test, test, test, test.**



- `Ⓔ(s1p10v0s3b4101T` Select a proportional, 10-point, upright, bold, CG Times font.
- This is a test, test, . . . Print the desired text (2-inch column).
- `Ⓔ*p750x0Y` Move the cursor to the top margin and to a horizontal position 750 dots from the left logical page boundary (2-inch column + 1/2-inch space between text and graphics = 2.5 inches = 750 dots).
- `Ⓔ*r0F` Set the raster presentation mode to print in the orientation of the logical page.
- `Ⓔ*t300R` Set raster resolution to 300 dots per inch.
- `Ⓔ*r450s225T` Define the raster image size (1.5 inches wide by .75 inches high, which equates to 450 by 225 dots).
- `Ⓔ*r1A` Start raster graphics at the current cursor position.
- `Ⓔ*b0M` Set raster compression mode for simple binary transfer (unencoded).
- `Ⓔ*b57Wdata...` Transfer raster graphics data.
- `Ⓔ*rC` End raster graphics. (Use `Ⓔ*rB` if using the LaserJet III or IIID printers.)

Auto-Rotation of Raster Images

The raster presentation mode command specifies the direction of print of a raster image on the logical page. The command provides two options:

- The `ESC*r0F` command prints the image in the direction of print, as defined by the print direction command (the default print direction is zero, which is the same as the logical page). For instance, let's say you are printing a newsletter on a portrait logical page; the newsletter has a scanned image in the left column. If you decide to change the logical page orientation and then print the same newsletter, the `ESC*r0F` command causes the image to rotate along with the text so that the image is at the same relative position in the left column of that print direction. For most applications, this is the way the user would want to see the image.
- The default option, `ESC*r3F`, would print the same image along the width of the physical page, (it would start printing at the current active cursor position, but would be sideways relative to the text at the new orientation). This option is beneficial for those applications requiring that an image be placed along the width of the physical page. For most applications, this is not the desired mode; it is, however, the default mode in order to provide compatibility with earlier LaserJet printers.

Figure 9-3 shows an example of both presentation modes.

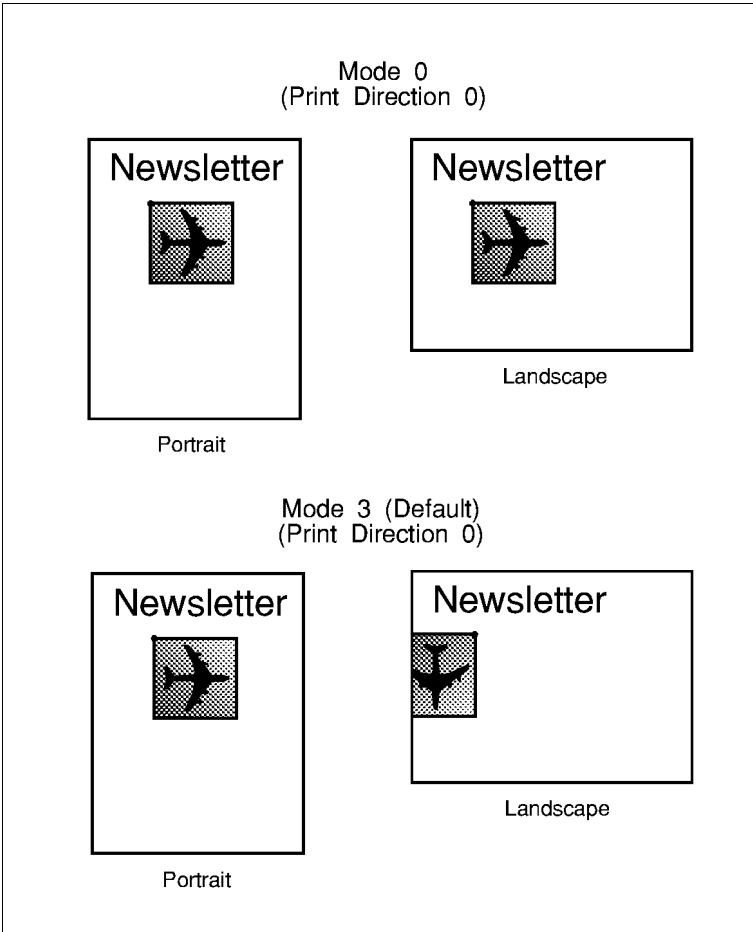


Figure 9-3. Raster Graphics Presentation Mode and Auto-Rotation

Contents

Introduction.	10-1
Additional LaserJet 4 Features.	10-1
The Picture Presentation Directives	10-2
PCL Picture Frame & HP-GL/2 Plot Size	10-3
The Picture Frame Anchor Point.	10-4
Basic Steps for Importing an HP-GL/2 Plot	10-4
Basic Steps for Creating an HP-GL/2 Plot	10-5
When to Use Vector Graphics vs. Raster Graphics . .	10-6
HP-GL/2 & PCL Orientation Interactions.	10-7
Vector Graphics Limits	10-9
The Scaling Factor and the Picture Frame.	10-10
Creating a Page-Size Independent Plot	10-10
The Scaling Mechanism.	10-11
Memory usage and HP-GL/2	10-12
Using HP-GL/2 Commands	10-13
Mixing PCL Text With HP-GL/2 Graphics	10-14
Special Font Effects Using HP-GL/2.	10-15

Introduction

As part of the enhanced PCL 5 feature set, the LaserJet printers provide vector graphics capabilities using HP-GL/2, the Hewlett-Packard Graphics Language. PCL 5 printers have the ability to print raster graphics and text on the same page as HP-GL/2 vector graphics, providing the ability to import existing HP-GL/2 graphics files or combine vector graphics functions along with the existing raster printing features.

Since PCL 5 includes HP-GL/2, you can offer some extra features to your customers that would be difficult to implement or wouldn't be feasible without HP-GL/2. Some of these features include rotating text at any angle, creating mirrored text images, combining HP-GL/2 plots with high-quality LaserJet fonts, and importing existing HP-GL/2 files. As you would expect, using HP-GL/2 you can draw circles, rectangles, lines, and bezier curves (LaserJet 4 only) much more efficiently than using raster graphics, especially at 600 dots per inch.

Additional LaserJet 4 Features

The LaserJet 4 printer provides some HP-GL/2 features that are not available with other PCL 5 printers:

- **Bezier curves**

Efficient bezier curves are created using the BZ (Bezier Absolute) and BR (Bezier Relative) commands. These two commands allow you to specify cubic parametric curves using four control points (the current pen position and three additional control points).

- **Non-zero Winding Fill Type**

The LaserJet 4 printer provides an additional fill type for filling polygons: non-zero winding fills. This type of fill gives software greater flexibility and control over fill patterns as they are applied to polygons.

- **PCL-Compatible Label Origin**

The LaserJet 4 printer provides a PCL-compatible label origin option for label placement. Using the LO21 command,

labels are rendered from the character reference point, just as they are when using PCL.

- **User-Defined Patterns for Fill Type and Screened Vectors**

Patterns created using the PCL user-defined pattern command ($\text{E}_c^*c\#W$) can be used to fill polygons and screened vectors. This is accomplished using the FT22 (Fill Type) command and the SV22 (Screened Vectors) command. (User-defined fill patterns are also supported by the LaserJet IIIP printer, however the feature was previously undocumented.)

Note



These HP-GL/2 features are described in detail in the *PCL 5 Technical Reference Manual*.

The Picture Presentation Directives

Incorporating HP-GL/2 within PCL 5 is accomplished using a group of commands called *picture presentation directives*. The *picture presentation directives* provide the means to incorporate HP-GL/2 within PCL 5, supplementing or overriding the usual HP-GL/2 functionality when necessary. They allow the application to enter and exit the HP-GL/2 environment and to define a picture frame for a vector graphic image. They also allow the printer to calculate a scaling factor with which to scale images. This enables the user to create a graph or to import an existing HP-GL/2 graph so that it can be scaled to a desired size and placed anywhere on the PCL logical page.

PCL Picture Frame & HP-GL/2 Plot Size

When importing existing HP-GL/2 images, the ability to scale them to a specific size is achieved using the *PCL Picture Frame* and the *HP-GL/2 Plot Size* commands. The picture frame commands specify the width and height of the rectangle, or picture frame, that will define the image area on the PCL page; the HP-GL/2 plot size commands specify the width and height of the original HP-GL/2 plot.

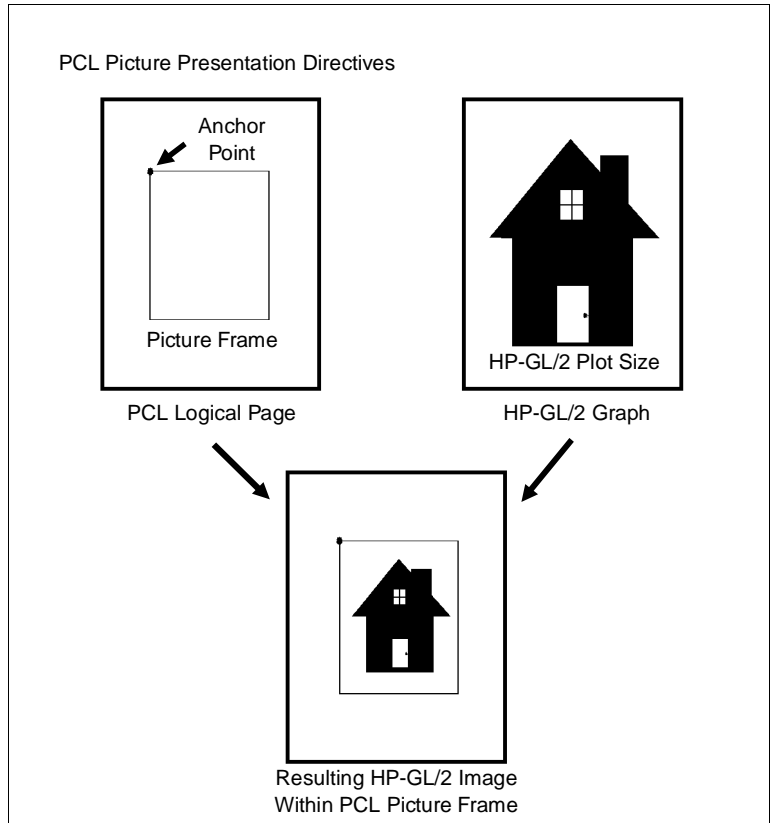


Figure 10-1. Picture Frame, Plot Size, and Anchor Point

The ratio of the PCL picture frame size to the HP-GL/2 plot size defines the *picture frame scaling factor*. (There are actu-

ally two scaling factors, one for the x direction and one for the y; to simplify this discussion, they will be referred to collectively as the *picture frame scaling factor*.)

When an HP-GL/2 plot is imported for printing on a PCL 5 printer, the plot is scaled to fit within the rectangular area referred to as the *PCL picture frame* by applying the picture frame scaling factor. After applying this scaling factor, if portions of the plot extend outside of the picture frame area, they are clipped; no part of the HP-GL/2 plot can extend beyond the picture frame.

The Picture Frame Anchor Point

As mentioned, the *PCL picture frame* and *HP-GL/2 plot size* commands determine the size of the vector image that is printed. The actual position of the picture frame within the logical page is specified using the *picture frame anchor point* command. This command positions the upper-left corner of the picture frame at the current active position in the PCL environment.

If an orientation change is desired, the orientation of the HP-GL/2 plot is specified by the HP-GL/2 Rotate command (RO) (see Figure 10-3). A change in print direction has no effect on the picture frame anchor point, the physical position of the picture frame, or the HP-GL/2 orientation.

Basic Steps for Importing an HP-GL/2 Plot

In general, follow the steps below to incorporate existing HP-GL/2 images into an application:

- Specify the picture frame dimensions (the size of the rectangular area that the image will occupy on the page).

- Specify the picture frame anchor point (the position on the logical page where the upper left corner of the picture frame is placed).
- Specify the HP-GL/2 plot size (the size of the original HP-GL/2 image).
- Enter HP-GL/2 mode.
- Include the HP-GL/2 plot.
- Enter PCL printer language mode.

Note



Before the HP-GL/2 mode is exited, the last HP-GL/2 command prior to returning to PCL mode must be terminated by a semicolon.

The commands for these steps are discussed in detail in the *PCL 5 Printer Language Technical Reference Manual* and are used in the examples in this chapter. Study the examples to get a feel for how the commands are used to print HP-GL/2 images.

Basic Steps for Creating an HP-GL/2 Plot

In general, follow the steps below to create HP-GL/2 images within your application software:

- Specify the picture frame dimensions (the size of the rectangular area that the image will occupy on the page).
- Specify the picture frame anchor point (the position on the logical page where the upper left corner of the picture frame is placed).
- Enter HP-GL/2 mode.
- Create the vector graphics plot using HP-GL/2 commands.
- Enter PCL Printer Language mode.

Note



In many cases, the default picture frame dimensions (equal to the default text area) and default picture frame anchor point (0,0) can be used when integrating HP-GL/2 graphics into a PCL document.

As mentioned previously, the last HP-GL/2 command before returning to PCL mode must be terminated by a semicolon.

When to Use Vector Graphics Instead of Raster Graphics

When deciding to print graphics in an application, the decision must be made whether to use raster graphics or vector graphics. In some cases, the choice is obvious, such as importing an existing HP-GL/2 file or using raster graphics to print a scanned photograph. In other cases, the choice is not as easy. There are some general rules, however, that can help you steer toward one mode or the other for printing certain types of graphic images.

In general, HP-GL/2 is the preferred mode when:

- The time saved by HP-GL/2 in reducing I/O data transfer and eliminating host vector-to-raster conversion exceeds the printer's vector-to-raster conversion time. For example, using HP-GL/2 to print an 8- by 10-inch plot (~ 1 Mbyte at 300 dpi and 4 Mb at 600 dpi) is preferable to using raster graphics if the user has an IBM AT or compatible and is using a serial interface.
- The image is composed of regular figures (circles, polygons, lines, or bezier curves [LaserJet 4 only]).
- The image needs to be resolution independent.

In general, compressed raster graphics is preferred for:

- Small, complex images
- Images that can't be accomplished with HP-GL/2, such as scanned photographs.

HP-GL/2 & PCL Orientation Interactions

The orientation of the HP-GL/2 coordinate system within the picture frame is determined by the orientation of the PCL logical page and the HP-GL/2 rotate command (RO). When a “RO 0;” is in effect, the origin of the HP-GL/2 coordinate system defaults to the lower-left corner of the PCL picture frame. The relationship between the PCL picture frame, picture frame anchor point, and HP-GL/2 coordinate system is shown in Figure 10-2.

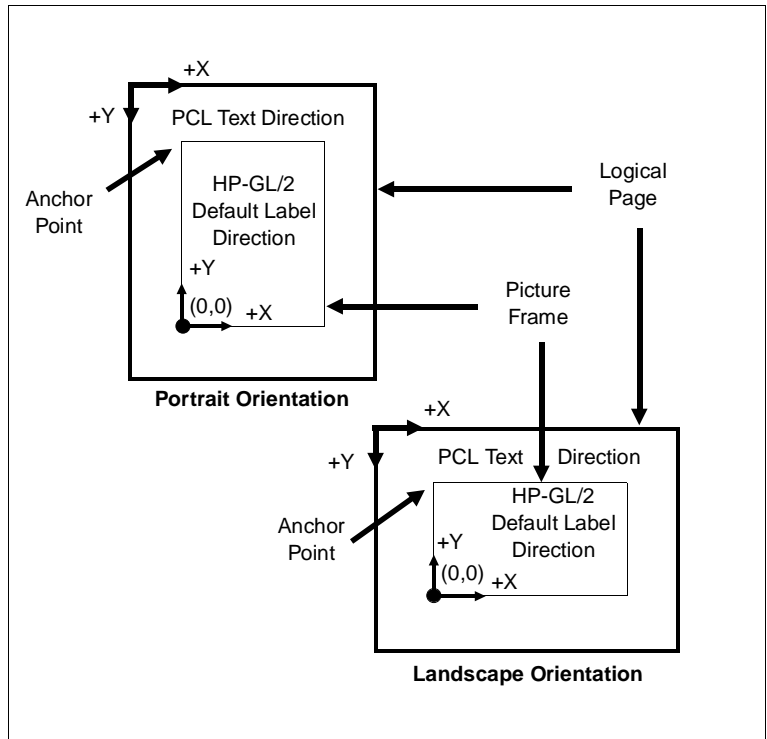


Figure 10-2. HP-GL/2 & PCL Orientation Interactions

The default HP-GL/2 coordinate system tracks the PCL logical page coordinate system so that the X axes are parallel and increasing in the same direction and the Y axes are parallel and increasing in opposite directions. This axes rela-

tionship is the default HP-GL/2 orientation. The default HP-GL/2 orientation is therefore controlled from the PCL environment by altering the default logical page orientation.

Note 

A change in print direction has no effect on the HP-GL/2 orientation, the physical position of the picture frame, or the picture frame anchor point.

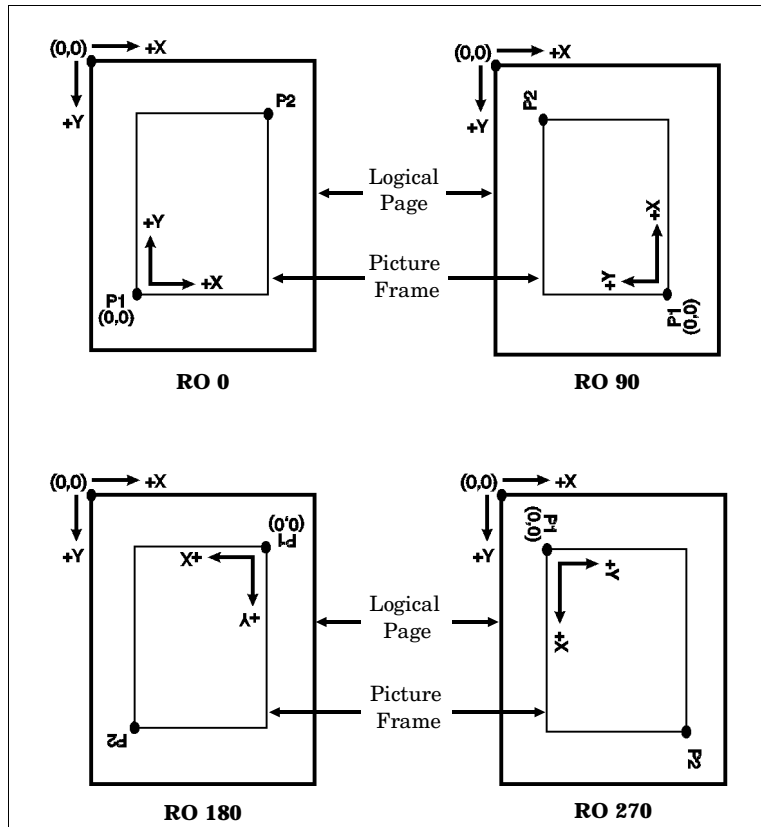


Figure 10-3. Modifying the Default HP-GL/2 Orientation

The default HP-GL/2 orientation can be modified from the HP-GL/2 environment with the HP-GL/2 rotate command (RO). Rotations specified by the RO command are relative to the default HP-GL/2 orientation. Figure 10-3 shows how the RO command modifies the default HP-GL/2 orientation.

Vector Graphics Limits

The area occupied by the HP-GL/2 image is limited by four boundaries that are imposed on the vector graphics area:

- Hard-clip Limits (Printable Area)
- Soft-clip Limits
- Logical page
- PCL Picture frame

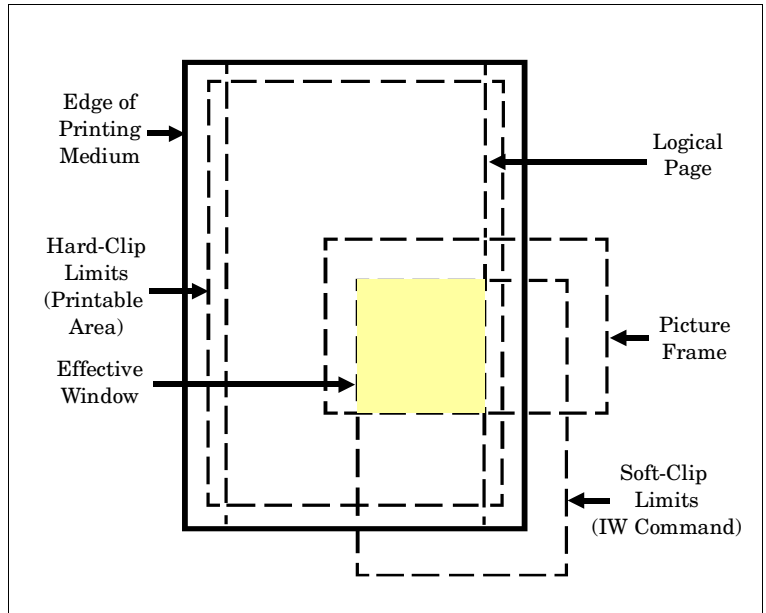


Figure 10-4. The Effective HP-GL/2 Graphics Window

The *hard-clip limit* refers to the printable area boundary, the limits beyond which the printer cannot print. The *soft-clip limit* refers to a software-controlled boundary that temporarily limits the device to a particular area defined by the HP-GL/2 IW (Input Window) command. An HP-GL/2 graphic will appear on the page only if it falls within the effective window, which is the area within the intersection of the hard-clip limits, the PCL logical page, the PCL picture frame, and the soft-clip window (see Figure 10-4).

The Scaling Factor and the Picture Frame

When printing an HP-GL/2 plot, no picture frame scaling occurs if an HP-GL/2 plot size is not specified, because the plot and the picture frame are assumed to be the same size. This is the preferred mode for:

- Integrated text and graphics (that is, those applications where the graphic image is developed as part of the overall page rather than being imported from an external source and positioned on the page).
- HP-GL/2 plots which are page-size independent.

Creating a Page-Size Independent Plot

If an HP-GL/2 plot is page-size independent, it can be automatically scaled to fit different page sizes without specifying the HP-GL/2 plot size. In order for a plot to be page-size independent, it must not specify any parameters in absolute units. This implies that:

- No parameter of any command is in plotter units. The scaled mode (SC command) must be used exclusively; either the default locations of P1 and P2 are used or their positions are specified with the IR (input relative P1 and P2) command. The default window is used or the window is specified in user units.
- For labels, only the SR (relative character size) mode is used; the SI (absolute character size) mode is not used.

- The pen width unit selection mode (WU) is specified as relative instead of metric.
- The pattern length for line types (LT) is specified as relative instead of metric.

If a plot does not meet the above criteria and the plot is not the same size as the picture frame, the HP-GL/2 plot size must be specified in order to accomplish the desired scaling. If it is not specified, the plot is clipped to the effective window, and no scaling occurs.

The Scaling Mechanism

The scaling mechanism is a straightforward linear transformation; that is, the aspect ratio of the original plot is not maintained unless the picture frame has the same aspect ratio as the HP-GL/2 plot size. Labels (text) printed using a bitmapped font may not respond accurately to picture frame scaling, so scalable fonts are recommended. As with the SI and SR commands, the raster font which most closely matches the desired character size is used.

HP-GL/2 uses four types of units when creating plots:

- Absolute units
- Plotter units (plu)
- User units
- Picture frame units

The printer chooses the type of units using the following criteria:

- If the picture frame is the same size as the HP-GL/2 plot (no scaling is required), and user scaling is not in effect, units are *plotter units*.
- If the picture frame is not the same size as the HP-GL/2 plot, but user scaling is not in effect, units are *picture frame units*. In effect, picture frame units are plotter units multiplied by the current picture frame scaling factor.
- If user scaling is in effect (that is, if an SC command has been issued), units are in *user units*.

- Some commands operate using only *absolute units* and not user units. For example, the IP (Input P1 and P2) command parameters are always plotter units. Likewise, SI (Absolute Character Size) parameters are always in centimeters. The picture frame scaling factor is always applied to parameters of commands in this category.

Memory usage and HP-GL/2

To completely eliminate the possibility of print overrun (Error 21), the PCL 5 LaserJet printers offer the *page protection* feature option. This option is available for those users that install at least an extra 1Mbyte of printer memory (except LaserJet 4). (See the section on *Memory Usage* in Chapter 2 for more information about page protection.)

Note



The LaserJet 4 printer has enough base memory to allow page protection at 300 dpi, but requires additional memory at 600 dpi.

The LaserJet 4 printer allows applications to turn page protection on and off using PJJ. See the *PJL Technical Reference Manual* for more information.

For those users printing without page protection, efficient programming can help lessen the possibility of print overrun, and can also increase printer performance. Tips for enhancing performance are discussed in Chapter 13 under *Vector Graphics*.

Note



“Stroke to fill” techniques are not as efficient as the use of filled structures (circles, wedges, polygons and fonts) and should be avoided as they consume large amounts of user memory and increase processing time. *LaserJet printers do not emulate plotter memory management techniques.*

Using HP-GL/2 Commands

There are two classes of vector graphics commands:

- PCL printer language commands
- HP-GL/2 commands

The *PCL printer language* commands include the *picture presentation directives* that allow the user to enter HP-GL/2 mode and describe the plot size, picture frame size and anchor point of the HP-GL/2 plot.

The *HP-GL/2* commands include the kernel command set plus the dual-context and palette extensions to the language. These command extensions were developed to interact with the PCL environment, such as the command used to enter the PCL mode.

This portion of the chapter contains examples that show how the PCL Printer Language and HP-GL/2 actually work together. Examples such as printing text at any angle, printing outlined fonts, and anisotropically scaling fonts are covered to show how the PCL 5 printers use HP-GL/2.

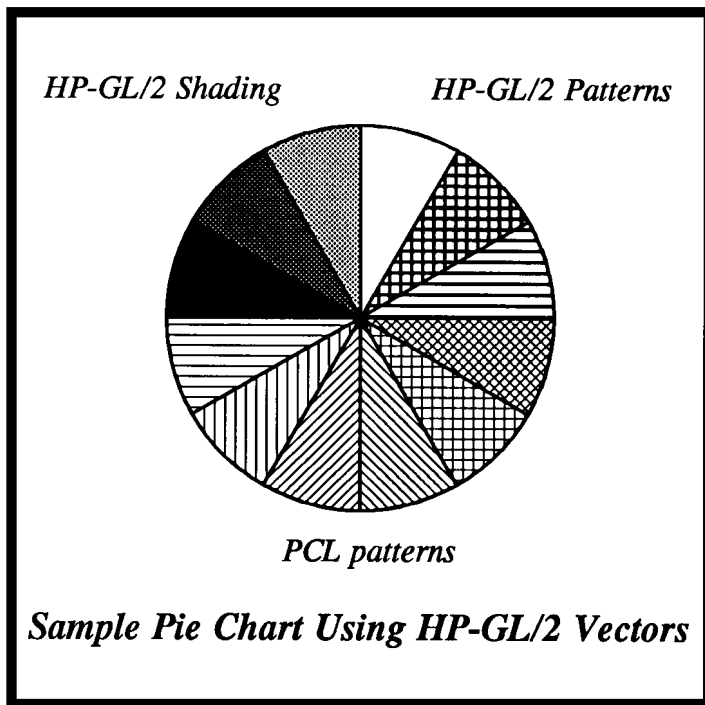
Note



For a complete description of both the PCL and HP-GL/2 commands, consult the *PCL 5 Printer Language Technical Reference Manual*. This manual contains many examples to help you learn the functionality of the HP-GL/2 command set.

Mixing PCL Text With HP-GL/2 Graphics

The PCL 5 LaserJet printers provide the ability to combine HP-GL/2 graphics with high-quality LaserJet fonts. The image below shows an HP-GL/2 pie chart created with LaserJet fonts, HP-GL/2 and PCL fill patterns, and HP-GL/2 shading.



The above example is a good one for demonstrating the interaction between PCL and HP-GL/2. The source and executable code for this pie chart (PIEWEDGE.C) are included on disk in the back of this manual. (For more information on the examples contained on the included disks, see Appendix B.)

Special Font Effects Using HP-GL/2

There are some font effects that can only be achieved using the built-in HP-GL/2 functionality of the PCL 5 LaserJet printers. These special effects can be applied to all available LaserJet *scalable* (not bitmapped) fonts. This is due to the printer's unique implementation of HP-GL/2. As with all HP-GL/2 functionality in the PCL 5 printers, the user must first enter the HP-GL/2 mode to access these features. The examples below demonstrate the special font functionality that is accessible using HP-GL/2.

Note



Due to the vector method used to create the HP-GL/2-enhanced fonts, there may be a noticeable difference between the print quality and throughput speed of vector-processed fonts compared to the standard method of printing fonts (that is, those fonts that are printed at 0, 90, 180, or 270 degrees and are proportionately [isotropically] scaled using PCL). (Vector-processed fonts may not be formed quite as precisely and may print slower compared to the standard method of printing fonts; they are also more likely to cause an Error 21 when page protection is off.) These shortcomings are usually not crucial, but depend on the application—the added HP-GL/2 features provide benefits that frequently override the associated limitations.

Printing Text at Any Angle

Using only the PCL printer language, the PCL 5 LaserJet printers can print text in four directions: 0, 90, 180, and 270 degrees. Using the HP-GL/2 DI (Absolute Direction) or DR (Relative Direction) commands, the printer can print any scalable LaserJet printer font at practically any angle (in 1-degree increments).

Note



Please refer to the *PCL 5 Printer Language Technical Reference Manual* for a complete description of the DI and DR commands and their options.

**Example:
Rotating Fonts at
Any Angle**

Below is a simple example showing the words “LaserJet Printer” rotated at a 30-degree angle using the Univers typeface. This effect can be done using any LaserJet printer scalable font.

LaserJet Printer

<code>ⒺcE</code>	Reset the printer.
<code>Ⓔc%1B</code>	Enter HP-GL/2 mode at the current PCL position.
<code>IN;</code>	Send the HP-GL/2 <i>initialize</i> command to set the HP-GL/2 environment to a known state. (The default HP-GL/2 environment is discussed in the <i>PCL 5 Printer Language Technical Reference Manual</i> .)
<code>SC0,100,0,100;</code>	Use the scale command (SC) to establish a user unit coordinate system so that the units that describe the picture frame area extend from 0,0 to 100,100. This command allows you to describe the picture frame in units that <i>you</i> like to work with, instead of absolute or plotter units.
<code>SP1;PA0,0;</code>	<i>Select pen</i> (SP). Establish <i>absolute plotting</i> (PA) and move the pen to absolute position 0,0.

DT~,1;	Define the ~ character as a <i>label terminator</i> . Note that the comma in the command is necessary if the 1 is included.
PU50,50;	Send the <i>pen up</i> command to move the pen to the middle of the page (50,50).
DI.87,.5;	Use the <i>absolute direction command</i> (DI) to set the text angle to 30-degrees (.87 is the cosine and .5 is the sine of a 30-degree angle).
LO12;	Set the <i>label origin</i> (LO) to position 12.
SD1,21,2,1,4,25,5, 1,6,0,7,4148;	Use the <i>standard font definition</i> command (SD) to select the ASCII, 25-point, italic, Univers font.
SS;	Send the <i>select standard font</i> command (SS) to select the font just designated with the SD command; this font will be used for subsequent labeling.
LBLaserJet Printer~;	Print the “LaserJet Printer” text using the Label (LB) command. Note that this is the last HP-GL/2 command before exiting HP-GL/2 mode; as demonstrated here, the last HP-GL/2 command must be followed by a semicolon.
^c^A	Enter PCL mode at the cursor position that was in effect when the HP-GL/2 mode was entered.
^cE	Send the <i>reset</i> command to eject a page and reset the printer to the user default environment.

Anisotropically Scaled Fonts

The PCL 5 LaserJet printers have the ability to anisotropically scale fonts, which means they can be scaled disproportionately. To put it another way, they can be scaled so that their size increases a different amount in the X direction than in the Y direction.

PCL 5 printers can anisotropically scale fonts in two ways. The *first method* is the most straightforward and utilizes the *relative character size* (SR) or absolute character size (SI) commands. Both commands scale fonts, but there is a reason for choosing one command over the other, depending on how the image will be used. The *SR* command allows the font to be further scaled in the event that the picture frame with which it is associated is enlarged or reduced. The *SI* command scales the font an absolute amount (in centimeters) and will not allow the fonts to change size along with the picture frame.

The *second method* of anisotropically scaling fonts was alluded to in the explanation of the first method: allowing the fonts to scale according to changes that are made in the size of the HP-GL/2 picture frame. When this method is used, the printer applies the picture frame scaling factor to the font. As mentioned, this is done in conjunction with the SR command. (See the *Creating a Page-Size Independent Plot* discussion earlier in this chapter.)

Figure 10-5 shows the use of the SI and SR commands for anisotropically scaling fonts. The PCL Picture Frame was defined as 4 by 4.75 inches and the SI command was used to print the text on the top half of the rectangle. Since the text on the top was scaled absolutely (using the SI1,.5 command), the character size is approximately 1 cm x .5 cm. On the other hand, since the text on the bottom was scaled relatively, in order to scale it to a comparable size, the command parameters had to be much larger (the SR9,4.5 command was used). It is interesting to note that both samples were scaled from a 10-point CG Times font.

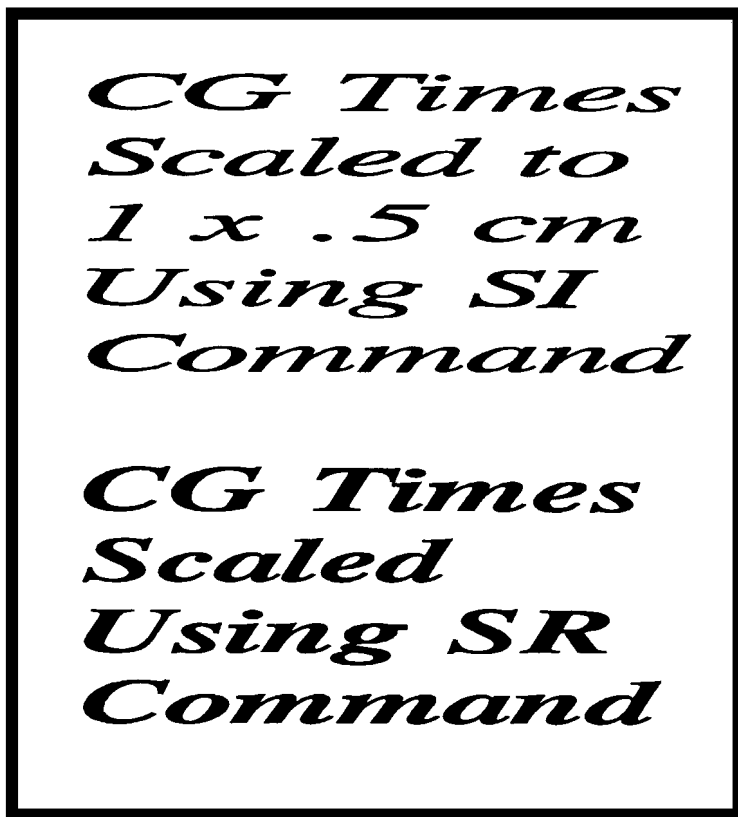


Figure 10-5. Anisotropic Scaling Using SI and SR

Figure 10-6 shows the effect that changing the PCL picture frame size has when using the SI and SR commands. The same file that was used to print Figure 10-5 was used to print the sample in Figure 10-6, but the PCL picture frame size was reduced to 2 x 2.375 inches. Notice that the text that is scaled to an absolute size gets clipped at the picture frame boundaries while the relatively scaled text adjusts to fit the new picture frame.

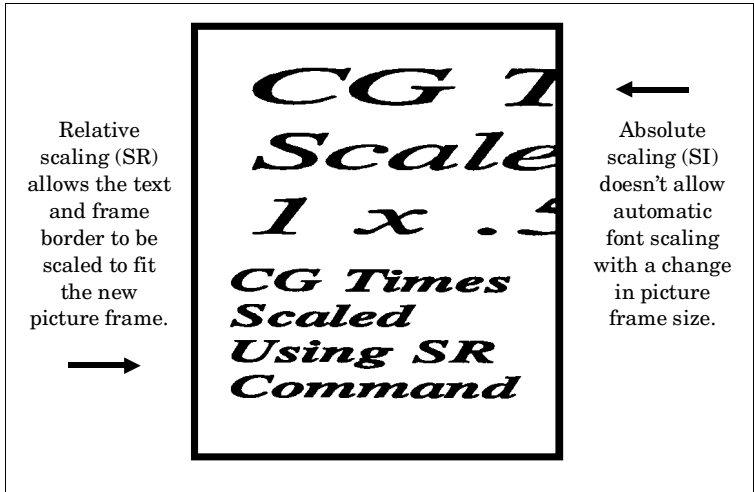


Figure 10-6. Picture Frame Size Changes and Scaling

For your further study, the source and executable code for a similar SI/SR command comparison is contained on a disk supplied with this manual. (SISRSAMP.EXE is the filename). In addition to that file, many other HP-GL/2 examples are contained on the disk. See Appendix B for a look at the other examples that are included.

Contents

The Print Model—Filling With Patterns	11-1
How the Print Model Works	11-3
Using Rectangular Area Fill	11-5
Patterning Other Images	11-7
Using the Print Model Commands	11-7
User-Defined Patterns	11-13

The Print Model — Filling Images, Rectangles, and Fonts With Patterns

The Print Model defines how images, rectangles and fonts can be filled with shading or patterns. Instead of offering your LaserJet customers only black type, your software application can offer reverse type, or type filled with patterns or shades of gray. Your software can also allow users to fill raster images with shades or patterns, or overlay images on top of each other to create special effects.

Using the Print Model, the “1” bits of raster data can be printed using a selected pattern type, including shading, HP-defined patterns, and user-defined patterns. Transparency modes allow the “0” bits of the source image or pattern data to be treated as transparent or opaque, providing flexibility in making composite images.

Note



In PCL mode, only the LaserJet IIIIP and LaserJet 4 printers allow you to fill areas with *user-defined patterns*.

In HP-GL/2 mode, the Transparency Mode (TR) command also offers Print Model capabilities. For all PCL 5 printers, you can create your own patterns for filling images using the HP-GL/2 Raster Fill Definition (RF) command.

The Print Model employs the following terms to describe its operation:

- Pattern
- Source Image
- Destination Image
- Source Transparency Mode
- Pattern Transparency Mode

Each term is described below and illustrated in Figure 11-1:

The *pattern* is the design or arrangement of pixels that is used to fill images using the Print Model. The “1” bits of the pattern represent black pixels and the “0” bits represent white pixels. The transparency modes affect how the

pattern affects the printed image. The white areas of the pattern are visible if the pattern transparency mode is set to 1 (*opaque*); the white areas are not visible if the pattern transparency mode is set to 0 (*transparent*).

The *source image* is the image that is to be filled with a shade or pattern. A source image acts as a “stencil” or “mask” whose “1” bits define the area where the *pattern* will be allowed to be visible on the page. The white areas of the source image are visible if the source transparency mode is set to 1 (*opaque*), but not if set to 0 (*transparent*).

The *destination* image is the area on the page where the pattern/source image combination will be placed. In other words, the destination image is the area of the paper that will be affected by the pattern and source image, and includes any images placed through previous operations. For example, if you placed three images on top of each other and then tried to print a shaded font on top of the images, the combination of the first three images would be considered the destination image.

The source transparency mode controls whether the white pixels of the source image are visible on the final image. When the source transparency mode is set to 0 (*transparent*), the white pixels of the source image have no effect on the destination image. When the mode is set to 1 (*opaque*), the white pixels of the source image are transferred to the destination.

The pattern transparency mode controls the effect of the pattern’s white pixels on the printed image. When the pattern transparency mode is set to 0 (*transparent*), the white pixels are not visible on the printed image. When it is set to 1 (*opaque*), the white pixels *are* visible.

Figure 11-1 illustrates Print Model operation using a gray shade as the *pattern*, a character as the *source image*, and a black background as the *destination*. Notice that when the source transparency mode is set to 1 (*opaque*), the white areas of the entire character cell are visible.

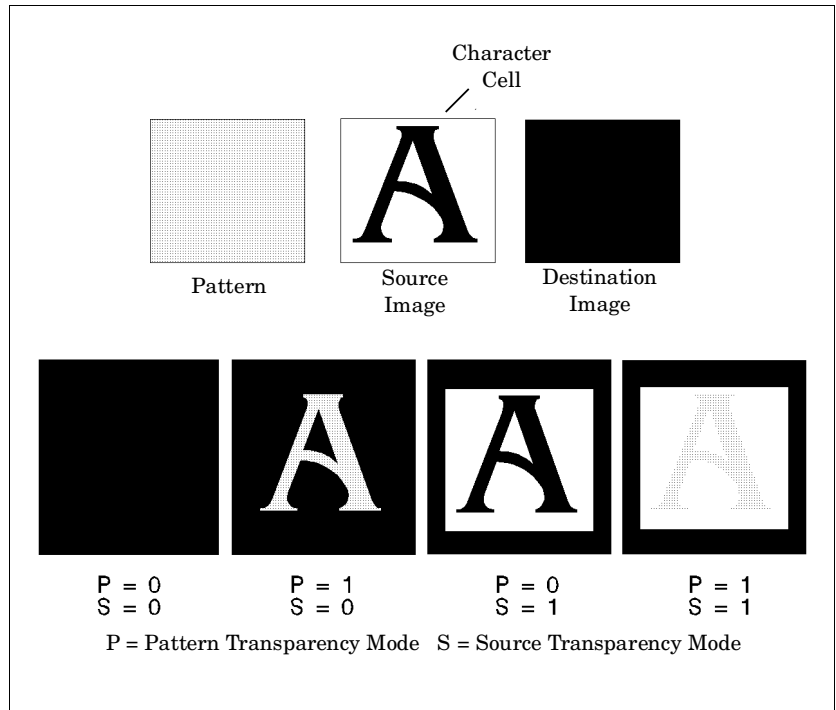


Figure 11-1. Varying the Transparency Mode Values

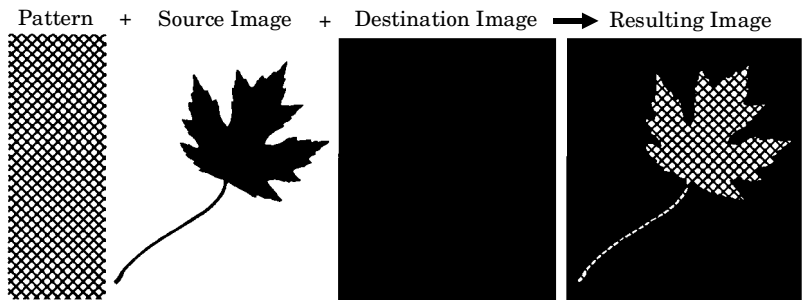
How the Print Model Works

The operation of the Print Model can be compared to painting, where the images are “painted” with white paint onto the paper, or destination. The black pixels of the source image describe the areas where the white paint may be applied to the destination. (Figure 11-1 illustrates this—when the pattern transparency mode is set to 1 (opaque), the white pixels of the pattern are visible in the letter A.)

The white areas of the source image are visible if the source transparency mode is set to “opaque”. If the source transparency mode is set to “transparent”, you will not see the white areas of the source in the printed image.

Similar to the way the source transparency mode operates, the pattern transparency mode affects whether the white portions of the pattern are visible on the final image. When the pattern transparency mode is set to 0 (transparent), the white areas of the pattern are not visible on the destination image. When it is set to 1 (opaque), the white areas of the pattern are applied directly to the destination image.

The image below uses a raster graphic image of a maple leaf as the source image. In this case, the source transparency mode is set to 0 (transparent) and the pattern transparency mode is set to 1 (opaque).



Note



To get a better idea of how the source and pattern transparency modes affect the final printed image, the *PCL 5 Printer Language Technical Reference Manual* shows images representing all the possible combinations of these modes.

The operation of the Print Model is more easily understood with the help of a few examples. To help you use the Print Model in your software applications, the examples in this section explain many of the effects you can produce with fonts and images.

Using Rectangular Area Fill

Filling rectangular areas (or rules) with shading, hatch patterns, or user-defined patterns (LaserJet IIIP/4 only) is accomplished using a few simple commands, generally in the following order:

- Move the cursor to the position that will be the upper left corner and the starting point of the rectangle.
- Set the *pattern transparency mode*, usually to opaque. (The *source transparency mode* has no effect on the rectangular area since the Print Model recognizes the rectangular source image as solid black [no white pixels].)
- Specify the size of the rectangle to be filled using the *horizontal* and *vertical rectangle size* commands.
- Fill the rectangle with the desired pattern or shade using the *fill rectangular area* command.

The following example illustrates the way to print filled rectangular areas.

Example: Rectangular Area Fill and the Pattern Transparency Mode

This example shows a way of filling rectangular areas (rules) with patterns. The example also demonstrates the effect of the *pattern transparency mode* on the resulting image. The sample on the top was produced in exactly the same way as the one on the bottom, with the exception of one command; the one on the top was printed with the *pattern transparency mode* set to opaque (1) while the one on the bottom was printed with the *pattern transparency mode* set to transparent (0).



ⒺE	Reset the printer
Ⓔ*p300x300Y	Move the starting cursor position (CAP) one inch to the right of the left-most printable position and one inch below the default top margin (which is one-half inch below the top of the physical page).
Ⓔ*c600a150b0P	Set the <i>horizontal rectangle size</i> to 600 PCL Units, and the <i>vertical rectangle size</i> to 150 PCL Units. Fill the rectangular area with solid black (Ⓔ*c0P).
Ⓔ*p-50x+50Y	Position the cursor to print the pattern-filled rectangle by moving the cursor position 50 PCL Units to the left and 50 PCL Units beneath the current cursor position. (The cursor position before this command is at the top left corner of the black rule.)
Ⓔ*v10	Set the <i>pattern transparency mode</i> to opaque. This is done so that the white areas of the cross-hatch pattern are visible on top of the black rule that was just printed (the destination). In the sample on the bottom, this command was set to transparent (the default).
Ⓔ*c4G	Set the <i>pattern ID</i> number to pattern number 4.
Ⓔ*c700a50B	Set the <i>horizontal rectangle size</i> to 700 PCL Units, and the vertical size to 50 PCL Units.
Ⓔ*c3P	Fill the rectangular area with an HP-defined pattern.

Patterning Other Images

The LaserJet printer's capability to fill images with shades and patterns provides an easy way to print special type and graphic effects. There are extensive possibilities for variation including, but not limited to:

- Gray-shaded images on white, black, shaded, or patterned backgrounds
- Shaded, patterned or white type on black, shaded or white backgrounds
- Combinations of shaded images on top of each other
- Type with shaded portions or type filled with patterns

Using the Print Model Commands

In general, the sequence of commands when using the Print Model is as follows:

- Move the cursor to the desired starting position (keep in mind that the cursor position is different for text compared to graphics).
- Set the pattern transparency ($\text{E}_c*\text{v}\#\text{O}$) and source transparency modes ($\text{E}_c*\text{v}\#\text{N}$) to the desired settings.
- Select a gray-shading percentage or choose an HP-defined pattern ($\text{E}_c*\text{c}\#\text{G}$).
- Select a pattern type ($\text{E}_c*\text{v}\#\text{T}$) or use the current pattern (solid black if no pattern has been specified). (Since the transparency mode and pattern type command have the same first three characters (E_c*v), the commands can be linked together if desired [for example, $\text{E}_c*\text{v}10\text{0}\text{n}2\text{V}$]).
- Send the source image (raster, text, or rules) to the printer.
- Reset the pattern ID to black so any text or images that follow are printed black unless otherwise specified.

The following examples demonstrate the use of the Print Model to fill raster images and fonts with shades and patterns. Use these examples as a guideline for programming your software applications.

Example: Pattern-Filled Raster Graphics

This example uses a raster image of a mountain range to demonstrate filling an image with patterns. The image on the left is the original raster image, the one in the middle was created using the commands listed below, and the image on the right was created the same way using a cross-hatch pattern instead of a shading pattern.



`Ec*p450x450Y`

Move the cursor 450 dots to the right of the left-most printable spot on the page and 450 dots below the top margin.

`Ec*v0n1O`

Set the *source transparency mode* to transparent and the *pattern transparency mode* to opaque. (In this example, this command is not necessary because the background [destination] is blank; it is included for instructional purposes because in actual applications it is usually desired if the image overlaps something else.)

`Ec*c30G`

Set the pattern area fill ID to 30 (for 30% gray).

`Ec*v2T`

Select the current pattern to an HP-defined shading pattern.

`Ec*b#Wdata...`

Send the raster data to print the image.

`Ec*v0T`

Set the current pattern type to solid black. This command is sent so that all succeeding images and text will print black instead of pattern-filled.

Example: Pattern-Filled Raster Graphics on a Black Destination

This example prints a pattern-filled raster image on top of a filled rectangular area (rule). The image was printed using the following commands:



`Esc*p0x0Y`

Position the cursor at position 0,0.

`Esc*c200a200b0P`

Draw a 200-dot by 200-dot rectangular area with solid black fill.

`Esc*p65x45Y`

Move the cursor for placing the graphic image (65 dots to the right and 45 dots down from the 0,0 position).

`Esc*v0n1O`

Set the *source transparency mode* to transparent; this is necessary to keep the white areas surrounding the raster image from showing on the destination image. Set the *pattern transparency mode* to opaque; this is necessary to allow white areas of the pattern to show against the black background.

`Esc*c3G`

Set the pattern area fill ID to 3. This command could indicate either a 3% gray shade pattern or the HP-defined cross-hatch pattern number 3; the following command notifies the printer that the cross-hatch pattern has been selected instead of a gray-shade pattern.

`Esc*v3T`

Set the current pattern type to an HP-defined cross-hatch pattern.

`Esc*r1AEsc*b9Wdata...`

Send the raster data.

`Ec*v0T`

Reset the pattern ID to print black so that subsequent printing will be black instead of pattern-filled.

**Example:
Reverse and
Pattern-Filled Type**

The following example demonstrates printing reverse and pattern-filled type on top of a filled rectangular area.



`Ec*p300x300Y`

Move the cursor one inch (300 dots) to the right of the left-most printable position and one inch down from the top margin.

`Ec*c756a225b0P`

Specify a rectangular area (rule) that is 756 dots wide by 225 dots high and fill the rule with solid black.

`Ec*v0n1O`

Set the source transparency mode to transparent and the pattern transparency mode to opaque.

`Ec*v1T`

Set the pattern type to solid white for printing the word “Soft”.

`Ec(0U`

Select the ASCII symbol set.

`Ec(s1p40v1s3b4101T`

Select a bold, italic, 40-point CG Times font. The printer automatically scales the resident typeface to 40 points. (A downloaded bitmapped font would also work in this example, but it would have to be created and downloaded to the printer before selecting it.)

<code>Ec*p+160y+90X</code>	Move the cursor position 160 dots down and 90 dots to the right of the current position (which is the upper left corner of the black rectangle). This relative cursor move centers the word within the black rectangle.
“Soft”	Send the first four letters of text.
<code>Ec*c3G</code>	Set the pattern area fill ID to 3.
<code>Ec*v3T</code>	Set the current pattern type to 3 (HP-defined cross-hatch pattern).
“ware”	Send the last four letters of text.
<code>Ec*v0T</code>	Set the pattern type back to black to “turn off” the printing of pattern-filled images.

**Example:
Pattern-Filled Type**

This example prints pattern-filled type:



<code>Ec*p300x300Y</code>	Move the cursor to desired position.
<code>Ec*v0n1O</code>	Set the <i>source transparency mode</i> to transparent and the <i>pattern transparency mode</i> to opaque.
<code>Ec*c6G</code>	Set the pattern ID to pattern 6.
<code>Ec*v3T</code>	Set the pattern type to 3 (HP-defined cross-hatch pattern).
<code>Ec(0U</code>	Select the ASCII symbol set
<code>Ec(s1p40v1s3b4101T</code>	Select a bold, italic, 40-point CG Times font.
“LaserJet”	Send text for the word “LaserJet”.
<code>Ec*v0T</code>	Set the pattern type to solid black.
“LaserJet”	Send text for the second “LaserJet”.

Note



If the *source transparency mode* is set to opaque, white areas of the character cells that overlap each other erase part of the adjacent characters. The *pattern transparency mode* is set to opaque so that white areas of the characters print correctly if they happen to overlap any non-white areas.

Example: Drop-Shadow Effects Using the Print Model

This example builds on the previous example. In this case, the black “LaserJet” is printed first and then the pattern-filled word is moved so that it overlaps the first word with a slight offset to produce a drop-shadow effect.

`Ec*p300x300Y`

Move the cursor to desired position.

`Ec(0U`

Select the ASCII symbol set

`Ec(s1p40v1s3b4101T`

Select the bold, italic, 40-point CG Times font.

“LaserJet”

Send text for the black word that is used for the drop shadow.

`Ec*p-502x-7Y`

Using a relative cursor move command, move the cursor 502 dots to the left and 7 dots up. This places the baseline of the next word 7 dots higher and slightly to the left of the first word. (In a software application, the cursor position could be *pushed* before printing the first word (or phrase) and then *popped* after printing; then the cursor could be positioned according to the degree of offset requested by the user. For example, the commands
`Ec&f0SLaserJetEc&f1SEc*p-7x-7Y`
would offset the cursor by 7 dots above and to the left of where the

	first word was started—without requiring any calculating.)
$\text{E}_c^*\text{v}\text{0n}1\text{O}$	Set the source transparency mode to transparent and the pattern transparency mode to opaque.
$\text{E}_c^*\text{c}6\text{G}$	Set the pattern ID to 6.
$\text{E}_c^*\text{v}3\text{T}$	Set the pattern type to 3 (HP-defined cross-hatch pattern).
LaserJet	Send text for the cross-hatched word.
$\text{E}_c^*\text{v}\text{0T}$	Reset the pattern type to solid black.

User-Defined Patterns

The HP LaserJet IIP and LaserJet 4 printers allow you to fill type and images with user-defined patterns. These patterns consist of a binary raster image which is downloaded to the printer using the *user-defined pattern* command. Once downloaded, the pattern can be used to fill images—similar to how the HP-defined patterns are used.

The basic scenario for using user-defined patterns is:

- Assign an ID number to the pattern to be downloaded ($\text{E}_c^*\text{c}\#\text{G}$)
- Set the pattern reference point to the desired position ($\text{E}_c^*\text{p}\#\text{R}$)
- Specify the pattern to be permanent or temporary ($\text{E}_c^*\text{c}\#\text{Q}$)
- Download the user-defined pattern using the $\text{E}_c^*\text{c}\#\text{W}[\textit{header bytes}] [\textit{pattern data}]$ command.

Once the pattern is downloaded, it can be used the same way as one of the HP-defined patterns. The commands listed above are discussed in detail in the *PCL 5 Comparison Guide*. The example on the following page shows how the user-defined patterns can be used.

**Example:
User-Defined
Patterns
(LaserJet III/4 only)**

This example shows the creation and use of a star-shaped user-defined pattern. User-defined patterns in the PCL mode are only available on the LaserJet III/4 and LaserJet 4 printers, however the other PCL 5 printers have this capability using the HP-GL/2 RF (Raster Fill Definition) command.



The raster data for the star pattern is shown below in binary and hexadecimal format:

1)	00000000	00010000	00000000	(00	10	00)
2)	00000000	00010000	00000000	(00	10	00)
3)	00000000	00111000	00000000	(00	38	00)
4)	00000000	00111000	00000000	(00	38	00)
5)	00000000	01111100	00000000	(00	7C	00)
6)	00000000	01111100	00000000	(00	7C	00)
7)	00000000	11111110	00000000	(00	FE	00)
8)	00000000	11111110	00000000	(00	FE	00)
9)	11111111	11111111	11111110	(FF	FF	FE)
10)	00111111	11111111	11111000	(3F	FF	F8)
11)	00011111	11111111	11110000	(1F	FF	F0)
12)	00000111	11111111	11000000	(07	FF	C0)
13)	00000011	11111111	10000000	(03	FF	80)
14)	00000111	11111111	11000000	(07	FF	C0)
15)	00001111	11111111	11100000	(0F	FF	E0)
16)	00011111	11000111	11110000	(1F	C7	F0)
17)	00011111	00000001	11110000	(1F	01	F0)
18)	00111100	00000000	01111000	(3C	00	78)
19)	00111000	00000000	00111000	(38	00	0C)
20)	01100000	00000000	00001100	(60	00	0C)
21)	10000000	00000000	00000010	(80	00	02)

The commands on the following page were used to print the “Stars!” image above.

<code>Ec*p300x600Y</code>	Move cursor position to the point desired as the starting position.
<code>Ec*c5G</code>	Assign a pattern ID number of 5 to the pattern to be created.
<code>Ec*p0R</code>	Set the pattern reference point to the current cursor position and enable rotation of the pattern with the print direction.
<code>Ec*c4Q</code>	Make the pattern temporary.
<code>Ec*c71W[header bytes] [pattern data]</code>	Send the <i>user-defined pattern</i> command. Specify 71 bytes for the pattern, 8 <i>header</i> bytes (00 00 01 00 00 15 00 18—Hex) plus 63 <i>pattern data</i> bytes. (This command, including the header bytes, is discussed in detail in the <i>PCL 5 Printer Language Technical Reference Manual</i> .)
<code>Ec*v0n1O</code>	Set the source transparency mode to transparent and the pattern transparency mode to opaque.
<code>Ec*c5G</code>	Set the pattern ID to pattern number 5 (the pattern just defined).
<code>Ec*v4T</code>	Set the pattern type to 4 (user-defined).
<code>Ec(0U Ec(s1p72v0s3b4101T</code>	Select a 72-point CG Times font.
Stars!	Type the word “Stars!” to be filled with the user-defined pattern.
<code>Ec*v0T</code>	Set the pattern type to black so that any succeeding text or images won’t be filled with the pattern.

Contents

Introduction	12-1
Suggestions For Using Macros	12-1
Macro Cartridges and Macro SIMMs	12-2
Creating Macros	12-2
Using Macros	12-2
General Macro Management	12-4
HP-GL/2 in the Macro Environment	12-6
Summary of Rules Concerning Macros	12-6

Introduction

Macros provide a means of consolidating a large amount of printer commands and print data into one command, so that they can be downloaded to the printer once and then used a number of times. Macros can significantly increase performance and decrease memory requirements, especially if a particular graphic image or form is used repetitively within an application. This chapter covers some of the guidelines for using macros and includes examples to demonstrate their use.

Suggestions For Using Macros

Since macros can be downloaded once and then used repeatedly, they are useful when repeating a large group of commands and data more than once, or a small group of commands and data several times. For example, if you create a small logo for printing on every page of a report, using a macro overlay to print the logo is an ideal solution. You would download the macro once and enable automatic overlay, causing the logo to automatically print on every page of the document. Likewise, if you create a large graphic image that will be printed more than once, using a macro saves considerable download time and printer memory, even if it is only printed twice.

Note



The LaserJet 4 printer is the only PCL 5 printer that allows, HP-GL/2 commands to be executed from within macros.

A particularly good use of macros is for printing forms for businesses that use the same forms day after day. The forms can be downloaded as macros that can be quickly printed and simultaneously filled in with data. Complex forms that may take awhile to download the first time can be printed much quicker after they are downloaded as a macro.

Macro Cartridges and Macro SIMMs

The PCL 5 printers support macro cartridges that plug into the printer's standard font cartridge slots. The LaserJet 4 printer also supports SIMM-based macros. For more information about macro cartridges and SIMMs, contact the HP Boise Division Product Specials Group at (208) 323-3684.

Creating Macros

To create a macro, do the following:

- Designate a macro ID number.
- Send the start macro definition command ($\text{E}_c\&f\text{0X}$).
- Send the escape sequences, control codes and data that comprise what you want to print with the macro.
- Send the stop macro definition command ($\text{E}_c\&f\text{1X}$).

Using Macros

Once a macro has been created, it can either be called, executed, or overlaid. The difference between the macro call and macro execute commands is how the print environment is affected following the macro implementation.

If a macro is called, it operates using the current modified print environment. Changes made to feature settings during a macro call are recorded in the modified print environment; however, these changes are not retained upon completion of the macro call. After the macro call, the modified print environment that existed prior to the macro call is restored; however, the cursor position is not stored. To avoid losing the cursor position, it can be saved using the push/pop cursor position command as demonstrated in the next example.

If a macro is executed, it operates using the current modified print environment. Changes made to feature settings during macro execution are recorded in the modified print environment; these changes are retained upon completion of the macro execution.

If a macro is *overlaid*, the automatic overlay is the final operation each time a page is printed. Before the macro is executed, the current modified print environment is saved and replaced by the overlay environment. Upon completion of the macro overlay, the modified print environment that existed prior to the macro overlay is restored.

Example: Storing the Cursor Position Before Macro Use

This example demonstrates the use of the *push/pop cursor* command to store the cursor location following a macro call. The cursor position is stored prior to calling the macro and is then restored following the macro call.

<code>ⒺcE</code>	Reset the printer.
<code>Ⓔc&11X</code>	Set the number of copies to 1.
<code>Ⓔc&1⓪O</code>	Select portrait orientation.
<code>Ⓔc&f25Y</code>	Specify a macro ID number of 25.
<code>Ⓔc&f⓪X</code>	Start the macro definition.
PRINT DATA	Send the contents of the macro.
<code>Ⓔc&f1x1⓪X</code>	Stop the macro definition and make the macro permanent.
PRINT DATA	Print the data that precedes the spot in the document where the macro will be placed.
<code>Ⓔc&f⓪S</code>	Push (store) the current cursor position before calling the macro.
<code>Ⓔc&f25x3X</code>	Call macro number 25. This command causes the macro action to occur.
<code>Ⓔc&f1S</code>	Pop (recall) the cursor position. This command moves the cursor to the position it was in prior to calling the macro. (The macro could have caused the cursor to move.)
PRINT DATA	Print the remaining part of the job.
<code>ⒺcE</code>	Printer reset at end of print job.

General Macro Management

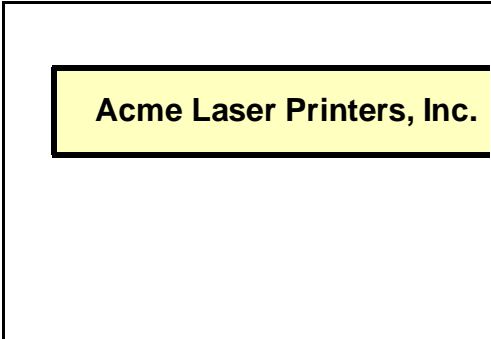
Macros are managed in much the same way as fonts are managed. Using the *macro control* command as indicated in the *PCL 5 Printer Language Technical Reference Manual*, macros can be assigned an ID number, made permanent or temporary, and deleted.

The decision to make a macro permanent or temporary depends on the number of users using the printer and what their applications are. A macro that will be used many times throughout a print job but would never be used by other jobs would best be created as a temporary macro. Since temporary macros are erased with a printer reset, the next print job sending a reset would clear the macro from memory, freeing space for other data.

Conversely, macros that will be used by many users or by many jobs should be created as permanent macros so that they can be easily accessed without downloading them repeatedly.

Example: Automatic Forms Overlay

This example automatically overlays a form on every page of a document and demonstrates good macro management techniques. This simple form consists of a shaded rule with a black border and a fictitious company name. The black border around the shaded area was easily made using the Print Model by specifying a 1-inch black rule with a slightly smaller 10% gray rule on top of it, placed so that the black rule creates a border.



Acme Laser Printers, Inc.

(In this example, it is assumed that the proper job setup and page setup commands precede the commands shown.)

<code>Ec&f37y0X</code>	Start the definition of a macro with an ID number of 37.
<code>Ec*p75x75Y</code>	Designate the starting cursor position.
<code>Ec*c2240a300b0P</code>	Specify a black rule that is 2240 PCL Units wide by 300 PCL Units high (7.46 by 1 inch).
<code>Ec*p+10x+10Y</code>	Move the cursor 10 PCL Units to the right and 10 PCL Units down from the cursor position where the black rule was started.
<code>Ec*v10</code>	Set the pattern transparency mode to opaque so the white areas of the following rule are visible on top of the black rule.
<code>Ec*c10G</code>	Specify a 10% shading pattern for the following rule.
<code>Ec*c2220a280b2P</code>	Print a shaded rule that is 20 PCL Units narrower and shorter than the black rule.
<code>Ec*p150x200Y</code>	Move the cursor position for the beginning of text.
<code>Ec(8U</code>	Select the Roman-8 symbol set.
<code>Ec(s1p18v3b4148T</code>	Select an 18-point bold Univers font.
Acme Laser Printers, Inc.	Send text for the logo.
<code>Ec&f1X</code>	Stop Macro definition.
<code>Ec&f37y10x4X</code>	Make the last-specified macro ID number (# 37) permanent and enable it for automatic overlay.

PRINT TEXT	Position the cursor and send text to be printed on the page.
CR-FF	Send a carriage return and form feed to eject the current page.
Ⓔ&f37y8X	At the end of the print job, delete macro number 37 to free associated memory.

HP-GL/2 in the Macro Environment

For all printers except the LaserJet 4, HP-GL/2 is not supported in the macro environment. The command to enter the HP-GL/2 mode ($\text{Ⓔ}\% \# \text{B}$) is ignored by the other PCL 5 printers when used within a macro.

Summary of Rules Concerning Macros

Here is a summary of the rules concerning macros:

- When using downloaded and cartridge macros, the printer uses the following priority system: downloaded macros take priority over the left cartridge macros, which take priority over the right cartridge macros.
- Up to 32,767 macros can be downloaded to the PCL 5 LaserJet printers, which means the number is usually only limited by the amount of available printer memory.
- Other than the *call* or *execute* commands, no macro control operations may occur within a macro.
- A printer reset is not allowed in a macro.
- Unlike the LaserJet series II printer, font management commands are allowed in macros; that is, fonts may be downloaded, deleted, or made permanent in a macro.
- Macro execution can be nested two deep (i.e., a macro may execute a macro that executes another macro).

- The macro enabled for auto macro overlay is executed on each page until the macro is disabled or deleted, a reset occurs (E_cE or control panel), or the page length, page size or orientation is changed.
- HP-GL/2 commands may only be used within macros that will be sent to the LaserJet 4 printer.
- Display functions mode is not allowed in a macro.

Contents

Introduction	13-1
General Tips	13-1
Combining Escape Sequences	13-1
Job Setup	13-2
For Non-PJL LaserJet Printers	13-2
For PJL LaserJet Printers	13-2
Page Setup	13-4
General Print Job Initialization	13-5
Non-PJL Example	13-7
LaserJet IIISi Example	13-7
LaserJet 4 Example	13-8
Using Fonts	13-8
Font Support	13-9
Raster Graphics	13-9
The Print Model	13-10
Vector Graphics	13-11
Macros	13-12

Introduction

As with most programming languages, the PCL printer language sometimes offers more than one way to solve a particular programming problem. Some of these methods are more efficient than others, and this chapter focuses on how to provide the most efficient and effective way to approach the use of specific features.

General Tips

Combining Escape Sequences

To reduce the size of print files and help speed the printing process, escape sequences can be combined. If two or more escape sequences have the first three characters in common (including the escape character), they can be combined into one escape sequence. For example, the following two cursor positioning commands, `^c*p300X^c*p330Y`, may be combined into one command, `^c*p300x330Y`.

Although combining escape sequences only appears to save a few characters, the difference may be tremendous when considering an entire print job. For more information about combining escape sequences, please refer to the *Parameterized Escape Sequences* discussion in the *PCL 5 Printer Language Technical Reference Manual*.

Job Setup

Job setup involves sending commands to create the desired state for your application. Job setup is different for non-PJL printers than it is for PJL printers. Job setup for both non-PJL and PJL printers is described below:

For Non-PJL LaserJet Printers

Sending the `␣E` (reset) command as the first command of your job sets the printer to the user default environment, providing a starting place from which to begin creating a desired state.

Once the printer is reset, your software must consider how the control panel settings may be configured. You can override the control panel settings as long as your software provides a way to set the same features (so that the customer doesn't get "locked out" from a control panel feature). (See Chapter 2 for more information about job setup.)

Once the printer is reset and commands are sent to override the control panel, commands can be sent to set the printer to the desired state. Remember, job setup commands need only be sent at the beginning of the print job, with one exception. Sending the reset command at the *end* of each print job "cleans up" the printer for the succeeding job. (For a good example of a job setup string, see the *General Print Job Initialization* discussion later in this chapter.)

For PJL LaserJet Printers

For the LaserJet IIISi and LaserJet 4 printers, the job setup procedure is basically the same as for non-PJL printers, except the initial `␣E` reset must be preceded by a UEL command (`␣%-12345X`) and a `PJL ENTER LANGUAGE` command. The UEL command causes the printer to exit the current printer language and to enter PJL mode. The `PJL ENTER LANGUAGE` command explicitly selects a page description language. In addition to placing these two commands prior to the initial `␣E`, the `␣E` at the end of the job should also be followed by the UEL command.

For example, the PCL commands and print data should be encapsulated by PJP commands as follows:

```
ESC%-12345X@PJP <CR><LF>  
@PJP ENTER LANGUAGE=PCL <CR><LF>  
ESC<PCL commands and data>ESCESC%-12345X
```

Note



Notice that there are no spaces after the X in the initial ESC%-12345X command, and that there is a line feed immediately preceding the ESC. The line feed characters are required to terminate each PJP command.

LaserJet 4 Considerations

The LaserJet 4 printer allows you to use the PJP SET command to set resolution enhancement (RET), page protection, and resolution to the desired values. The following information should help you decide how to set each of these three features:

- **Resolution Enhancement**—in most cases, RET improves the quality of the printed image, so it should be turned on unless there is a specific reason to turn it off.
- **Resolution**—the resolution can be set at either 300 or 600 dots per inch. Rendering fonts requires less internal memory when the device resolution is 300 dpi than at 600 dpi. At the beginning of the job, resolution should always be set to the resolution of any bitmapped graphics that are part of the print job. For example, bitmapped images generated at 300 dpi should be printed with device resolution set to 300 dpi. *Note that when the resolution setting is changed, all downloaded fonts and macros are erased from printer memory.*
- **Page Protection**—page protection should not be turned on unless it is needed, since it requires a large amount of printer memory. It is, however, very useful for printing complex pages and those pages containing vector graphics. Applications can take advantage of the

LaserJet 4 printer's status readback capability, only turning page protection on if unsolicited status indicates a print overrun error (error 21) has occurred. To accomplish this, use the @PJM USTATUS DEVICE = ON command and check for a print overrun error (status error code 30017). If print overrun occurs, resend the job with page protection turned on so that the job prints correctly the second time. *As with changing the resolution, changing page protection status erases all downloaded fonts and macros from printer memory.*

For the LaserJet 4 printer, other PJL commands in addition to the ones just mentioned may be used for such actions as modifying the control panel display and setting printer defaults. For more detailed PJL information for the LaserJet 4 printer, see the *PJL Technical Reference Manual*.

Page Setup

Although it is possible to send page setup commands for each page of data, it is not efficient to do so. After the first page has been sent to the printer, send page setup commands only when a change is necessary from the previous page. For example, if a top margin is sent before the first page of data, resending the top margin command isn't necessary on succeeding pages unless the user wants a different top margin distance than the initial setting.

General Print Job Initialization

Chapters 2 and 3 discuss in detail the proper job and page setup commands. For quick reference, the following general setup sequence is included here. Commands that are only used for PJJ printers are noted as such. As indicated below, you may not be sending all of the commands at the beginning of the print job because you don't want to lock out features from your users. However, the commands you do use at the beginning of your print job should be sent in the order listed below:

<code>εc%-12345X@PJJ <CR><LF></code>	(PJJ PRINTERS ONLY) Send the UEL command, followed immediately by @PJJ<CR><LF> to exit the current printer language and enter PJJ mode. (The “@PJJ” prefix must always immediately follow the UEL command, with no spaces in between the X and the @. The “@PJJ” may be followed by a <CR><LF> (or just <LF>) or it may be the prefix for the next PJJ command, such as: @PJJ ENTER LANGUAGE = PCL<CR><LF>.)
<code>@PJJ SET RESOLUTION = # <CR><LF></code>	(LASERJET 4 ONLY) Set the resolution to match the resolution of any bitmapped graphics in the print job.
<code>@PJJ SET PAGEPROTECT = # <CR><LF></code>	(LASERJET 4 ONLY) Set page protect off unless it is needed to process a complex job.
<code>@PJJ SET RET = # <CR><LF></code>	(LASERJET 4 ONLY) Leave resolution enhancement on unless there is a good reason to turn it off (such as the rare case where an application prints better with RET off).
<code>@PJJ ENTER LANGUAGE = # <CR><LF></code>	(PJJ PRINTERS ONLY) Use the ENTER command to explicitly select a printer language.

ⒺE	Reset the printer. For non-PJL printers, this is the first command in the print job. For PJL printers, this command should immediately follow the <LF> at the end of the PJL ENTER command.
Ⓔ&l#X	Set the number of copies. This feature overrides the COPIES= feature on the control panel, so only send this command if your application allows the user to specify the number of copies.
Ⓔ&l#S	Select simplex or duplex printing. This feature overrides the DUPLEX= feature on the control panel, so only send this command if your application allows the user to select duplex printing. This feature is not ignored on non-duplexing machines, and may result in a conditional page eject.
Ⓔ&l#H	Select a paper source. This command overrides the TRAY= and MANUAL FEED= features on the control panel; only send this command if your application allows the user to select a different paper source or manual feed.
Ⓔ&l#A	Specify the page size. This command overrides the PAPER= and ENVELOPE= feature on the control panel
Ⓔ&l#O	Specify the logical page orientation. This command overrides the ORIENTATION= feature on the control panel; only send this command if your application allows the user to select an orientation.
Ⓔ&l#C	Designate the line spacing value (VMI). This command overrides the FORM= feature on the control panel
Ⓔ&l#E	Set the top margin if you desire a different value than the default.
Ⓔ&l#F	Set the text length to the desired number of lines (based on the previously specified VMI command) if you want a text length other than the default.
Ⓔ&a#L	Set the left margin if the default is not desired.
Ⓔ(ID	Specify a primary symbol set. This feature overrides the control panel's SYM SET= feature
Ⓔ(s#p#h#v#s#b#T	Specify a primary font. This feature overrides the control panel's FONT NUMBER= feature; only set this feature if your application allows font selection.

Non-PJL Example

A sample non-PJL initialization string shown in the same order as the commands on the previous page is:

```
␣␣␣␣&l1x1s1h2a0o8c6e54F␣␣&a5L␣␣(␣U␣␣(s1p9vs3b4101T
```

This set of commands would reset the printer, specify 1 copy, specify simplex (1-sided) printing, specify the paper tray as a paper source, choose letter-size paper, select portrait orientation, VMI=8 (6LPI), set top margin to one inch (6 lines), select a 9-inch text length (54 lines), a 5-column left margin, ASCII symbol set, and a proportional, 9-point, upright, bold, CG Times font.

After receiving the above-listed commands, the printer would be ready to receive PCL print data. After the print data, the application would send the ␣␣E command to complete the job.

LaserJet IIISi Example

For the LaserJet IIISi printer, the same job as the non-PJL example above would be sent as follows:

```
␣␣%-12345X@PJL <CR><LF>
@PJL ENTER LANGUAGE = PCL <CR><LF>
␣␣␣␣&l1x1s1h2a0o8c6e54F␣␣&a5L␣␣(␣U␣␣(s1p9vs3b4101T
```

The commands listed above would set up the LaserJet IIISi printer to accept PCL print data. After the PCL print data, the following commands would be used to complete the job:

```
␣␣␣␣␣␣%-12345X
```

LaserJet 4 Example

For the LaserJet 4 printer, the same job would be sent as shown below. The example assumes that any graphics in the job are 600 dpi, there is no need for page protection, and that the user desires the RET setting to be medium.

```
Ⓔ% -12345X@PJL <CR><LF>  
@PJL SET RESOLUTION = 600 <CR><LF>  
@PJL PAGEPROTECT = OFF <CR><LF>  
@PJL RET = MEDIUM <CR><LF>  
@PJL ENTER LANGUAGE = PCL <CR><LF>  
ⒺⒺⒺ&l1x1s1h2a0o8c6e54FⒺⒺ&a5LⒺⒺ(0UⒺⒺ(s1p9vs3b4101T
```

The commands above would initialize a PCL job that is destined for the LaserJet 4 printer. After the PCL print data, the following commands would be used to complete the job:

```
ⒺⒺⒺⒺ% -12345X
```

Using Fonts

For documents in which two fonts are used frequently, an efficient method exists that allows you to easily switch between the two fonts. Specifying one font as a primary font and the other font as a secondary font allows you to alternate between the fonts using the Shift In/Shift Out control codes. (See the portion of Chapter 5 entitled *Primary and Secondary Fonts*.) Using this method is much more efficient than frequently transmitting the entire font select escape sequence.

When downloading fonts, those with a large point size take more time to download than smaller fonts. For situations such as printing headlines, an effective time-saver is to download only those characters that will be in the headline, instead of the whole symbol set.

Font Support

As explained in Chapter 6, TFM integration allows your application to be automatically compatible with all new HP font products. All HP font customers receive AutoFont Support files with each font product they purchase. This disk contains TFM files for all the fonts they have purchased. Integrating AutoFont Support ensures that your customers get instant font compatibility, requiring you to only write one driver for each printer instead of one for each font product. AutoFont Support provides all the font metric data that your software needs to format a page and saves you time and resources that would be spent writing drivers for each font product. (See Chapter 6 for more information about TFM support.)

To increase performance when reading TFM files, read only those tags that are needed for your application.

Raster Graphics

Below are some general tips that should improve raster graphics performance:

- Use one or more of the raster compression modes to reduce the size of the print file and to reduce I/O transfer time.
- Use the Y offset command to help compress raster data even further.
- Do not use PCL positioning commands (other than the Y offset) while in the raster graphics mode—they will have the effect of an *end raster graphics* command.
- Use the raster height and raster width commands to reduce the size of the resulting print file and the I/O transfer time, especially when using the Print Model.
- For horizontal and vertical lines, use rectangular area fill (rules) instead of raster graphics. Rules print more

efficiently and occupy less memory than the equivalent raster graphics image.

- With the use of adaptive compression (LaserJet IIIP/4 printers only), you can minimize the space required for storing graphics and possibly reduce memory requirements.
- Incorporating FASST is an easy way to support more efficient raster graphics (see Appendix E).

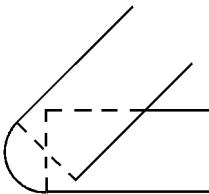
The Print Model

Borders around a page or a frame can be created by overlaying a white rule centered over a slightly larger black rule, with the difference between rule sizes creating the width of the border (the *pattern transparency mode* is set to opaque). Creating a border this way saves several bytes of data compared to sending four separate rule commands and several cursor positioning commands. (See the example in Chapter 12 entitled *Automatic Forms Overlay* for a demonstration of this technique.)

Vector Graphics

Printing performance can vary greatly depending on several factors:

- Contrary to what one might expect, there is no performance penalty involved in switching between the PCL printer language and HP-GL/2 modes. This switching can be performed over 600 times on a single page without causing the printer to slow from eight pages per minute.
- Horizontal and vertical lines are more efficient than any other lines since the printer converts them to rules for printing; rules print much more efficiently than the equivalent image performed with raster graphics.
- The complexity of the image. The more complex the image, the slower the performance.
- Use of the HP-GL/2 graphics commands. Making the most efficient use of graphics commands increases performance; for example, using the edge rectangle commands (ER and EA) allows you to draw a rectangle using only one coordinate pair instead of four.
- Use of the PE command. Use of the polyline encoded (PE) instruction to send coordinates can increase I/O performance significantly.
- Line joins and ends. There are several types of HP-GL/2 line joins and line ends and they vary in their efficiency. In general, the following join types can be paired with all line ends without performance penalties: mitered, mitered/beveled, triangular, and beveled. Of all the join types, a round join is the most efficient, while the triangular join is the least efficient to process. The most efficient join/end combination is the round join/butt end, while the least efficient coupling for the round join is the round join/triangular end combination. See Chapter 22 of the *PCL 5 Printer Language Technical Reference Manual* for an illustration of all the types of line joins and ends, and for the values used to select each type.



The Round Join/Butt End
is the Most Efficient
Join/End Combination

Note



Font manipulations within HP-GL/2 and complex HP-GL/2 vector graphics may cause print overruns (Error 21). Error 21 occurs when the printer cannot process data at the rate required to keep pace with the physical speed of the page as it moves through the printer. Before PCL 5, the PCL printer language provided no remedy for an Error 21 condition other than for the user to reduce the complexity of the desired page.

Predicting just how complex a particular HP-GL/2 graphic can be without causing an Error 21 is not an exact science. Now, however, if the error occurs the user has a viable solution. By enabling the printer's *page protection* feature from the printer's front panel, or in the case of the LaserJet 4 printer, using a PJI command, the entire page will format without the potential for error 21. (The LaserJet 4 printer allows page protection at 300 dpi without additional memory. Page protection at 600 dpi requires 4 Mb. All other PCL 5 printers require at least 1 Mb additional memory to enable page protection.)

Macros

By their very nature, macros provide for efficient programming because they provide access to a number of commands using only one command (once the macro has been created). Any raster graphic or sizable sequence of commands that is used more than once in a document should be printed using a macro. Refer to Chapter 12 for more information about when to use macros.

Contents

Introduction	14-1
Missing Characters/Graphics at Edges of the Page . .	14-1
Running Out of Memory (Error 20)	14-1
Print Overrun (Error 21)	14-3
Reset (E _c E) Deleting Temporary Fonts and Macros . .	14-3
PJL-Specific Problems	14-3
Reset (E _c E) Causing Printing of Partial Pages	14-4
Clipped Graphic Images	14-4
HP-GL/2 Images Not Printing Properly	14-5

Introduction

This chapter is not an exhaustive list of potential problems, but it does indicate some of the more common problems that programmers experience. If you have not already done so, please read through these few pages to familiarize yourself with these potential problems so that they can be avoided. If you are already experiencing difficulty with any of these situations, hopefully this will help you solve the problem. Be sure to check the *PCL 5 Printer Language Technical Reference Manual* for related topics if you are still having problems.

Missing Characters or Graphics Along the Edges of the Page

Attempting to print too close to the edge of the page may cause characters or graphics to extend into the unprintable region, clipping them at the boundary.

For text, unexpected clipping can be prevented by calculating the page boundaries and then using font metric data to be sure characters fall within the printable area. (See Chapter 4 for examples demonstrating printing at the outer-most limits of the page.)

Running Out of Memory (Error 20)

With the standard printer memory, downloading too many fonts, rasterizing a large bitmap character from a scalable font, inefficient use of HP-GL/2 commands, or too large of a raster graphics image can cause an error 20, meaning that the printer requires more user memory to print a particular job. Solutions to this problem include the following:

- Downloaded fonts and macros may be occupying a large amount of user memory. If these are deleted, there may be plenty of memory left to print the job successfully. This solution may be a good one, although other users in

a multi-user workgroup may be adversely affected. See the *Memory Usage* section in Chapter 2 for more information.

- Reduce the graphics resolution to reduce the memory required to print the graphics image.
- When using large fonts, send only the characters necessary to print the job. For example, if the job requires a 65-point headline, download only those characters that are required in the headline. The job will print faster and will require less user memory to print successfully. (With scalable fonts, the universal characters and limited sensitivity characters [such as “. ; , |”, Line Draw and Greek characters] are already in the printer, so they won’t have to be downloaded. Note also that for scalable fonts, all of the parts of compound characters must be downloaded.)
- For LaserJet IIIP/4 applications, use adaptive compression to reduce the size of raster graphics so that less memory is required.
- Install additional user memory.
- Whenever possible, use printer fonts and the highest level of HP-GL/2 commands in your vector drawings. Stroke-to-fill techniques are extremely inefficient for both data transfer and user memory requirements.

Print Overrun (Error 21)

In some situations, complex job formatting causes the printer to respond with a print overrun (data saturation) error (Error 21). If the printer is a PCL 5 printer and has enough memory installed, a *page protection* mode is accessible from the control panel (for the LaserJet 4 printer, it can be enabled using the PJL SET command). See the *Memory Usage* section in Chapter 2 for more information on this feature. Note that changing the page protection setting reconfigures printer memory, and in the process erases all perishable data.

Reset (E_CE) Deleting Temporary Fonts and Macros

A printer reset clears all temporary fonts and macros from memory. Turning the printer power off clears *all* downloaded fonts and macros (temporary *and* permanent). In situations that use the same fonts repeatedly, fonts can be downloaded as permanent so that printer resets will not clear them from memory. See the *PCL 5 Printer Language Technical Reference Manual* for information on temporary and permanent fonts.

PJL-Specific Problems

Listed below are some things to be aware of when using PJL.

- The UEL command (E_C%-12345X) must always begin *and* end a job destined for a PJL printer. The UEL command at the beginning of the job must be immediately followed by the @PJL command prefix. There can be no spaces or other characters between the “X” in the UEL command and the “@” in the PJL command prefix.
- Each line of PJL code must end with a line feed character (<LF>).

- Changing the language resolution or page protection setting causes all perishable data to be lost, including downloaded fonts and macros.
- After the @PJL ENTER LANGUAGE = PCL <LF> command is issued, PCL data must immediately follow it, beginning with the $\text{E}cE$ command.

Reset ($\text{E}cE$) Causing Printing of Partial Pages

If the printer receives a reset after it receives printable characters, the current page is ejected and a new page begins to be formatted. This problem is avoided if the reset is issued as the first and last command in a print job, but not anywhere in-between.

Clipped Graphic Images

With the Raster Graphics System, the size of the picture boundaries is defined by the raster width and raster height commands. Any portion of the image that would have extended beyond these boundaries is clipped. Unexpected clipping can result when the raster height and raster width commands are not set to the desired size.

In the HP-GL/2 mode, those images that are defined absolutely (instead of relatively) will be clipped at the PCL picture frame boundaries. Solutions include increasing the picture frame size or using relative commands so that the image is automatically scaled to fit the picture frame.

HP-GL/2 Images Not Printing Properly

If you are experiencing difficulty printing an HP-GL/2 image, one of the following syntax items may be the cause:

- Whenever the HP-GL/2 mode is used, the last HP-GL/2 command before returning to PCL mode must be terminated by a semicolon.
- When using the DT (Define Label Terminator) command, a comma must be placed after the designated terminator and before the mode parameter. For example, if you are designating the circumflex (caret) as a terminator and don't want the label terminator printed, either of the following commands are correct: DT^,1; or DT^; .
- A pen *must* always be selected using the SP command, whether white (SP0) or black (SP1). Failure to select a pen can cause unexpected results.
- When using multiple line widths in an HP-GL/2 drawing, it is preferable to use one logical pen (SP1) and modify the line widths with the PW command, rather than specifying several logical pens at different line widths. Using the SP1 and PW commands as mentioned ensures that line widths will scale correctly.

LaserJet Printer Features and Compatibility

A

Contents

Feature Support Table.	A-1
LaserJet Compatibility Issues	A-10
Memory and Performance	A-10
Lossy/Lossless Compression.	A-11
Floating CAP.	A-12
Video Interface and Expanded I/O	A-14
Modular I/O.	A-14
Miscellaneous Compatibility Issues.	A-14

Feature Support Table

The table below compares the features of the different printer models in the PCL 5 LaserJet printer family. The section following the matrix discusses some of the differences that affect compatibility between the PCL 5 LaserJet printers and the PCL 4 printers.

Note



In the following table, “ns” indicates the feature is not supported and “YES” indicates it is supported. If the command value field parameters are not listed, all parameters are supported by all PCL 5 printers that support that command.

The value in parentheses in the “Command” column identifies the parameter value for that particular selection. For example, “A4 (26)” means that 26 is the value required to select A4 paper using the Page Size command ($\text{E}_c\&l26A$).

PCL COMMANDS						
Command Function	Command	LaserJet Printer Model				
		III	IIID	IIISi	IIIP	4
JOB CONTROL						
Universal Exit Language (UEL)	$\text{E}_c\% - 12345X$	ns	ns	YES	ns	YES
Reset	E_cE	YES	YES	YES	YES	YES
Number of Copies	$\text{E}_c\&l\#X$	YES	YES	YES	YES	YES
Simplex/Duplex Print	$\text{E}_c\&l\#S$	ns	YES	YES	ns	ns
Left (Long-Edge) Offset Registration	$\text{E}_c\&l\#U$	YES	YES	YES	YES	YES
Top (Short-Edge) Offset Registration	$\text{E}_c\&l\#Z$	YES	YES	YES	YES	YES
Unit of Measure	$\text{E}_c\&u\#B$	ns	ns	ns	ns	YES

PCL COMMANDS						
Command Function	Command	LaserJet Printer Model				
		III	IIID	IIISi	IIIP	4
PAGE CONTROL						
Page Size	Ⓔ&l#A	YES	YES	YES	YES	YES
	Executive (1)	YES	YES	YES	YES	YES
	Letter (2)	YES	YES	YES	YES	YES
	Legal (3)	YES	YES	YES	YES	YES
	A4 (26)	YES	YES	YES	YES	YES
	Intl. B5 Envelope (100)	ns	ns	ns	ns	YES
	Monarch Envelope (80)	YES	YES	YES	YES	YES
	Com10 Envelope (81)	YES	YES	YES	YES	YES
	Intl. DL Envelope (90)	YES	YES	YES	YES	YES
	Intl. C5 Envelope (91)	YES	YES	ns	YES	YES
Page Length	Ⓔ&l#P	YES	YES	YES	YES	YES
Orientation	Ⓔ&l#O	YES	YES	YES	YES	YES
Page Side Selection	Ⓔ&a#G	ns	YES	YES	ns	ns

PCL COMMANDS						
Command Function	Command	LaserJet Printer Model				
		III	IIID	IIISi	IIIP	4
Paper Source	ⒺⒻⒼⒽ	YES	YES	YES	YES	YES
	Upper Tray (1) or Standard Cassette	YES	YES	YES	YES	YES
	Manual Feed (2)	YES	YES	YES	YES	YES
	Manual Feed Envelope (3)	YES	YES	YES	YES	YES
	Lower Tray (4) or Multipurpose Tray	ns	YES	YES	YES	YES
	Large Capacity Paper Source (5)	ns	ns	ns	ns	YES
	Envelope Feeder (6)	ns	YES	YES	ns	YES
Paper Destination	ⒺⒻⒼⒾ	ns	ns	YES	ns	ns
Print Direction	ⒺⒻⒼⒿ	YES	YES	YES	YES	YES
Left Margin	ⒺⒻⒼⓀ	YES	YES	YES	YES	YES
Right Margin	ⒺⒻⒼⓁ	YES	YES	YES	YES	YES
Clear Horizontal Tab	ⒺⒻⒾ	YES	YES	YES	YES	YES
Top Margin	ⒺⒻⒼⓂ	YES	YES	YES	YES	YES
Text Length	ⒺⒻⒼⓃ	YES	YES	YES	YES	YES
Perforation Skip	ⒺⒻⒼⓄ	YES	YES	YES	YES	YES
Horizontal Motion Index	ⒺⒻⓀⓂ	YES	YES	YES	YES	YES
Vertical Motion Index	ⒺⒻⒼⒸ	YES	YES	YES	YES	YES
Line Spacing	ⒺⒻⒼⒹ	YES	YES	YES	YES	YES

PCL COMMANDS						
Command Function	Command	LaserJet Printer Model				
		III	IIID	IIISi	IIIP	4
CURSOR POSITIONING						
Horizontal Position	Esc&a#C	YES	YES	YES	YES	YES
	Esc&p#X	YES	YES	YES	YES	YES
	Esc&a#H	YES	YES	YES	YES	YES
Vertical Position	Esc&a#R	YES	YES	YES	YES	YES
	Esc&a#Y	YES	YES	YES	YES	YES
	Esc&a#V	YES	YES	YES	YES	YES
Half Line Feed	Esc=	YES	YES	YES	YES	YES
Line Termination	Esc&k#G	YES	YES	YES	YES	YES
Push/Pop Position	Esc&f#S	YES	YES	YES	YES	YES
FONT SELECTION						
The “primary” font selection commands in this table can be specified as “secondary” by replacing the left parenthesis “(” in the command with a right parenthesis “)”.						
Symbol Set (Primary)	Esc(ID	YES	YES	YES	YES	YES
Spacing (Primary)	Esc(s#P	YES	YES	YES	YES	YES
Pitch (Primary)	Esc(s#H	YES	YES	YES	YES	YES
Height (Point Size, Primary)	Esc(s#V	YES	YES	YES	YES	YES
Style (Primary)	Esc(s#S	YES	YES	YES	YES	YES
Stroke Weight (Primary)	Esc(s#B	YES	YES	YES	YES	YES
Typeface (Primary)	Esc(s#T	YES	YES	YES	YES	YES
Font Selection by ID # (Primary)	Esc(#X	YES	YES	YES	YES	YES

PCL COMMANDS						
Command Function	Command	LaserJet Printer Model				
		III	IIID	IIISi	IIIP	4
Select Default Font (Primary)	Ⓔc(3@	YES	YES	YES	YES	YES
Underline	Ⓔc&d0D	YES	YES	YES	YES	YES
	Ⓔc&d3D	YES	YES	YES	YES	YES
	Ⓔc&d@	YES	YES	YES	YES	YES
Transparent Print Data	Ⓔc&p#X[data]	YES	YES	YES	YES	YES
FONT MANAGEMENT						
Assign Font ID#	Ⓔc*c#D	YES	YES	YES	YES	YES
Font Control	Ⓔc*c#F	YES	YES	YES	YES	YES
USER-DEFINED SYMBOL SET						
Symbol Set ID Code	Ⓔc*c#R	ns	ns	ns	YES	YES
Define Symbol Set	Ⓔc(f#W[data]	ns	ns	ns	YES	YES
Symbol Set Control	Ⓔc*c#S	ns	ns	ns	YES	YES
SOFT FONT CREATION						
Font Descriptor (Font Header)	Ⓔc)s#W[data]	YES	YES	YES	YES	YES
Character Code	Ⓔc*c#E	YES	YES	YES	YES	YES
Download Character	Ⓔc(s#W[data]	YES	YES	YES	YES	YES
MACROS						
Macro ID	Ⓔc&f#Y	YES	YES	YES	YES	YES
Macro Control	Ⓔc&f#X	YES	YES	YES	YES	YES

PCL COMMANDS						
Command Function	Command	LaserJet Printer Model				
		III	IIID	IIISi	IIIP	4
PRINT MODEL IMAGING						
Source Transparency Mode	E _c *v#N	YES	YES	YES	YES	YES
Pattern Transparency Mode	E _c *v#O	YES	YES	YES	YES	YES
Area Fill ID	E _c *c#G	YES	YES	YES	YES	YES
Select Current Pattern	E _c *v#T	YES	YES	YES	YES	YES
	Solid Black (0)	YES	YES	YES	YES	YES
	Solid White (1)	YES	YES	YES	YES	YES
	Shading Pattern (2)	YES	YES	YES	YES	YES
	Cross-Hatch Pattern (3)	YES	YES	YES	YES	YES
	User-Defined Pattern (4)	ns	ns	ns	YES	YES
USER-DEFINED PATTERN						
Define Pattern	E _c *c#W[data]	ns	ns	ns	YES	YES
Set Pattern Reference Point	E _c *p#R	ns	ns	ns	YES	YES
User-Defined Pattern Control	E _c *c#Q	ns	ns	ns	YES	YES
RASTER GRAPHICS						
Graphics Resolution	E _c *t#R	YES	YES	YES	YES	YES
	75 dpi (75)	YES	YES	YES	YES	YES
	100 dpi (100)	YES	YES	YES	YES	YES
	150 dpi (150)	YES	YES	YES	YES	YES
	200 dpi (200)	ns	ns	ns	ns	YES
	300 dpi (300)	YES	YES	YES	YES	YES
	600 dpi (600)	ns	ns	ns	ns	YES

PCL COMMANDS						
Command Function	Command	LaserJet Printer Model				
		III	IIID	IIISi	IIIP	4
Graphics Presentation	ⒺⒸ*Ⓡ#F	YES	YES	YES	YES	YES
Raster Height	ⒺⒸ*Ⓡ#T	YES	YES		YES	YES
Raster Width	ⒺⒸ*Ⓡ#S	YES	YES	YES	YES	YES
Start Graphics	ⒺⒸ*Ⓡ#A	YES	YES	YES	YES	YES
Y Offset	ⒺⒸ*Ⓑ#Y	YES	YES	YES	YES	YES
Set Compression Mode	ⒺⒸ*Ⓑ#M	YES	YES	YES	YES	YES
	Unencoded (0)	YES	YES	YES	YES	YES
	Run-Length (1)	YES	YES	YES	YES	YES
	TIFF (2)	YES	YES	YES	YES	YES
	Delta Row (3)	YES	YES	YES	YES	YES
	Adaptive (5)	ns	ns	ns	YES	YES
Transfer Raster Data	ⒺⒸ*Ⓑ#W[data]	YES	YES	YES	YES	YES
End Graphics	ⒺⒸ*ⓇB	YES	YES	YES	YES	YES
	ⒺⒸ*ⓇC	YES	YES	YES	YES	YES
RECTANGULAR AREA FILL						
Horizontal Rectangle Size	ⒺⒸ*Ⓒ#A	YES	YES	YES	YES	YES
	ⒺⒸ*Ⓒ#H	YES	YES	YES	YES	YES
Vertical Rectangle Size	ⒺⒸ*Ⓒ#B	YES	YES	YES	YES	YES
	ⒺⒸ*Ⓒ#V	YES	YES	YES	YES	YES

PCL COMMANDS						
Command Function	Command	LaserJet Printer Model				
		III	IIID	IIISi	IIIP	4
Pattern ID (Area Fill ID)	ε _c *c#G	YES	YES	YES	YES	YES
	# = 1 to 100 for shading	YES	YES	YES	YES	YES
	# = 6 for Cross-Hatch Patterns	YES	YES	YES	YES	YES
	# = 0 - 32767 for User-Defined Patterns	ns	ns	ns	YES	YES
Fill Rectangular Area	ε _c *c#P	YES	YES	YES	YES	YES
	Black (solid) (0)	YES	YES	YES	YES	YES
	White (erase) (1)	YES	YES	YES	YES	YES
	Shaded (gray) (2)	YES	YES	YES	YES	YES
	Cross-Hatch (3)	YES	YES	YES	YES	YES
	User-Defined (4)	ns	ns	ns	ns	YES
	Current Pattern (5)	YES	YES	YES	YES	YES
PICTURE FRAME (For Vector Graphics)						
Picture Frame Horizontal Size	ε _c *c#X	YES	YES	YES	YES	YES
Picture Frame Vertical Size	ε _c *c#Y	YES	YES	YES	YES	YES
Set Picture Frame Anchor Point	ε _c *c#T	YES	YES	YES	YES	YES
HP-GL/2 Horizontal Plot Size	ε _c *c#K	YES	YES	YES	YES	YES
HP-GL/2 Vertical Plot Size	ε _c *c#L	YES	YES	YES	YES	YES

PCL COMMANDS						
Command Function	Command	LaserJet Printer Model				
		III	IIID	IIISi	IIIP	4
Enter HP-GL/2 Mode	Ⓔ%#B	YES	YES	YES	YES	YES
	Ⓔ%0B	YES	YES	YES	YES	YES
	Ⓔ%1B	YES	YES	YES	YES	YES
Enter PCL Mode	Ⓔ%#A	YES	YES	YES	YES	YES
PROGRAMMING HINTS						
Display Functions	ⒺY	YES	YES	YES	YES	YES
	ⒺZ	YES	YES	YES	YES	YES
End-of-Line Wrap	Ⓔ&s0C	YES	YES	YES	YES	YES
	Ⓔ&s1C	YES	YES	YES	YES	YES
PCL STATUS READBACK						
Set Status Readback Location Type	Ⓔ*s#T	ns	ns	ns	ns	YES
Set Status Readback Location Unit	Ⓔ*s#U	ns	ns	ns	ns	YES
Inquire Status Readback Entity	Ⓔ*s#I	ns	ns	ns	ns	YES
Flush All Pages	Ⓔ&r#F	ns	ns	ns	ns	YES
Free Memory Space	Ⓔ*s1M	ns	ns	ns	ns	YES
Echo	Ⓔ*s#X	ns	ns	ns	ns	YES
HP-GL/2 GRAPHICS						
Bezier Curves	BZ and BR	ns	ns	ns	ns	YES
All other HP-GL/2 Commands	See the <i>PCL 5 Comparison Guide</i> for a list of all the HP-GL/2 commands supported by the PCL 5 printers.	YES	YES	YES	YES	YES

The following table lists the PJL commands supported by the PCL 5 LaserJet printers. For more information about PJL commands, see the *PJL Technical Reference Manual*.

PJL COMMANDS					
PJL Command Name	LaserJet Printer Model				
	III	IIID	IIISi	IIIP	4
COMMENT	ns	ns	YES	ns	YES
DEFAULT	ns	ns	ns	ns	YES
DINQUIRE	ns	ns	ns	ns	YES
ECHO	ns	ns	ns	ns	YES
ENTER	ns	ns	YES	ns	YES
EOJ	ns	ns	ns	ns	YES
INFO	ns	ns	ns	ns	YES
INITIALIZE	ns	ns	ns	ns	YES
INQUIRE	ns	ns	ns	ns	YES
JOB	ns	ns	ns	ns	YES
OPMSG	ns	ns	ns	ns	YES
PJL (@PJL prefix followed by <LF>)	ns	ns	ns	ns	YES
RDYMSG	ns	ns	ns	ns	YES
RESET	ns	ns	ns	ns	YES
SET	ns	ns	ns	ns	YES
STMSG	ns	ns	ns	ns	YES
Universal Exit Language (UEL)	ns	ns	YES	ns	YES
USTATUS	ns	ns	ns	ns	YES
USTATUSOFF	ns	ns	ns	ns	YES

LaserJet Compatibility Issues

Memory and Performance

As a result of formatter enhancements made to allow for future product developments, the PCL 5 LaserJet printers may appear to be incompatible in some instances with PCL 4 LaserJet printers. The remainder of this chapter describes some of these differences:

The PCL 5 LaserJet printers allocate memory differently than the LaserJet series II and other PCL 4 printers. Every effort has been made to maintain backward compatibility between PCL 4 and PCL 5 so that jobs that run on the LaserJet series II, for example, will also run on a similarly configured PCL 5 printer. That is, a job running on a LaserJet series II printer with no additional memory or fonts should run the same on a similarly configured PCL 5 printer (with respect to output speed and memory). In most cases, the PCL 5 printers will outperform the PCL 4 printers with respect to speed and memory.

However, there may be some corner cases in which this does not occur. In order to support features such as font rotation and scaling, the PCL 5 printers have the capability to dynamically allocate as well as deallocate memory. In cases when the printer has dynamically allocated memory to store rotated characters or an entire scaled font bitmap, and subsequently runs into a “memory low” situation, memory is dynamically deallocated to accommodate the new request.

This automatic memory management may result in reduced output performance for the PCL 5 printers. When this occurs, the printers attempt to format the new page request by first deallocating all available RAM and then by flushing the paper path. In these cases, increasing the amount of internal memory will increase the printer’s output speed. However, as stated above, the likelihood of this occurring is minute, especially due to the substantially greater amount of memory present in the standard PCL 5 LaserJet printers compared to that in the PCL 4 printers (see the “Lossee/Lossless Compression” discussion later in this chapter).

In addition, the auto-rotation feature in the PCL 5 LaserJet printers can cause what appears to be a memory incompatibility with the LaserJet series II and IIP printers. This is due to the fact that in the process of internally “rotating” a font, the printer creates and stores the rotated version of each character bitmap in RAM.

Therefore, each time the printer rotates a font, it must allocate additional memory to store this font information. To minimize the memory impact, the printer dynamically rotates on a character-by-character basis rather than rotating the entire font.

Once the character has been rotated, it is available in memory for printer access. In order to maintain compatibility with the LaserJet series II printer, the PCL 5 printers have an internal font set that contains both the portrait and landscape bitmaps of the fonts that are internal to the LaserJet series II printer.

Lossee/Lossless Compression

The LaserJet 4 printer has a unique way of handling low memory situations. The printer provides the best image for the amount of available memory using data compression techniques. When the printer detects a low memory situation, it examines the page data and attempts to compress any bitmap images so that they print successfully. If an image is compressed at this point, the image is printed without any data loss (*lossless* compression). However, if the printer attempts to compress the image and there is still not enough memory, it compresses the image using a technique that requires less memory, but loses some of the data in the process (*lossee* compression). In most cases, the resulting image is very acceptable, providing a favorable way of handling a memory shortage.

Floating CAP

Unlike the LaserJet series II printer, the PCL 5 LaserJet printers will not allow the cursor position to “float” before printable data is received following a form feed. (The term *floating CAP* refers to a state at the start of a new page in which the current active cursor position is not yet defined.)

Because of this difference between the two printers, some situations may occur where the text at the top of the page is printed in a different vertical position on the PCL 5 LaserJet printers than on the PCL 4 printers. This change will not impact most applications, however there may be some problems in applications that print rules and text near the top of the page. The following example demonstrates how any problems with floating CAP can be avoided.

Example: Compatibility and Floating CAP

In this example, the cursor position is pushed *after* the VMI command is sent, resulting in a situation in which both the PCL 5 printers and the LaserJet series II printer produce compatible output. If the cursor was pushed *before* the VMI command, the PCL 5 printer would print the text above the rule and the LaserJet series II printer would print it below the rule (see Figure A-1).

E_cE	Reset the printer.
$E_c&l2A$	Select letter-size paper.
$E_c&l20C$	Specify a VMI of 20.
$E_c&f0S$	Push the cursor position.
This is a test.	Send text to the printer.
$E_c&f1S$	Pop the cursor position.
$E_c*c4560h36V$	Define a rule.
E_c*c0P	Print the rule.

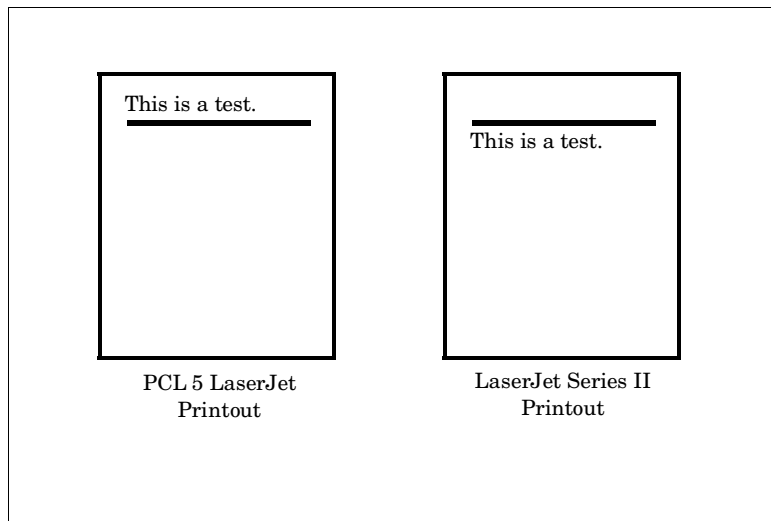


Figure A-1. The Results of Floating CAP

Video Interface and Expanded I/O

The expanded I/O slot in the back of the LaserJet series II printer is able to support both expanded I/O as well as video interface functionality. The video interface portion of the I/O slot is not included in the design of the PCL 5 LaserJet printers, so those products using the LaserJet series II video I/O will not operate properly with the PCL 5 printers.

In addition, the expanded I/O specifications for the PCL 5 printers are different than for the LaserJet series II printer. Those products that upgrade to the PCL 5 LaserJet printer specifications, however, will be backward compatible with the LaserJet series II printer. (See the *Modular I/O* discussion on the next page.)

Modular I/O

The LaserJet IIISi and LaserJet 4 printers have a modular I/O which features high speed, bi-directional signals, link/printer independence, and I/O configuration from the front panel. The modular I/O (MIO) is not compatible with the expanded I/O implementation.

Miscellaneous Compatibility Issues

Listed below are some subtle differences between the LaserJet series II and PCL 5 LaserJet printers. Most of these differences are so obscure or insignificant that they probably will be unnoticed. They are included here for completeness.

Default Symbol Set

If a PCL 5 LaserJet printer receives a symbol set command for which there is no matching font, the default symbol set is used. On the other hand, the LaserJet series II printer first attempts to use the previously selected symbol set before it uses the default symbol set.

Eighth Bit Shift Not Supported

The PCL 5 LaserJet printers do not support the eighth bit shift feature which allows shifting to the secondary default font based on the eighth bit of the character. The LaserJet series II printer had this feature, although it wasn't documented. Eighth bit shift has been used mostly in minicomputer environments and is a carry-over from previous

technologies. It is not the preferred method for font selection on the LaserJet family of printers.

Defaulting HMI After Font Changes

The PCL 5 LaserJet printers default the HMI of the newly selected font after all font parameter changes; the LaserJet series II printer does not default HMI if the only parameters that were changed are style, stroke weight, or typeface.

Internal Units

The internal unit of measure used by the PCL 5 LaserJet printers and all current PCL 4 printers is different than that of the LaserJet series II printer. All LaserJet printers introduced after the LaserJet series II use 1/7200th of an inch increments instead of the 1/3600th of an inch increments used in the LaserJet series II. This measurement unit is used by the printer to represent PCL Units, d-cipoints, column, and row positioning commands. The positioning command values are translated to internal units during formatting and rounded to physical dot positions when data is printed. The LaserJet series II printer, in addition to using a different internal unit, also truncates that value rather than rounding it. In some obscure cases, this difference in internal units may slightly affect printer output.

Upper-Case Binary Transfer Character

Unlike the LaserJet series II printer, the PCL 5 LaserJet printers only support the uppercase binary transfer character (W) in the transfer raster data command, ϵ_c ***b#W[raster data]**. In other words, if this command is terminated with a lower-case w, the printer will ignore the command.

Return Model Number Not Supported

The PCL 5 LaserJet printers do not support the “return model number” escape sequence and will ignore it. Although it was not documented or supported, the LaserJet series II printer had this command as one of its features. The LaserJet 4 printer provides a similar feature using the PJI INFO ID command. See the *PJL Technical Reference Manual* for more information about using this feature.

Self Test and Font Printout

Pressing the PRINT FONTS key on the control panel or running a printer self test (either from the control panel or programmatically) automatically resets the PCL 5 LaserJet printers. As with any reset, any existing temporary fonts or macros are automatically erased. (Unlike the PCL 5 LaserJet printers, the LaserJet series II printer doesn't perform a reset during a PRINT FONTS or self test execution.)

Esc?DC1

The PCL 5 LaserJet printers do not support the Esc?DC1 escape sequence and will ignore it upon receipt. The LaserJet series II printer was the last printer to support this command (it was an undocumented feature).

Display Functions Mode and Macros

The PCL 5 LaserJet printers ignore the display functions mode command if it is invoked from within a macro. This is not the case for the LaserJet series II printer.

Font Management Commands and Macros

The PCL 5 LaserJet printers will not ignore font management commands within a macro as the LaserJet series II printer does.

Logical Page Position

The logical page position with respect to the physical page is different for the PCL 5 LaserJet printers than it is for the LaserJet series II printer. For all LaserJet printers introduced after the LaserJet series II, the logical page is centered on the physical page. The left and right margins for all paper sizes in portrait mode (at 300 dpi) are 75 dots and in landscape mode they are 50 dots each. For the LaserJet series II printer, the logical page is not centered on the physical page; there is a 50-dot left margin for all paper sizes, a 100-dot right margin for US paper sizes, and a 92-dot right margin for A4 paper (in portrait mode).

Starting Cursor Position

The algorithm that calculates the starting cursor position for each page is different for PCL 5 than for the PCL 4. For the PCL 5 LaserJet printers, the starting cursor position is below the top margin by 75% of the current VMI; for LaserJet series II, the value is 72%. This difference will cause the top line of text to be printed slightly lower on the page for the PCL 5 LaserJet printers than for the LaserJet series II printer.

Unsupported Page Length or Page Size Command

An unsupported page length command is one that exceeds the maximum supported page size of the printer. Upon receiving an unsupported page length or page size command, if printable data has been received the PCL 5 LaserJet printers eject a page and continue printing and formatting for the current paper size. In comparison, the LaserJet series II printer ignores the command. Because of this difference, the PCL 5 LaserJet printers may appear more sensitive to erroneous page length/size commands than the LaserJet series II printer.

Raster Images Across Page Boundaries

The LaserJet series II printer allows a raster image to cross page boundaries and maintain the raster parameters (mode, resolution, and left margin). The PCL 5 LaserJet printers will not allow this to happen and will automatically close any open raster images as well as all associated raster parameters. The effect of this closure is that raster escape sequences that alter the resolution, mode, or left margin on the subsequent page will be recognized by the PCL 5 LaserJet printers but not by the LaserJet series II printer.

Contents

Introduction.	B-1
Fonts and Print Direction	B-1
Printing Rules Using the Print Model.	B-3
Print Model Font Effects	B-3
HP-GL/2 Font Effects	B-6
HP-GL/2 Graphics	B-9
Raster Graphics Compression	B-10

Introduction

Hewlett-Packard provides samples of short programs written in C programming language to demonstrate some of the PCL 5 LaserJet printer features. The executable code and source code for each program are included on the HP-provided disks so that you may see how each feature was used.

This appendix shows scanned images and file names of the print samples that each of the included programs prints. It also gives a short explanation of the demonstrated features. Although many of the programs demonstrate more than one feature, they are organized by the most dominant feature.

Fonts and Print Direction

The PCL 5 LaserJet printers allow you to print in four orientations: portrait, reverse portrait, landscape, and reverse landscape. Using the *print direction* command, text and graphics may be printed in more than one orientation on the same page.

PDIRSIMP.C

This program demonstrates using the print direction command to print the word “hello” in portrait orientation (0 degrees) and the word “there” in landscape orientation (90 degrees).

```
hello there
```

PDIRFONT.C

This program prints samples of the scalable CG Times typeface ranging in size from 1 point to 39 point. The print direction command is used to print the same fonts upside-down (reverse portrait orientation).



Printing Rules Using the Print Model

Rules can be printed as black, shaded, patterned, or white rules using the Print Model.

RULEMODS.C

This program shows a sample of a black rule, a white rule and a shaded rule that were created using the Print Model. The black rule is printed first and then the other two rules “erase” the black using the *source* and *pattern transparency mode* commands.



Print Model Font Effects

The LaserJet printers containing PCL 5 can print bit-mapped and scalable fonts in a wide range of sizes and styles. Using the print model, these fonts may be filled with shades and patterns or printed as reverse type. The following sample programs demonstrate some of the font effects that the PCL 5 LaserJet printers can easily perform.

SHADFONT.C

This sample uses the *Print Model* to produce fonts filled with two different shading patterns.

25 pt Sample Italic Shaded Text

25 pt Sample Italic Shaded Text

RULEFONT.C

In this program example, the *Print Model* is used to print shaded and pattern-filled fonts on a black background.



25 pt Sample Italic Patterned
25 pt Sample Italic Patterned
25 pt Sample Italic Patterned

PTNFNTOI.C

Similar to the RULEFONT.C program, this program prints more sizes of fonts filled with shades of gray and patterns. To print this file, you need to download the appropriate 36- and 48-point Helv Outline fonts or the printer will substitute another font.



25 pt Sample Italic Patterned
36 pt Helv Outline
Patterned Text
48 pt Helv Out
Patterned Text

SHADOW.C

The SHADOW.C program produces a landscape 200-point CG Times Italic font that is given the appearance of depth using the Print Model. The word “Galaxy” is printed in black, and then overlaid with six slightly offset (in the y direction) gray-shaded images of the same word.



SHADOW2.C

This program is almost identical to the SHADOW.C program except that the images are offset in the x *and* y direction.



HP-GL/2 Font Effects

With the printer's HP-GL/2 capability, fonts may be rotated to any angle and special effects such as mirroring and outlining may be done.

HPGLDI45.C

This program demonstrates the use of the HP-GL/2 *DI* (absolute direction) command to print the word "Galaxy" at a 45 degree angle. The sample on the left uses the HP-GL/2 stick font and the sample on the right uses the CG Times Italic font.

Galaxy

Galaxy

Galaxy

Galaxy

HPGLAB82.C

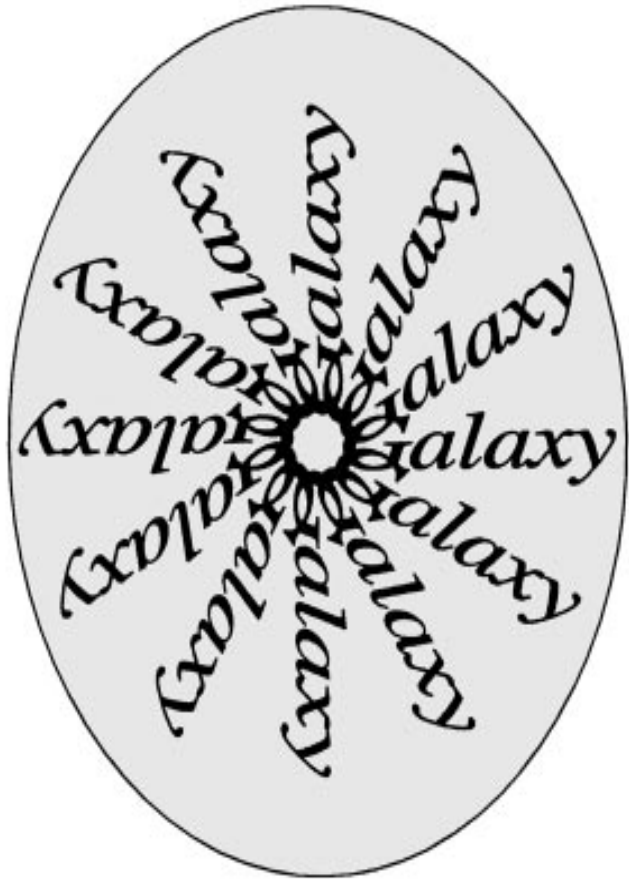
The HPGLAB82.C program produces an anisotropically scaled font along with its mirror image. The HP-GL/2 *SI* (absolute character size) command was used to create the mirror image.

Galaxy

Galaxy

HPGLLAB5.C

This program prints the word “Galaxy” rotated every 30 degrees and surrounded by an anisotropically scaled circle. The HP-GL/2 DI (absolute direction) command is used to rotate the fonts. The example also demonstrates anisotropic scaling of the CG Times font.

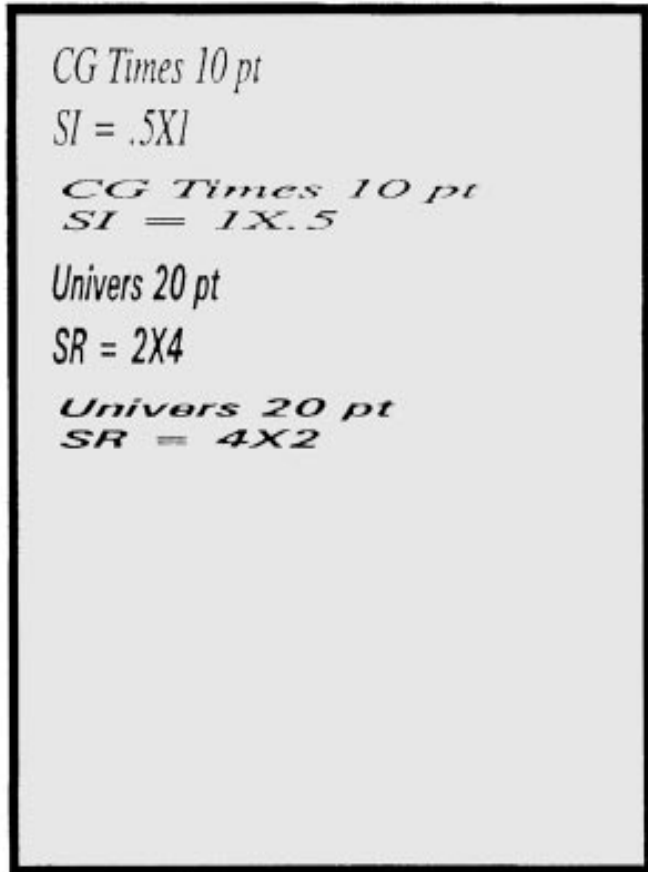


HPGLLAB4.C

This program is nearly identical to the HPGLLAB5.C program except the HP-GL/2 stick font is used instead of the CG Times font.

SISRSAMP.C

The SISRSAMP.C program shows the use of the *SI* (absolute size) and *SR* (relative size) commands to scale fonts. If the picture frame size is reduced, the border is automatically made proportionately narrower and the Univers (relatively scaled) fonts are also scaled in proportion to the change in picture frame size. The CG Times text, because it is scaled using the *SI* (absolute size) command, stays the same size regardless of picture frame size.

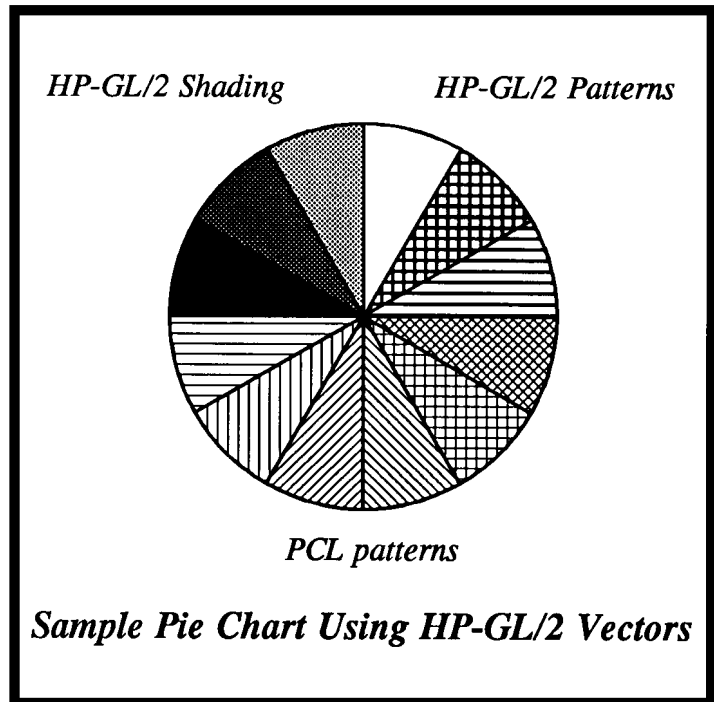


HP-GL/2 Graphics

HP-GL/2 allows you to import existing HP-GL/2 graphs or to create graphics from within your application.

PIEWEDGE.C

The PIEWEDGE.C program demonstrates a fairly complex pie chart that prints successfully without enabling the page protection mode. The program also demonstrates using HP-GL/2 shading and hatching patterns and also uses PCL patterns. The CG Times font is used to label the pie chart.



Raster Graphics Compression

The PCL 5 LaserJet printers support several raster data compression modes as discussed in Chapter 9. Two of these are demonstrated in the following examples.

ARROW1.C

This program prints an arrow using raster data that has been compressed using run-length encoding (mode 1).



AROWTIFF.C

This program prints the same arrow as ARROW1.C using the TIFF, or PackBits compression mode (mode 2).

AROWSHAD.C

The AROWSHAD.C program prints a shaded version of the same arrow. The Print Model is used to shade the image and the TIFF compression mode is used to condense the data.

AROWPMOD.C

This program demonstrates the use of the Print Model to print a shaded arrow on top of a black background. The raster data is encoded using mode 2 compression (TIFF).



Contents

Introduction.	C-1
Rambo vs. Marie.	C-1
Unbound Scalable Fonts and Character Data	C-2
What Does Rambo Do?.	C-2
The Rambo Command Line	C-5
Return Codes	C-9
Symbol Set Maps	C-9
Example Implementation Using Rambo	C-12
Show a List of Typefaces	C-12
Show a List of Symbol Sets	C-16
Make a Scalable Font File	C-16
Show Command-Line Usage	C-17
Printing the Test Files	C-18

Introduction

The Rambo utility allows you to create downloadable scalable fonts from Intellifont Library files. As mentioned in Chapter 7, integrating Intellifont without use of the Auto-Font Support Installer involves adding the Rambo program module to your software application. If you are familiar with the Marie utility which was distributed with the HP LaserJet III printer introduction, the Rambo utility replaces Marie.

Rambo vs. Marie

In the LaserJet III, IIID, and IIISi printers, downloadable scalable fonts must be bound to a symbol set before downloading. This symbol set “binding” means that only those characters contained in the symbol set are included in the font. Therefore if another symbol set in the same scalable font is needed, then another scalable font must be downloaded. The LaserJet IIIP and LaserJet 4 printers eliminate the need to download another scalable font just to get another symbol set, and the Rambo utility provides this functionality. In contrast, the previously used Marie program was written to produce only symbol-set-bound scalable fonts.

To provide the “unbound” capability in the LaserJet IIIP/4 printers, a new font header type and format have been defined for unbound scalable fonts. Fonts in this format can be attached to any valid symbol set in the printer—either an internal symbol set or a soft symbol set (user-defined symbol set). The format of this new type of font header looks very much like the Intellifont Scalable Font Descriptor format (descriptor format 10).

Note



The definition of the font header fields and values is described in detail in the *PCL 5 Technical Reference Manual*.

An unbound scalable font contains enough characters to allow the font to be printed with any symbol set, as long as the font and the symbol set are compatible. The key

difference between bound and unbound scalable fonts is that a bound scalable font is fixed to the symbol set it was downloaded with, while an unbound scalable font can be bound to any symbol set at print time.

Unbound Scalable Fonts and Character Data

The character data requirements are slightly different for an unbound scalable font than for a bound scalable font. With bound fonts, the characters are numbered according to their code position in their symbol set. With unbound fonts, the character numbering changes. Since the LaserJet IIIP/4 printer's internal symbol sets are based on the HP Master Symbol List (MSL) numbering scheme, character numbers must match the MSL numbers when creating an unbound scalable font. For example, suppose the first character of an unbound scalable font is an exclamation point. The MSL number for the exclamation point is 1 whereas the value is 33 in the US ASCII symbol set. The application downloads a character code location of 1 when creating the unbound scalable font; at print time, when a symbol set is attached, the printer converts the character into a "bound" scalable character with character location 33.

HP MSL numbers of all characters in the HP LaserJet IIIP/4 printers are listed in Appendix H.

What Does Rambo Do?

Rambo generates both bound *and* unbound scalable fonts. Rambo is a superset of the Marie code and operates almost identically to Marie. Its operation is covered in this chapter. The Rambo utility uses a typeface library file as input and writes the PCL code for either a bound or unbound scalable font. As output, the Rambo program creates scalable soft fonts which can be downloaded directly to and scaled by the HP LaserJet IIIP/4 printers.

As mentioned in Chapter 7, Rambo builds a scalable font from a Library file by extracting character data from only those typeface-sensitive characters in the specified symbol set. (Most typefaces have 290 typeface-sensitive characters.) The resulting file is encapsulated by a PCL header, creating a scalable font file.

Note



The executable Rambo code is available on disk from HP; the name of the executable file is RAMBO.EXE.

When creating fonts for non-HP printers, a program module similar to Rambo can be used to convert Library files to another printer format.

The following Rambo-related files are included on the HP-supplied disks:

GT-SYM.C	Code to read in the symbol set data
R.C	The Rambo shell
WT-PCL.C	Code to write the font data for scalable fonts
IOBYTES.C	Code that handles low-level I/O
RD-LIB.C	Code that reads the library file
RAMBO.EXE	The executable code for Rambo
R.EXE	The executable code for the Rambo shell
ERRORS.H	Rambo include file
FAIS.H	Rambo include file
LIBRARY.H	Rambo include file
MARIE.H	Rambo include file
PCLFNTHD.H	Rambo include file
PORT.H	Rambo include file
STRIPPED.H	Rambo include file
RAMBO.MAK	Make file for RAMBO.EXE
R.MAK	Make file for R.EXE
TDD1.PCL	A user-defined symbol set needed to print DINGBATS.TXT
TDD1.PCL	A user-defined symbol set needed to print DINGBATS.TXT

TDD2.PCL	A user-defined symbol set needed to print DINGBATS.TXT
TDD3.PCL	A user-defined symbol set needed to print DINGBATS.TXT
TDDV.PCL	A user-defined symbol set needed to print DINGBATS.TXT
HPCG.NDX	Index file used to make unbound scalable fonts
ERRORS.TXT	Used by Rambo shell to print error message strings
MSL.TXT	Prints out all the characters in the MSL for an unbound scalable font
DINGBATS.TXT	Prints out all the Dingbats characters for an unbound scalable font
QMSL.TXT	A quick, three-page MSL printout, similar to MSL.TXT

The Rambo Command Line

The Rambo program is command-line driven and is used as described below:

```
RAMBO Typeface# SymSet [/TTypeface_dir]  
[/Ssymbol_set_dir]/[/Foutput_directory]/[/Ooutput_file]/[/U]
```

Where:

Required Parameters:

Typeface# is the number of the library file (i.e. 92500 for CG Times).

SymSet is the two-character abbreviation for the symbol set (i.e. R8 is Roman-8) or the two-character index file designator (i.e. CG for HPCG.NDX).

Optional Parameters:

/T specifies the directory where the library (*.TYP) files are found (default = \TD\TYPE).

/S specifies the directory where the symbol sets (*.SYM) and index files (*.NDX) are located (default = \TD\SYMBOLS).

/F specifies the directory to write the output of Rambo (default = \TD\FONTS).

/O sets the name of the output file (the default follows the naming convention discussion under *File Naming Convention* in chapter 6).

/U means that the user wants the scalable font to be unbound. If */U* is not present, the font will be bound to the specified symbol set.

The Rambo program parameters are defined in more detail as follows:

Typeface# (Required)—the *Typeface#* parameter is the 5-digit Agfa number (CG number) given to the scalable typeface. Some of the typeface numbers are listed in Table C-1, but rather than looking up numbers in the table, the typefaces and their corresponding numbers can be retrieved

by using the HP-supplied routine *type_util* as used in the R.EXE shell program. Refer to the example application entitled *Show a List of Typefaces* (later in this chapter).

Table C-1. Sample Typeface Numbers

Typeface#	Typeface Name
90133	Dom Casual
90267	Microstyle
90268	Microstyle Bold
90326	Brush
90349	Park Avenue
90460	Microstyle Bold Extended
90508	Uncial
92500	CG Times
92501	CG Times Italic
92504	CG Times Bold
92505	CG Times Bold Italic
94021	Univers Medium
94022	Univers Medium Italic
94023	Univers Bold
94024	Univers Bold Italic

The 5-digit typeface number is also the filename of the Library file stored in the \TD\TYPE directory when the typeface is installed with the AutoFont Support Installer or Type Director 2.X. For example, the library file 90133.TYP is the Dom Casual typeface as shown in the table.

SymSet (Required)—the *SymSet* parameter is a 2-character symbol set code (or the 2-character index file designator). The symbol set file is usually located in the \TD\SYMBOLS subdirectory and has a name such as TDUS.SYM, where the two characters after TD in the filename are the symbol set code. An index file has a name such as HPCG.NDX, where the two characters after HP are the index file designator. The symbol set files are included on the HP-supplied disks and also as standard Type Director symbol sets. A partial list of the Type Director symbol sets and their two-character codes is shown in Table C-2:

Table C-2. Sample Symbol Set Values

SymSet	Symbol Set Name
DT	DeskTop
E1	ECMA-94 Latin 1
LG	Legal
PC	PC-8
R8	Roman-8
TS	PS Text
US	ASCII
VI	Ventura International
VM	Ventura Math
VU	Ventura US
WN	Windows

typeface_dir (Optional)—The *typeface_dir* parameter is the path to the typeface directory where library-format typefaces (such as 90133.TYP) are stored. When typefaces are installed using the AutoFont Support Installer or the Type Director *Install Typefaces* menu, the resulting library files are placed in the \TD\TYPE subdirectory. The default path is C:\TD\TYPE, and is the path used if no parameter is specified. (The *Glue File* discussed at the end of Chapter 6 also contains typeface and path information.)

symbol_set_dir (Optional)—the *symbol_set_dir* parameter is the path to the symbol set directory where the symbol set files such as TDUS.SYM are stored. When Type Director is installed, it automatically stores all of the included symbol set files in the \TD\SYMBOLS subdirectory. The default path C:\TD\SYMBOLS is used if no parameter is specified.

output_dir (Optional)—the *output_dir* parameter is the path to the directory where the output file will be written (\TD\FONTS is the default).

output_file (Optional)—the *output_file* parameter is the file-name of the scalable font file created by the Rambo program. Note that this is not a full pathname for the file; it is just the file-name. The scalable font file will be written to the output directory (\TD\FONTS is the default). The default filename is determined by Rambo depending on the type-face, treatment, and symbol set. The file naming convention for scalable font files is described in the *File Naming Convention* discussion in Chapter 6.

**Example:
Using Rambo (#1)**

To create a Dom Casual scalable font file “bound” to the ASCII symbol set, and assuming the typeface and symbol set files are in the C:\TD\FONTS and C:\TD\SYMBOLS directories respectively, enter the following from the command line:

```
RAMBO 90133 US
```

The resulting file would be named DCR00USO.SFP and it would be placed in the current DOS directory.

**Example:
Using Rambo (#2)**

To create an unbound Dom Casual scalable font file, assuming that the typeface files are in the C:\TD\FONTS directory and the index file (HPCG.NDX) is in the C:\TD\SYMBOLS directory, enter the following from the command line:

```
RAMBO 90133 CG /U
```

The resulting file would be named DCR00CGO.SFP.

Example: Using Rambo (#3)

To create an unbound scalable font file in the Park Avenue typeface (while changing the typeface directory to D:\FONTS\TYPE and changing the output filename to PARK.UNB), enter the following:

```
RAMBO 90349 CG /U /TD:\FONTS\TYPE /OPARK.UNB
```

The resulting file would be named PARK.UNB and it would be placed in the current DOS directory. Note the lack of space between the options and the arguments in the command line. For example, there is no space between “/T” and “D:\FONTS\TYPE”, and between “/O” and “PARK.UNB”.

Return Codes

The error codes listed in Table C-3 are returned by Rambo to the calling process. These error codes are contained in the file ERRORS.TXT included on the HP-supplied disks. (ERRORS.H is also included and is necessary to compile R.C, the Rambo shell program.)

RAMBO.EXE will not print any error messages. R.EXE uses ERRORS.TXT to display errors returned by Rambo.

Symbol Set Maps

Symbol set files are included in Type Director in the *symbols* directory; the default *symbols* directory is C:\TD\SYMBOLS. The symbol set files have names such as TDss.SYM, where *ss* is the two-character symbol set code for a given symbol set. You can read symbol set files from the Type Director symbols directory, or copy only the files you wish to support into a separate directory or into the application directory. Symbol set maps for all of the default Type Director symbol sets are included in the *PCL 5 Printer Language Technical Reference Manual*.

For creating unbound scalable fonts, the index files (such as HPCG.NDX) should be copied to the *symbols* directory.

Note



The Type Director 2.X product is available for developers from HP upon request.

Table C-3. Rambo Error Codes

Error Code 0 (No Errors)

Error Codes 1-30 (Memory Errors)

- 1 Insufficient memory for font header
- 2 Insufficient memory for face header
- 3 Insufficient memory for global dir
- 4 Insufficient memory for global Intellifont segment
- 5 Insufficient memory for raster param segment
- 6 Insufficient memory for attribute header segment
- 7 Insufficient memory for validation structure
- 8 Insufficient memory for display header
- 9 Insufficient memory for compound char list
- 10 Insufficient memory for font alias table
- 11 Insufficient memory for copyright
- 12 Insufficient memory for typeface header segment
- 13 Insufficient memory for symbol
- 14 Insufficient memory for sorting codes
- 15 Insufficient memory for compound character parts

Error Codes 31-60 (Disk Access Errors)

- 31 Invalid destination path
- 32 Unable to open library file
- 33 Font file already exists
- 34 Unable to write to font file
- 35 Symbol set file not found
- 36 Disk full

Table C-3. Rambo Error Codes (Continued)

Error Codes 61-90 (File Structure Errors)

- 61 Library file is not a disk file
- 62 File segment directory not found
- 63 Segment size is larger than expected size
- 64 Too many characters
- 65 No global Intellifont segment found
- 66 Bad or missing raster param segment
- 67 Bad or missing attribute header segment
- 68 Bad or missing display header segment
- 69 Bad or missing compound char segment
- 70 Bad or missing font alias table
- 71 Bad or missing font alias segment
- 72 More than one typeface in font alias table
- 73 No HP font alias table found for this typeface
- 74 No copyright segment found
- 75 Bad or missing typeface header segment
- 76 Escape font descriptor size too large
- 77 Y-escapement for symbol is non-zero
- 78 Part of compound character not found
- 79 Symbol set read does not match request
- 80 Insufficient Symbol Set data
- 81 “/END” expected in symbol set file
- 82 Missing compound error

Error Codes 91-120 (Command Line Errors)

- 91 Invalid command line arguments

Example Implementation Using Rambo

An example application is included with Rambo to illustrate how to call the Rambo program and to work with typeface numbers and names. The source code is titled R.C and is designed to be a user interface to Rambo. In this manual, the sample application is referred to as the *Rambo shell*, or just the *shell*.

The shell can be used in two ways: by command line, or through menus. If R.EXE is called with no command line arguments, or they are invalid, the user will enter the menu interface. Through these menus, the user can set all the command line arguments, as well as getting information on available typefaces and symbol sets.

To fully understand the shell, print a listing of the R.C source code. The executable code for the shell application, R.EXE, is also supplied.

The Rambo shell can perform the following four functions:

- Show a list of available typefaces
- Show a list of standard symbol set abbreviations
- Make a scalable font file
- Show the command-line usage of the shell

Each of the four functions that the shell performs are accomplished by different routines. The routines are: *typeutil*, *print_error*, *menu*, *list_types*, *list_symbols*, *call_program*, *getstring*, and *valid_symset*. A map of the shell functions and its routines is shown in figure C-1. These four functions are explained in more detail in the following sections.

Show a List of Typefaces

Type Director maintains a typeface information file called TYPEFACE.TD that is updated every time a font is installed. The file is located in the \TD\TYPE subdirectory (default) and contains information about every typeface name and number that has been installed using Type Director. The shell program's *typeutil* routine reads typeface information from this file and performs five functions for

creating a list of typefaces. The *list_types* function of the shell program illustrates a way to use the *typeutil* routine to actually create a typeface list.



The TYPEFACE.TD file is created by Type Director and is used in the sample implementation as an example of what can be done to list typefaces; developers may create a similar file to indicate what typefaces are available to the user. See the *Glue File* discussion at the end of Chapter 6.

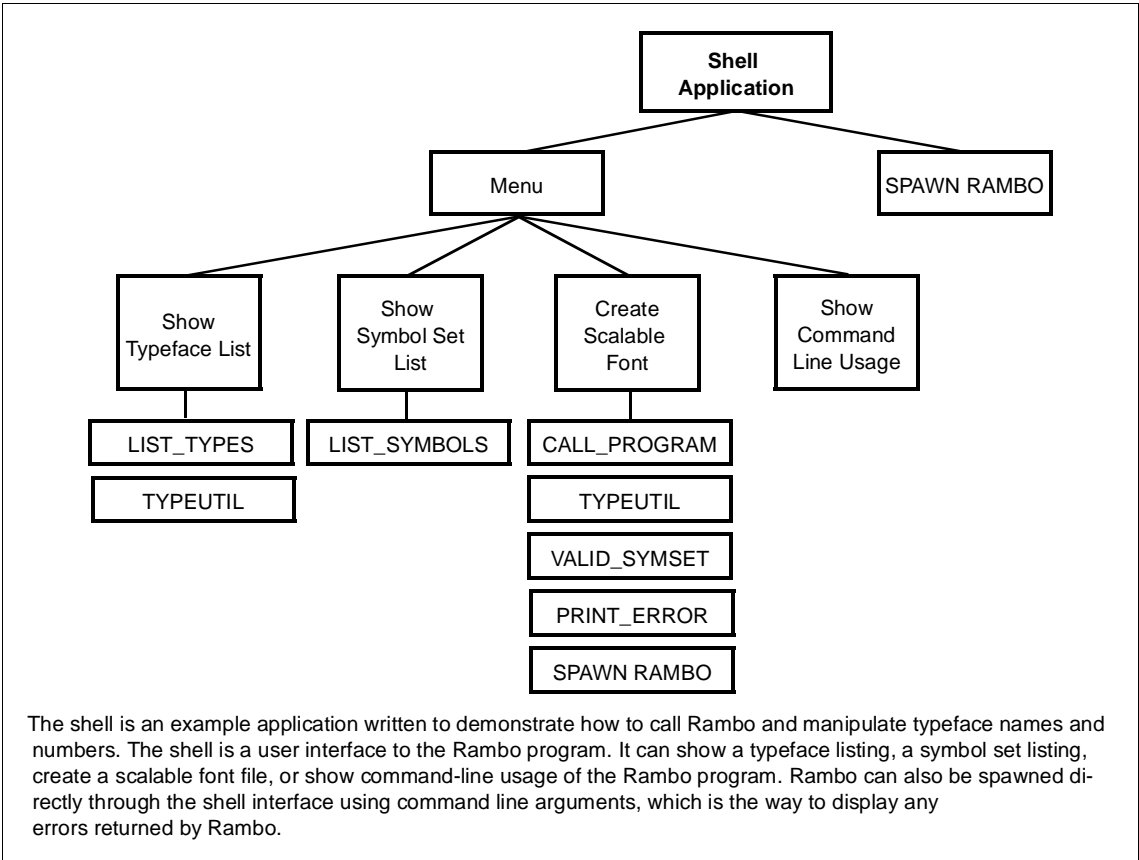


Figure C-1. The Rambo Shell Program's Four Functions

The *typeutil* routine performs the following five functions:

- 1) Returns data on the first entry in TYPEFACE.TD (first_entry)
- 2) Converts a given typeface name to its Agfa (CG) typeface number (name_to_num)
- 3) Converts a given Agfa (CG) typeface number to its typeface name (num_to_name)
- 4) Given an Agfa (CG) typeface number, returns the next typeface number in TYPEFACE.TD (next_num)
- 5) Given a typeface name, returns the next typeface name in TYPEFACE.TD (next_name)

Note



If a typeface is removed using Type Director, its typeface information still remains in the TYPEFACE.TD file.

Typeutil is an HP-supplied routine which can read the TYPEFACE.TD file in the typeface subdirectory (default: \TD\TYPE). It returns data on the first entry in the file, converts a typeface name to its number, converts a typeface number to its name, returns the next number in the file, or returns the next typeface name in the file.

Typeutil uses the following structure for conveying information, with the five entries in *option_list* corresponding to the five functions of *typeutil* (C programming language).

```
enum option_list {FIRST_ENTRY, NAME_TO_NUM,
                 NUM_TO_NAME, NEXT_NUM, NEXT_NAME};

typedef struct
{
    enum option_list option; /*Used to select desired function*/
    char l_name[51]; /* CG long typeface name */
    long number; /* CG Typeface ID # */
} TYPE_UTIL;
```

If a typeface name or number is not found, or if the end of file is reached during the `NEXT_NAME` or `NEXT_NUM` functions, *typeutil* will return “l_name = **NOT FOUND**” and “number = 0”.

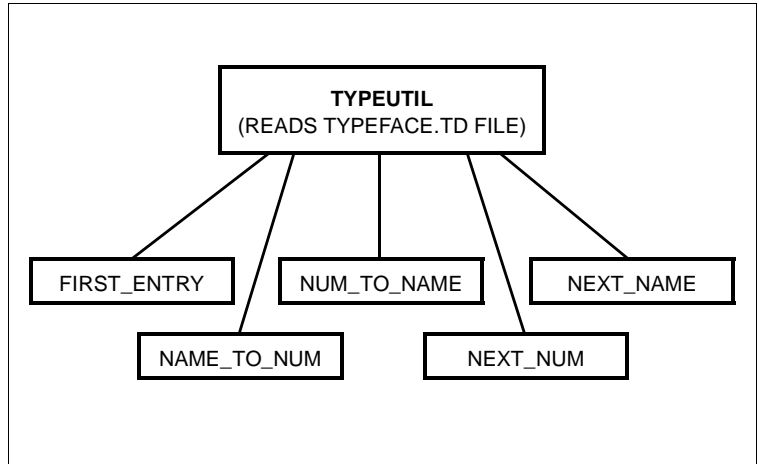


Figure C-2. The Functions of the Typeutil Program

The *list_types* function of the shell program is a good example of how to use the *typeutil* routine to create a typeface list. Refer to the routine *list_types* in the R.C source code. Each *.TYP file in the typeface directory (default C:\TD\TYPE) is read using DOS system calls. These file-names are converted to long integers and then *typeutil* is called to convert the typeface number to a typeface name. The typeface number and name are then displayed on the screen, and the next file is processed. You can use this same process to display a list of available typeface names to a user, and can then use the corresponding typeface number when calling the Rambo program.

Show a List of Symbol Sets

The *list_symbols* function of the shell program prints out a hard-coded list of two-character symbol set codes. (Refer to the *list_symbols* routine in the R.C program source code. Since the list is hard-coded, the symbol sets listed may or may not be in the user's symbol set directory (default C:\TD\SYMBOLS).

It is possible to read in each TDxx.SYM file from the symbol set directory and make a list of available symbol sets, but this is not implemented in the shell program. A similar process is recommended for software developers to use to display a list of symbol sets to an end user.

When using this shell program, if a user attempts to select an unavailable symbol set while making a scalable font, an error message will be printed on the screen (refer to the *valid_symset* routine at the end of the shell program source code).

Make a Scalable Font File

The *call_program* function of the shell program illustrates how to convert a scalable typeface file to a scalable PCL font file, which is the main reason for integrating Rambo.

The *call_program* routine performs a five-step process to set up and run the Rambo program:

- 1) It prompts the user for the typeface number.
- 2) It calls *typeutil* to read the TYPEFACE.TD file and converts the typeface number to a name to ensure that the typeface has been installed through Type Director. (NOTE: If a typeface is removed using Type Director, its information still remains in the TYPEFACE.TD file. This shell program does not check for the existence of the typeface file in the typeface directory; it just checks that the typeface was previously installed using Type Director. Also, instead of the TYPEFACE.TD file, the Glue File [discussed in Chapter 6] can also be used to generate a list of typefaces.)
- 3) It asks the user if the font should be unbound.

- 4) It prompts the user for the two-character symbol set code or index file designator.

If the font will be unbound, the shell checks to see if you chose “CG” as the index file; if not, a warning is printed.

If the font will be bound, the input is checked against the available symbol sets in the symbol set directory (default C:\TD\SYMBOLS) using the routine *valid_symset*. (Refer to the *valid_symset* routine in the R.C source code.) A file named TDss.SYM must exist there, where ss is the two-character abbreviation for the symbol set. If the selected typeface is “Dingbats”, the *call_program* routine also checks to ensure that the corresponding symbol set chosen is D1,D2, D3, DS, or DV (accepted Dingbat symbol sets); otherwise, an error message is printed.

- 5) It prompts the user for an output filename. If no filename is entered, the Rambo program uses the default filename. The construction of this default filename is discussed in the *output_file* section (earlier in this chapter).
- 6) Once the typeface number, symbol set, and the optional output filename are entered, the shell sets up the command line for the Rambo program. This command line is displayed on the screen so the user can see how the command line operates. The Rambo program is then called. Any returned error codes are printed using the routine *print_error*. Refer to the R.C source code for the *print_error* routine and the ERROR.TXT file for a listing of print errors.

Show Command-Line Usage

The shell program’s “Show Command-Line Usage” function prints information to the screen explaining how to use the shell to call the Rambo program using command-line arguments. If command-line arguments are used, the menu interface is bypassed. If the shell is called without command-line arguments, or with too many command-line arguments, the menu interface of the shell is then used. Otherwise, Rambo is called using the entered command line.

Command line usage of the shell is the same as command line usage for Rambo, and is demonstrated below:

```
R Typeface# SymSet [/Ttypeface_dir] [/Ssymbol_set_dir]
[/Foutput_dir] [/Ooutput_file] [/U]
```



The Rambo shell program does not function with Type Director 1.0 (the TYPEFACE.TD file format is different).

Printing the Test Files

The test files included with the Rambo packet include MSL.TXT, QMSL.TXT, and DINGBATS.TXT. These files may be used to test unbound scalable fonts that you create. MSL.TXT and QMSL.TXT are designed to print an unbound font that does not have any Dingbats characters. MSL.TXT prints a 10-page listing of characters and their descriptions. QMSL.TXT prints the same characters as MSL.TXT, but omits the descriptions, and is only 3 pages long.

To download and test an unbound scalable font, do the following steps:

1. Send an \textasciixc100D to set the font ID to 100.
2. Type `COPY /B UNBPCL.EO.PCL LPT1` to download the unbound scalable font.
3. Send an \textasciixc5F to make the font permanent.
4. Type `COPY /B MSL.TXT LPT1` to print the test file.

To print the DINGBATS.TXT test file:

1. Create an unbound ITC Zapf Dingbats scalable font.
2. Send an \textasciixc100D to set the font ID to 100.
3. Download the unbound ITC Zapf Dingbats font using `COPY /B`.
4. Send an \textasciixc5F to make the font permanent.

5. Use the binary copy command (COPY /B) to copy TDD1.PCL, TDD2.PCL, TDD3.PCL and TDDV.PCL to LPT1. (These are user-defined symbol sets needed to print dingbats.)
6. Use the binary copy command to download the DINGBATS.TXT file (COPY /B DINGBATS.TXT LPT1).

Contents

Introduction.	D-1
The BUILDSYM Kit	D-1
How Does BUILDSYM Work?	D-2
Using the Correct Character Numbers	D-3
Converting to MSL Numbers.	D-4
Using BUILDSYM	D-5
infile	D-5
outfile	D-5
Creating a Symbol Set Definition File (.SYM Files) . .	D-6
The .SYM Parameters	D-8
/PCL num/PCL char	D-8
/type.	D-8
/index.	D-8
/first code.	D-9
/last code	D-9
/requirements	D-9
/symbols	D-10
/end	D-10

Introduction

All PCL 5 LaserJet printers have many internal symbol sets that can be used with any of the internal scalable fonts or downloaded unbound fonts. The HP LaserJet IIIP and LaserJet 4 printers have the ability to add even more symbol set selections. If none of the internal symbol sets are suitable for the desired task, others can be added. These additions can be downloaded to the printer, much like a soft font or User-Defined pattern. Once in the HP LaserJet IIIP or LaserJet 4 printer, these soft symbol sets can be used with scalable typefaces just like any other symbol set.

The BUILDSYM utility discussed in this chapter is provided for developers to use in creating downloadable symbol sets for the LaserJet IIIP/4 printers.



For those wishing to bypass the use of BUILDSYM, the PCL 5 printer language commands that support User-Defined Symbol Sets are discussed in detail in the *PCL 5 Printer Language Technical Reference Manual*.

The BUILDSYM Kit

The following BUILDSYM files are included on one of the HP-supplied disks:

A2BIN.C	Converts the file CG-MSL.EXH from ASCII to a binary format
A2BIN.EXE	The executable ASCII-to-binary converter
A2BIN.MAK	Make file for A2BIN.EXE
BUILDSYM.C	Reads the .SYM file and generates the PCL code for a downloadable symbol set
BUILDSYM.EXE	The executable downloadable symbol set generator utility
BUILDSYM.MAK	Make file for BUILDSYM.EXE

CG_HP.DOC	List of CG numbers and their corresponding HP_MSL numbers
HP_CG.DOC	List of HP_MSL numbers and their corresponding CG numbers
CG2HP.C	Converts a .SYM file that has Agfa (CG) numbers to one containing HP Master Symbol List (HP MSL) numbers
CG2HP.EXE	The executable CG-to-HP MSL symbol file converter
CG2HP.MAK	Make file for CG2HP.EXE
CG-MSL.BIN	The binary version of CG-MSL.EXH read in by READCGMap in MAPPING.C
CG-MSL.EXH	An ASCII table of CG numbers and their corresponding HP numbers
MAPPING.C	Reads in CG-MSL.BIN and stores it into an array
TD???.SYM	Type Director Symbol Set files

How Does BUILDSYM Work?

BUILDSYM inputs a symbol set definition file (.SYM file) which contains symbol set information and outputs PCL code that allows the symbol set to be downloaded to the HP LaserJet IIIP/4 printers. Symbol set definition files may be created by modifying existing Type Director symbol set definition files, such as the TDxx.SYM files located on one of the HP-supplied disks, or in the Type Director \TD\SYMBOLS directory. They may also be created “from scratch.” Either way, to create a symbol set definition file, follow the guidelines in *Creating a Symbol Set Definition File* later in this appendix.

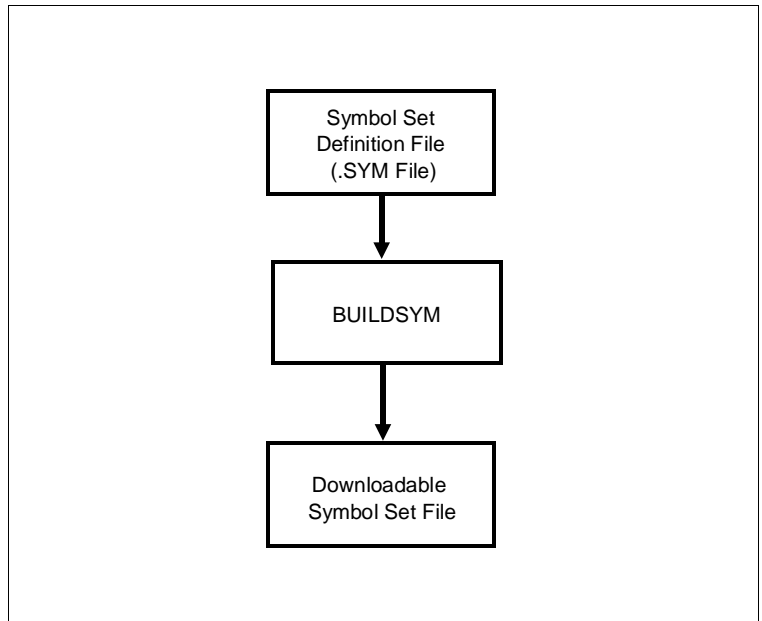


Figure D-1. BUILDSYM Operation

Using the Correct Character Numbers

Character location numbers within the symbol set definition file can be specified using one of two character numbering schemes:

- Agfa (CG) numbers
- HP MSL numbers

If you are modifying a Type Director symbol set, which contains Agfa character numbers, you will probably want to use Agfa numbers in the .SYM file you are creating. On the other hand, if you are starting from scratch, you may want to use the HP MSL numbers or convert existing Agfa numbers to HP MSL numbers.

If you are using Agfa character numbers (CG numbers), you must either convert the Agfa character numbers to HP MSL numbers using the CG2HP.EXE utility, *or* the file CG-MSL.BIN file must be resident in the current directory when BUILDSYM is run, which allows BUILDSYM to convert the Agfa numbers to HP MSL numbers. All download-

able symbol sets must contain HP MSL numbers in order to be used by the printer. The section below explains how to convert to HP MSL numbers using the CG2HP.EXE utility.

If your symbol set definition file (.SYM) file contains HP MSL character numbers, the character numbers will not require any conversion. The BUILDSYM utility will use the numbers as they are.

Converting to MSL Numbers

As mentioned above, if the .SYM file you are using contains Agfa numbers (as do symbol set files from Type Director), BUILDSYM automatically converts these to HP MSL numbers, as long as the CG-MSL.BIN file is resident in the current directory when BUILDSYM is run.

An alternative method of converting character numbers to HP MSL format is the CG2HP.EXE utility provided with the BUILDSYM kit. This utility is helpful to those developers that only wish to modify a few characters in an existing Type Director symbol set.

CG2HP.EXE Syntax

To convert character numbers in a symbol set from Agfa numbers to HP MSL numbers, the CG2HP.EXE utility is used as follows:

```
CG2HP [ infile [ outfile ] ]
```

infile The *infile* parameter is the .SYM file containing Agfa numbers.

outfile The *outfile* parameter is the .SYM file that will be written containing the HP MSL numbers.

Note



The *outfile* name must be different than the *infile* name.

Using BUILDSYM

The command format for the BUILDSYM utility is listed below:

```
BUILDSYM [ infile [ outfile ] ]
```

infile The *infile* parameter is the input .SYM file (symbol set definition file). Creating a symbol set definition file is discussed in detail later in this appendix under “Creating a Symbol Set Definition (.SYM) File.”

When using a .SYM file that does not contain HP MSL character numbers, the file CG-MSL.BIN must be in the current directory so that the Agfa character numbers may be converted to HP MSL numbers.

outfile The *outfile* parameter indicates the name of the PCL symbol set file that BUILDSYM creates. Any legal DOS file name is acceptable. For example, the file name could be in the format FILExx.DSS, where xx represents the two-character selection value and DSS represents downloadable symbol set. Using this sample format, the Roman-8 downloadable symbol set file would be named FILER8.DSS.

Example: Using BUILDSYM

A sample command line using BUILDSYM is as follows:

```
BUILDSYM TDxy.SYM OUTPUTxy.DSS
```

Where TDxy.SYM is the symbol set definition file and OUTPUTxy.DSS is the downloadable symbol set created with BUILDSYM.

Creating a Symbol Set Definition File (.SYM Files)

This section describes the format of the symbol set definition files. It serves as a reference for when you modify existing Type Director .SYM files or if you wish to create a .SYM file from scratch.

Symbol set definition files consist of several keyword parameters, such as */type*, */index*, */PCL char*, etc., each describing a particular feature of the .SYM file (see Table D-1 below). The .SYM files are formatted as ASCII files and have the following characteristics:

- Each parameter must appear on a separate line.
- The parameters appear in the form:
/parameter name = parameter value.
- At least one space must separate the equal sign (=) from the parameter value and the parameter name.
- All text on a line after a semicolon (;) is ignored and may be used for comments.
- The parameter values are not case-sensitive (except for */PCL char*).

Table D-1 lists the symbol set definition file parameters. The parameters are discussed in more detail following the sample .SYM file on the next page.

Table D-1. Symbol Set Definition Value Parameters

Parameter	Default	Range	Comments
<i>/PCL num =</i>	0	0 - 63	Optional
<i>/PCL char =</i>	@	ASCII 65 - 86	Optional
<i>/type =</i>	2	0,1,2	Optional
<i>/index =</i>	AGFA CG Nos.	HPMSL or <i>no entry</i>	Optional
<i>/first code =</i>	0	0 - 255	Optional
<i>/last code =</i>	255	0 - 255	Optional
<i>/requirements =</i>	0x0000000000000000	See Description	Optional
<i>/symbols =</i>	N/A	N/A	Required
<i>/end =</i>	N/A	N/A	Required

Example:
Sample .SYM File

An example of the symbol definition file for the US ASCII symbol set follows:

```
; symbol definition file for US ASCII
/type = 0          ; characters 32-127 are printable
/index = HPMSL    ; character values given in HP MSL order
/PCL num = 0      ; 0U is the symbol set selection parameter
/PCL char = U     ; 0U is the symbol set selection parameter
/first code = 32  ; first value defined in symbol table
/last code = 127 ; last value defined in symbol table
/requirements = 0x8000000000000000 ; standard Latin
                                complement required
/symbols =
32 0          ; space code, or printable thin space
33 1          ; exclamation mark
34 2          ; neutral double quote
35 3          ; number sign
36 4          ; dollar sign
...
126 96        ; one wavy line approximate
127 97        ; medium shading character
/end
```

The .SYM Parameters

The following discussion describes the .SYM file parameters in more detail.

/PCL num
/PCL char The */PCL num* and */PCL char* parameters are the PCL values that will be used to select the symbol set once it is downloaded. For example, the PCL selection value for the Roman-8 symbol set is 8U, so the TDR8.SYM file (Roman-8 symbol set definition file) will contain the parameters */PCL num = 8* and */PCL char = U*.

The valid range of values for */PCL num* is 0 - 63. The valid range of values for */PCL char* is A - V (ASCII 65 - 86)—this is the only case-sensitive parameter in a .SYM file.

/type The type parameter determines which characters in the symbol set are printable:

Value	Symbol Set Type
0	7-bit (characters 32 - 127 decimal are printable)
1	8-bit (characters 32 - 127 and 160 to 255 decimal are printable)
2	PC-8 (All character codes are printable except 0, 7, -15, and 27 decimal)

/index The */index* parameter is a flag stating whether the values in the following symbol table are Agfa (CG) character numbers or Hewlett-Packard Master Symbol List (HP MSL) character numbers.

- If the value of */index* is *hpmsl*, then the numbers that follow are interpreted as HP MSL character numbers.
- If the value of */index* is anything other than *hpmsl*, or it is not defined in the .SYM file, then the numbers that follow are interpreted as Agfa character numbers. If the value for */index* is not *hpmsl*, the CG-MSL.BIN file must be resident in the current directory when BUILDSYM is run.

/first code The */first code* parameter is the decimal value of the first defined symbol of the symbol set. The value range is 0 to 255.

/last code The */last code* parameter is the decimal value of the last defined symbol of the symbol set. The value range is 0 to 255.



The */last code* value must be greater than or equal to */first code*.

/requirements The */requirements* parameter is a 16-digit hexadecimal (64 bit) value that limits the type of fonts that are compatible with the defined symbol set. If this value is 0, the character set can be used with any character complement. The bits defined as of the publication date are:

Bit set to 1

Character set requires:

63	Latin complement (as Roman-8, ECMA 94 Latin 1, ASCII, etc.)
62	Eastern European Latin characters (as ECMA 94 Latin 2)
61	Turkish characters
57	Cyrillic characters for Russian, Bulgarian, Ukrainian, Byelorussian, Macedonian and Serbo-Croatian
54	Arabic characters
51	Greek characters
48	Hebrew characters
34	Math characters of Math-8, PS Math and Ventura Math sets
33	Semi-graphic characters of the PC-8 and PC-8 D/N sets
32	ITC Zapf Dingbats complement

An example of values for the */requirements* fields of a PC-8-like symbol set would be as follows:

0x8000000200000000 (character set requires both Latin and PC-semi-graphic characters).

/symbols

The */symbols* parameter indicates the start of the symbol table. The symbol table is used to map the character codes (of the symbol set being defined) to their character numbers. Within the symbol table, each line represents a symbol in the symbol set. The table may have up to 256 entries whose defined values are from 0 to 255. The format of the symbol table is as follows:

/symbols =

<decimal table entry number, 0-255> <Agfa CG or HP MSL number, as defined by */index*>

...

...

<decimal table entry number, 0-255> <Agfa CG or HP MSL number, as defined by */index*>

/end

/end

The */end* parameter indicates the end of the symbol table and must be the last line in the file.

Contents

Overview	E-1
The FASST Kit	E-1
File Formats	E-2
FASST Shell Usage	E-3
FASST Integration	E-5
Program Structure	E-5
Comparing the Compression Modes	E-5
Major Functions	E-6
pclwrite()	E-6
mode_0_PCL()	E-7
mode_0_2_PCL()	E-7
mode_0_2_3_PCL()	E-7
CompressLJ3()	E-7
Ergs_PCL()	E-7
CompressErgs()	E-8
choose_mode()	E-8
Mode3Out()	E-8
Mode2Out()	E-9
Modifying the Code	E-9
FASST Data Size	E-10
FASST Code Size	E-10
Testing Data	E-11
Testing Results	E-12

Overview

The FASST (Future Adaptive Smart Switching Technique) utility is designed to help application programmers optimize the various compression methods that are available on HP LaserJet printers. The utility uses an intelligent algorithm to decide a good compression method for each row of raster data.

The FASST Shell (FASST.EXE) is supplied as an example of how a software application would interface to the intelligent compression routines in the FASST code. The shell reads an uncompressed PCL raster data file, converts it to a binary raster file stripped of the PCL commands, and passes it to the compression routine in 64K blocks.

The FASST Kit

The following files are included on the HP-supplied disks:

BINS.ZIP	Graphics files used for testing (compressed using PK-ZIP)
COMP.C	Contains decision algorithm and compression routines
FASST.C	Sample shell program to show how to call library routine
FASST.EXE	Compiled shell program
FASST.MAK	Make file for the example program
LISVERGS.C	Compression routine specific to adaptive compression (LaserJet III/IIID/IIISi)
MD023PCL.C	Specific routine for printers using modes 2 & 3 (LaserJet III/IIID/IIISi)
MD02PCL.C	Specific routine for printers using mode 2 (LaserJet IIP)

MD0PCL.C	Specific routine for printers not using graphic compression
MODE2.ASM	Assembly code for mode 2 compression
PCL2BIN.C	Converts mode 0 PCL to .BIN format
PCL2BIN.EXE	Compiled PCL-to-.BIN converter
PCLWRITE.C	Chooses compression routines based on printer model
RAST2PCL.H	Include file for FASST routines

File Formats

The FASST shell reads a binary raster file that is created using PCL2BIN.EXE. This raster file has the form:

```
<row length> <raster data> <row length> <raster data> ...
<row length> <raster data>
```

The <row length> is a word with the high-order byte first. The <raster data> is a row of data that is <row length> bytes long. If <row length> is 0, it is immediately followed by another <row length>. As the FASST shell reads in each 64K block of the file, it is stored in memory in the same format as it is on disk. Additionally, 0xFFFF is appended to the end of the data after the last raster row to indicate the end of the block.

The .BIN format described above was chosen since each software developer has a different method of storing raster data. If you wish to use the FASST shell to compress raster graphics, you must either convert the graphics into the .BIN form using PCL2BIN.EXE or write your own utility to convert from your file format to the .BIN format.

For example, to convert a .TIF file to the .BIN format, one must first convert the .TIF file to an uncompressed PCL file using one of many available file conversion programs. The resulting PCL file can be converted to a .BIN file by running PCL2BIN.EXE as follows (see Figure E-1):

PCL2BIN GRAPHIC.PCL GRAPHIC.BIN

PCL2BIN.EXE can also be run without command line arguments, in which case it will prompt the user for file names.

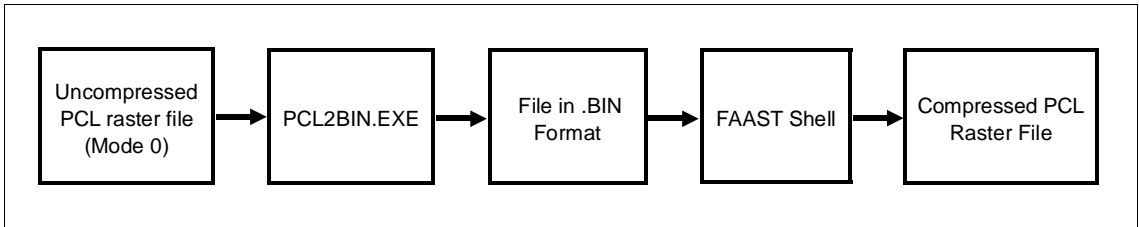


Figure E-1. Using the FASST Shell

FASST Shell Usage

The FASST shell can be used in two ways: interactively or through the command line. To use FASST in the interactive mode, simple start FASST.EXE without any command line arguments, for example:

FASST

The command line arguments for the shell are in the following form:

FASST [*input filename*][*output filename*][*printer mode*]]

Where:

Input filename is the file name of the .BIN file.

Output filename can be either a disk file or a device such as LPT1. If you do choose a device such as LPT1 as an output file, leave off the trailing colon (that is, use LPT1 instead of LPT1:).

Printer model is the 1- to 4-character abbreviation for the target printer. The printer abbreviations are as shown in the following table.

Table E-1. Abbreviations for the Printer Model

Abbreviation	Printer
I	LaserJet
I+	LaserJet PLUS
500	LaserJet 500 PLUS
II	LaserJet Series II
IIP	LaserJet IIP
IID	LaserJet IID
III	LaserJet III
IIID	LaserJet IIID
ELI	LaserJet IIISi
ROY	LaserJet IIIP
ROY	LaserJet 4

If any of the command-line options are invalid or not entered, the FASST shell will interactively prompt the user for the data. For example, if the input file does not exist, the shell will prompt the user for another input filename.

Note 

The FASST shell can be aborted at any time by pressing CTRL-BREAK.

FASST Integration

The following paragraphs are written to assist developers in integrating FASST into their own applications. Each developer will probably desire to customize the FASST code to accommodate their own method of processing raster graphics. This guide explains the code in enough detail to facilitate code modifications.

Program Structure

Two independent programs are provided in this utility.

- PCL2BIN.EXE converts an uncompressed PCL raster graphics file into the .BIN raster format.
- FASST.EXE is a sample application that calls the intelligent compression routines.

The PCL2BIN.EXE program operates as discussed in the “File Formats” discussion at the beginning of this appendix. The discussion below summarizes the differences between the various compression modes, followed by the operation of the intelligent compression algorithms in the FASST shell.

Comparing the Compression Modes

Compression Mode 0

In compression mode 0, no compression takes place. Each bit of raster data exactly corresponds to a dot on the printed page. In mode 0, it takes approximately 320 bytes to send a full 8.5-inch row of 300 dpi raster data.

Compression Mode 1

Compression mode 1 is also called Run Length Encoding. Data is sent in byte pairs. The first byte is a repetition count for the data in the second byte. This mode works well for long strings of repeated bytes, but if the data has several literal (non-repeating) bytes, then the compressed size is likely to be larger than the uncompressed size.

Compression Mode 2

Compression mode 2 is also known as TIFF (Tagged Image File Format). The TIFF mode uses a control byte that is sent before the raster data. This control byte indicates if the following data is a repeating run, or a literal run. In this way, mode 2 combines the features of modes 0 and 1.

Compression Mode 3

Compression mode 3, also called Delta Row Compression, samples the current row of data and outputs to the printer only those bytes that are different from the previous row. Mode 3 compression is best-suited to compressing vertical patterns that are repeated between rows.

Compression Mode 5

This mode is also called Adaptive Compression. Mode 5 is a block encapsulation of all the previous modes. This mode uses an escape sequence to notify the printer that a block of up to 32K of data is coming. There is a control byte on each line of data which tells the printer in which compression mode the current line is printed. There are also control bytes to tell the printer to repeat the last line “n” times and to print “n” blank rows.

Major Functions

The major functions of the FASST shell are described below:

pclwrite()

pclwrite() is the root function of the intelligent compression routines. This function simply calls the appropriate compression routine based on the printer model chosen. *pclwrite()* calls *mode_0_pcl()*, *mode_0_2_PCL()*, *mode_0_2_3_PCL()* or *Ergs_PCL()*, depending on what the target printer is.

- mode_0_PCL()** This function is for all LaserJet printers that do not support raster compression, specifically the HP LaserJet, LaserJet PLUS, LaserJet 500 PLUS, LaserJet series II, LaserJet IID, and LaserJet 2000 printers.
- This function simply wraps PCL code around the raster data that is passed to it.
- mode_0_2_PCL()** This function is only for the LaserJet IIP printer, which supports compression modes 0, 1, and 2. Hewlett-Packard has found that in most cases, mode 2 compression yields better results than mode 1. Therefore, mode 1 is not included in the FASST routines. Additionally, HP has found that mode 2 compression is usually better than no compression. Therefore, as this function looks at each row, it performs a mode 2 compression, and then compares it to the uncompressed row size. The function then writes the PCL code for the mode with the best result.
- mode_0_2_3_PCL()** The HP LaserJet III, IIID, and IIISi printers support compression modes 0 through 3. This routine prepares any variable length rows (if used) and calls *compressLJ3()* to actually compress and output the raster data.
- CompressLJ3()** After being called by *mode_0_2_3_PCL()*, *CompressLJ3()* will call *choose_mode()* to choose the best compression method for this row. Then it compresses the data and outputs it to the specified stream.
- Ergs_PCL()** The LaserJet IIP and LaserJet 4 printers support adaptive compression (mode 5 compression), which reduces raster data size by decreasing the amount of PCL code that is needed to send it to the printer. The LaserJet IIP/4 printers also perform decompression on-the-fly to reduce printer memory requirements for large graphic images. This function is very similar to *mode_0_2_3_PCL()* except that it writes out mode 5 PCL code for each raster row. Also note that ERGS sends data in a raster block, which is limited in

size to 32K, so this function will automatically split the input into multiple blocks if it exceeds 32K. Keep in mind that adaptive compression will work best with output blocks being as close to 32K as possible. Therefore, you should pass *Ergs_PCL()* raster data in chunks as large as possible so that when it calls *OutputMode5Block()* to output the data, it will be sending large blocks.

CompressErgs()

In this function, *choose_mode()* decides which compression mode will best compress a given line of data. Then the control bytes are inserted, followed by the data in the memory area passed in the function call. The data is not sent out to the printer at this time because this function tries to buffer a block that is close to 32K to maximize efficiency for the LaserJet IIIP/4 printers. *Ergs_PCL()* will output the data when the buffer is full.

choose_mode()

This function uses a heuristic algorithm to choose either mode 0, 2, or 3 as the best compression method for each row. Hewlett-Packard's tests show that *choose_mode()* selects the optimal compression mode almost every time. The algorithm essentially counts the number of bytes that are duplicated from the previous row, and the number of bytes that are repeated within the row. The bytes that are duplicated from the previous row are best for mode 3 compression. Bytes that are repeated within the row are good for mode 2 compression.

Testing has shown that if either of these values is greater than one-third of the uncompressed row size, compression is worthwhile. This function picks the mode whose byte count exceeds the threshold. If both counters exceed the threshold, the larger is chosen.

Mode3Out()

The actual mode 3 compression of row data is done in this function. Additionally, the current row's data is copied into the memory location of the last row's data so that the calling program doesn't have to do this.

Mode2Out()

Mode 2 compression is performed in this function. Since the last row is not passed to this function, it does not copy the current row into the last row.

Modifying the Code

Since many applications use a great deal of available memory, the FASST routines are designed to accept data in 64K blocks. If your application can access enough free memory to use larger blocks, simply change the value of *MAX_BLOCK* in RAST2PCL.H. However, in C you cannot address a data array larger than 64K with a far pointer, so if you do increase *MAX_BLOCK* to more than 64K, you need to use huge pointers for the data. To do this, change the definition of *PBYTE* in RAST2PCL.H from a “far” pointer to a “huge” pointer.

If your applications will be calling *pclwrite()* with variable length data rows, then you should make sure you have set *RAGGEDROWS* in RAST2PCL.H to 1 (true). Otherwise, set *RAGGEDROWS* to 0 (false).

Included with the FASST utility is a file called *MODE2.ASM* which does a mode 2 compression. This can be integrated into FASST if additional speed is desired.

FASST Data Size

The following table shows the amount of additional memory allocated by the FASST code.

Compression Method	Ragged Rows	Uniform Length Rows
ERGS 0,2 & 3	2 (longest row length) + 32K	longest row length + 32K
0, 2, & 3	3 (longest row length)	2 (longest row length)
0 & 2	longest row length	longest row length
0	none	none

FASST Code Size

The following table shows the expected increase in application code size (based on Microsoft C 6.0), depending on compression method.

Compression Method	Ragged Rows	Uniform Length Rows
ERGS 0, 2, & 3	1050 bytes	538 bytes
0, 2, & 3	1062 bytes	550 bytes
0 & 2	562 bytes	562 bytes
0	530 bytes	530 bytes

Testing Data

The FASST utility has been tested on many different types of graphics files to maximize compression in most cases. The results of some of this testing are discussed below.

Testing Methods

FASST was tested on six different binary graphics files consisting of the following scanned images (these image files are included on one of the HP-supplied disks in the BINS.ZIP file):

- HP.BIN** A small Hewlett-Packard logo (.3 by 1.7 inches).
- CAT.BIN** A full-page with a small photo (2.5 by 3.4 inches) and several small black and gray boxes in different areas on the page. In total, the graphics occupy about 20% of the page, but are spread out so that there is plenty of white space between the photo and the various blocks.
- DESI.BIN** An instruction sheet consisting of a full page of 10-point text, a logo (about 1.25 by 1.5 inches) and a graphic (brush-stroked letters) (about 3.9 by 1 inches).
- SPRD.BIN** A full-page, 4-column spreadsheet, with 1-point rules between each column and between each row (every .2 inches).
- EIN.BIN** A large medium-contrast portrait (7.75 by 8 inches) containing mostly gray shades (not much white or black).
- SELF.BIN** An HP LaserJet self test page containing a small bar chart (2 by 2.5 inches), a small pie chart (2 by 2.4 inches), text, and a border with various shading patterns and gray shades.

For the test, the FASST shell was stripped of the calls to *pclwrite()* to determine the overhead associated with reading each of the six graphics from the disk. The overhead time was subtracted from the test results to see the time spent compressing the graphics. In addition, the test results were written to files on a virtual disk in memory to minimize the time spent writing to a hard disk.

Tests were conducted using an HP Vectra RS/25c (25MHz 386) with 8 Mbytes RAM and a 100-Mbyte hard disk. The HP-supplied version of MS-DOS 4.01 was used, with no drivers loaded except the RAM disk and the SHARE utility. (The SHARE utility is required by HP DOS on large hard-disk partitions.) The RAM disk used was RAMDRIVE.SYS, which is included with Microsoft Windows 3.0.

Testing Results

The following compression statistics show that there is a cost associated with compressing the graphics data. In fact, with the six test cases used, there was an average 39% increase in the amount of time required to prepare a file for adaptive compression over simply encapsulating the image in mode 0 compression (no compression). The cost, however, is minor in comparison to the 70% average reduction in output data size. The following graph illustrates the file sizes that are achieved through compression.

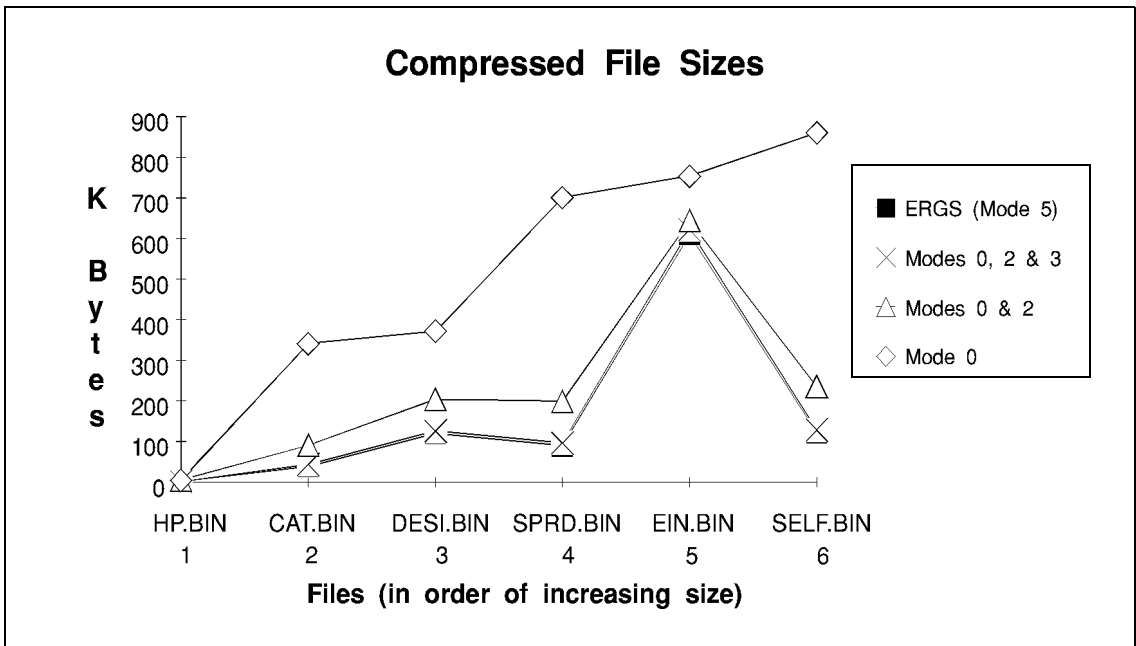


Figure E-2. The Effect of Compression on File Size

The following graph compares the time it takes to compress various graphics files using the different compression methods.

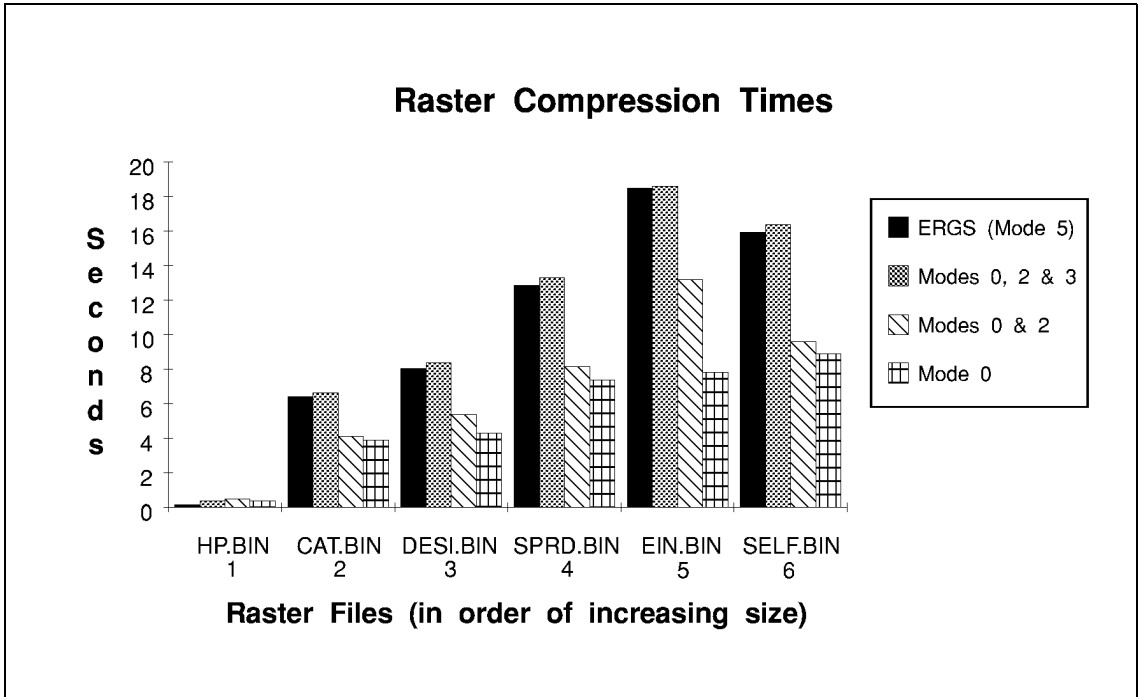


Figure E-3. Comparative Compression Times

Contents

Introduction	F-1
Installing and Running SRTool	F-2
Initialization	F-2
Configuring SRTool Manually	F-3
Automatic Configuration	F-3
The User Interface	F-5
The SRTool Display Windows	F-5
Error Handling	F-6
Using Macros to Send Escape Sequences	F-7
Using the DOS Command Line Within SRTool	F-7
Using Scopy Instead of DOS COPY	F-7
The SRTool Menus	F-8
PCL Status Readback	F-10
PJL Status Readback	F-13
PCL Command Macros	F-14
PJL Command Macros	F-17
Options	F-18
Files	F-18
Setting the Carriage Return	F-23
Break Points	F-23
Checking the Input Buffer	F-24
Data Pacing	F-24
Displaying Data Received from the Printer	F-24
Displaying Data Sent to the Printer	F-24
Clearing the Windows	F-24
Byte Counters	F-25
Displaying Line Status	F-25

Introduction

The status readback tool (SRTool) is a software utility used to query the status of a printer and to receive and record the returned printer status. With SRTool, queries may be made using either the keyboard or an input file.

SRTool Requirements

SRTool runs on IBM-compatible computers and requires 170K of RAM to operate. No special cables are required for communications with the printer.

Chapter Overview

This chapter is organized as follows:

- **Installing and Running SRTool**—This section describes how to install SRTool.
- **Initialization**—The Initialization section explains the parameters that must be set for SRTool to run, and the two ways in which they may be set.
- **The User Interface**—The User Interface section describes the windows that appear on the screen during execution and explains how the pull-down menus at the top of the screen are used.
- **PCL Status Readback**—The PCL Status Readback section lists the PCL commands that are implemented in SRTool and gives a brief description of each.
- **PJL Status Readback**—This section lists the PJL Status Readback commands that are implemented in SRTool and briefly describes each command.
- **PCL Command Macros**—The PCL Command Macros section lists and gives a brief description of the PCL command macros and how they are used in SRTool.
- **Options**—The Options section describes the choices found in the SRTool *Options* menu.

Installing and Running SRTool

Follow the instructions below to install and run SRTool.

Installing SRTool

1. Create a directory on your hard disk (such as `C:\SRTool`).
2. Copy all of the files from the SRTool disk into the new directory (for example, `COPY A:*.* C:\SRTOOL`).

Running SRTool

The SRTool program file is `SRTOOL.EXE`. Run SRTool by getting into the directory where SRTool is located, typing `SRTOOL`, and pressing the Enter key. See the “Initialization” discussion below to configure SRTool to run on your system.

Initialization

Before SRTool may begin communications with the printer, it requires the following information:

- Interface type (serial, bidirectional parallel, or standard parallel)
- Port number
- Baud rate (serial interfaces only)
- Flow control (serial interfaces only)

You have a choice of entering the configuration information each time you load SRTool, or you may use a setup file for automatic configuration. Both options are described in the following paragraphs.

Configuring SRTool Manually

When you run SRTool without a setup file (SETUP.SRT), the program immediately enters an initialization phase upon execution. A series of configuration menus prompts you for the options listed above. Configure the tool by making a selection in each menu that appears.

Note



When configuring I/O Type, select AFPP when using the Bi-Tronics bidirectional parallel interface. If you wish to use status readback, you must select AFPP as the I/O type. You do not need a special cable or interface card in your computer. The BiTronics port in the printer operates well with a standard parallel interface card in the host and a standard parallel cable.

Automatic Configuration

To avoid going through the configuration menus, you can configure SRTool automatically using a setup file (SETUP.SRT). Each time you load SRTool, it automatically looks for SETUP.SRT either in the directory from which you run SRTool or in the directory specified in your AUTOEXEC.BAT file. (You may run SRTool from any directory by adding to your PATH statement the directory where SRTOOL.EXE is located, and by adding the following line to your AUTOEXEC.BAT file:

```
SET SRTDIR=<path to SETUP.SRT file>
```

For example: SET SRTDIR=C:\SRTOOL

The SETUP.SRT file doesn't exist until you create one. This file is required in order to automatically configure SRTool. The SETUP.SRT file format is structured as follows:

<Mouse Support>

<I/O Type>

<Port Number>

<Baud Rate>

<Hardware Flow Control>

<Buffer Size>

The possible values for each option are listed in the table below:

Option	Value	Description
Mouse Support	0	No Mouse Support
	1	Mouse Support
I/O Type	0	BiTronics Parallel
	1	Serial
	2	Parallel (non-BiTronics)
Port Number	1 to 3	For Parallel I/Os
	1 to 4	For Serial I/Os
Baud Rate	2	300
	3	600
	4	1200
	5	2400
	6	4800
	7	9600
	8	19200
	9	38400
	10	57600
Hardware Flow Control (Serial Only)	0	No Flow Control
	1	Flow Control
Buffer Size (Serial Only)	Desired Buffer Size	

Note



SRTool always sets the parity to none, number of data bits to eight, and number of stop bits to one.

Creating the SETUP.SRT File

Create the SETUP.SRT file by typing the value for each option, followed by a carriage return (<CR>) and line feed (<LF>). For example, the following ASCII file can be used to configure SRTool for mouse support and bidirectional parallel I/O on LPT1:

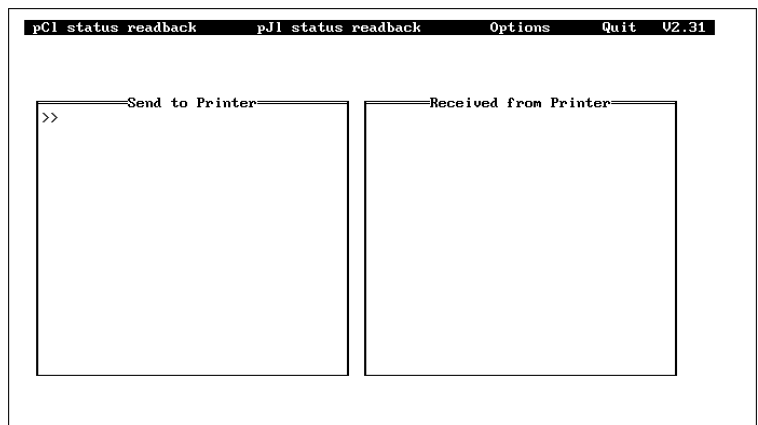
```
1<CR><LF>
0<CR><LF>
1 <CR><LF>
```

The User Interface

The SRTool Display Windows

While SRTool is running, two windows are continually displayed on the screen. The left window, labeled “Send to Printer,” is the input window. This window is used to type data that you want sent to the printer. The input window prompt is “>>”.

The right window, labeled “Received from Printer,” is the window in which the data received from the printer is displayed.



Every character that is sent to or received from the printer is displayed in one of these windows, unless the data is sent to the printer as a result of a DOS command. Control characters are displayed in a readable format (that is, a line feed is displayed as <LF>, spaces are displayed as <SP>, and backspaces as <BS> so that you are able to differentiate between blank space and actual spaces that are sent or received). Any blank space in either of the windows appears after a carriage return, line feed, or form feed, or when there is not enough room on a line for a control character. Blank space does not represent data sent to or received from the printer.

Note



If SRTool receives a character from the printer that it doesn't recognize, it displays the character's ASCII code in hexadecimal.

Characters sent to the printer from an input file, and escape sequences chosen from a pull-down menu are displayed in the input window.

Characters are sent to the printer as soon as they are typed; they are not buffered. You cannot, therefore, backspace over a typing error. If the backspace key is pressed, a <BS> is displayed on the screen and a backspace character is sent to the printer. This does not apply to PCL macros or DOS commands.

Error Handling

If an error occurs during execution, a window appears in the center of the screen, briefly explains the error, and tells you to press a key to continue. If the error is fatal (that is, if the port cannot be initialized) the tool is exited after you press any key. If the error is not fatal, the tool resumes execution after a key is pressed.

Using Macros to Send Escape Sequences

To help you send escape sequences to the printer, several PCL macros have been implemented. The PCL macros are used by typing a dot as the first character in the line, followed by the macro name, and ending with a carriage return. (see “PCL Command Macros” later in this chapter).

The macro is erased from the input window when you press Enter and is replaced with its escape sequence. (If you would like to send a period character (.) to the printer, place two periods in succession. Only one of them will be sent to the printer.) PCL macros can also be run from an input file (see the “Input Files” portion of the Options menu discussion).

Using the DOS Command Line from Within SRTool

SRTool provides you with the capability of running DOS commands from inside the tool. To execute a DOS command without exiting SRTool, type a dollar sign followed by the DOS command (at the input window prompt), and press the Enter key. For example:

```
$COPY C:\TEST.DOC PRN:
```

When you run a DOS command this way, the screen clears first and then the command is executed. You are then prompted to press a key to return to the tool. After you press a key, the SRTool display is restored.

Using Scopy Instead of the DOS COPY Command

If you use the DOS COPY command to download a font, macro or other type of file that is large enough to fill the printer’s buffer, or if the printer is off-line, the tool displays an error message and prompts you to press a key to continue. SRTool has its own copy command, *scopy*, that eliminates the need to monitor the tools progress. *Scopy* performs the same function as the COPY command, but unlike COPY, it waits for the printer to become ready for data and then continues to send data, without prompting you to press a key to continue. The syntax for *scopy* is:

```
$scopy <filename>
```

Note

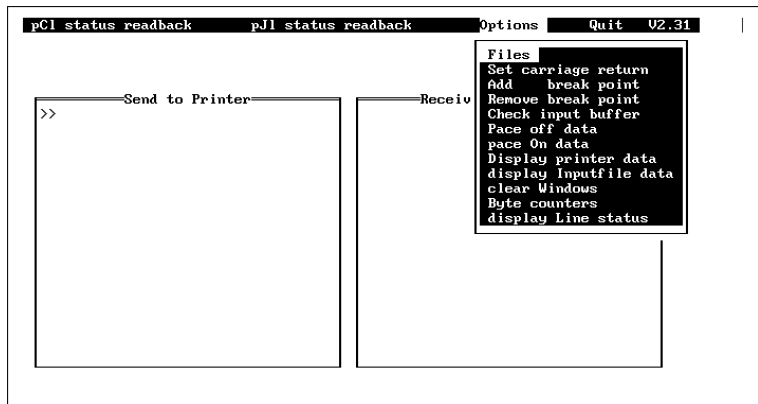


Scopy should only be used for serial transfer. It can be used for parallel transfer, but the parallel transfer should be fast enough that the buffer does not overflow.

Scopy performs a limited parsing of the input file; a carriage return is added to each line and any leading white space is also removed. If you do not wish to have your input file parsed, use the *copy2* command in place of *scopy*.

The SRTool Menus

A pull-down menu is displayed at the top of the SRTool screen. The menu may be accessed by pressing either F10 or the Alt key and pressing the capitalized letter in the menu you wish to select. (For example, Alt-O for the Options menu.) Alternatively, you can click the left mouse button on the chosen menu selection.



To move the highlight bar to a menu choice, you can use either your arrow keys or press the capitalized letter in the choice. To select a menu choice, first move the highlight bar to the item you wish to select, then press the Enter key. To leave the pull-down menu and return to your input window, press the Esc key.

Selecting with the Mouse

You can also use a mouse to select menu items. Highlight an item by clicking the left button on any item. Select the item by double-clicking on the item.

The PCL Status Readback menu allows you to send PCL status readback commands to the printer. When selecting an item, you are always given a choice between all possibilities or a range if a value is required in a choice.

Likewise, the PJI Status Readback menu contains selected PJI status readback commands. If a value is required when you select a command, you are always given a choice between all possibilities or a range of values.

The choices in the Options menu include file management, specifying the characters sent during a carriage return, break points, options to toggle the displaying of data, settings to tell the computer to ignore the data coming from the printer or to tell the printer to quit sending data, and counters to keep track of the number of bytes sent to and received from the printer.

Exiting SRTool

When you want to exit SRTool, choose *Quit* from the menu bar.

PCL Status Readback

The PCL Status Readback menu offers a choice of using pre-defined escape sequences or user-defined escape sequences. The pre-defined escape sequences are:

Free Memory Space ($E_C*s#M$)

Description	Returns the current amount of free memory in the device (in bytes)
Value (#)	Internal Memory Unit
Default Value	1
Range	1

Set Status Readback Location Type ($E_C*s#T$)

Description	Sets the status readback location type to the specified value.	
Value (#)	0	Invalid Location
	1	Currently Selected
	2	All Locations
	3	Internal
	4	Downloaded
	5	Cartridge
	7	User-Installable ROM
Default Value	0	
Range	0 to 5, 7	

Set Status Readback Location Unit (E_C*s#U)

Description	Sets the status readback location unit to the specified value.	
Value (#)	0	All Locations
	1 to 32767	Unit Number
Default Value	0	
Range	0 to 32767	

Inquire Status Readback Entity (E_C*s#I)

Description	Returns status information for all entities of the specified type in the current status readback location.	
Value (#)	0	Font
	1	Macro
	2	User-Defined Pattern
	3	Symbol Set
Default Value	N/A	
Range	0 to 3	

Echo (E_C*s#X)

Description	Echoes the parameter value back to the host.
Value (#)	Echo Value
Default Value	0
Range	-32767 to 32767

User-Defined Escape Sequences

You may wish to have the escape sequences that you use most frequently available in a pull-down menu. The user-defined escape sequences option allows you to specify those sequences and save them so that you can reuse them without re-typing them. Once you create the user-defined escape sequence, you need only select the sequence you want to send from the pull-down menu, instead of typing it again.

Creating a User-Defined Escape Sequence

To specify an escape sequence, select the *edit* option under *user defined escape sequences*. A window appears in which the escape sequences you have specified are listed. You can edit an existing sequence by using the arrow keys to move to the desired sequence and then making the necessary changes.

To add a new sequence, use the down arrow key to move to the first blank line in the window, then type in the new sequence. You can add unprintable characters using the Alt key and the ASCII number on the numeric control panel. For example, use Alt-27 for the escape character.

Editing is terminated when you press the Enter key. SRTool allows you to specify up to ten escape sequences. Each sequence can contain as many as thirty characters.

Note



SRTool displays most of the control characters in user-defined sequences as blank spaces. The escape character is displayed as a left-pointing arrow.

Sending a User-Defined Escape Sequence

To send a sequence that you have added to the list, select *send*. A menu containing all of your sequences appears. Use the arrow keys to move to the desired sequence, then press the Enter key.

The escape sequences that you define are saved to a file called `USRDEF.SEQ` when you exit the tool. If you have set the `SRTDIR` in your `AUTOEXEC.BAT` file, the escape

sequences are stored in that specified directory; otherwise, they are stored in your local directory. Each time SRTTool begins execution it reads this file so that you have access to those escape sequences without retyping them.

PJL Status Readback

The following PJL commands are implemented in SRTTool:

- UEL
- ENTER
- RESET
- INITIALIZE
- SET
- DEFAULT
- INQUIRE
- DINQUIRE
- INFO
- USTATUSOFF
- STMSG
- USTATUS

When any one of the above commands is selected, with the exception of the UEL command, the following characters are sent to the printer:

`<LF>@PJL<SP>`, then the command, then `<LF>`.

For example, for the USTATUSOFF command, the following is sent to the printer:

`<LF>@PJL<SP>USTATUSOFF<LF>`

The PJL status readback commands are accessed by pressing F10 and moving to the PJL menu, pressing Alt-J, or by clicking the left mouse button on *PJL status readback*. As you add commands by selecting them this way, the left window displays the characters that are actually sent to the

printer. Alternatively, PJJL commands can be sent by typing them in the left window.

PCL Command Macros

This section lists the PCL macros that have been implemented in SRTool. They are used by typing the macro name in the input window (including the period).

Functional Category	Macro Name	Escape Sequence Sent	Description
Font Management	.delete_all_fonts	E_c^*cF	Deletes all fonts.
	.delete_temp_fonts	E_c^*c1F	Deletes all temporary fonts.
Font Selection	.default_fonts	$\text{E}_c(3@E_c)3@$	Defaults the primary and secondary fonts.
	.default_primary	$\text{E}_c(3@$	Defaults the primary font.
	.proportional	$\text{E}_c(s1P$	Selects proportional spacing for the primary font.
	.stroke_weight_bold	$\text{E}_c(s3B$	Selects bold as the primary stroke weight.
	.stroke_weight_light	$\text{E}_c(s-3B$	Selects light as the primary stroke weight.
	.style_italic	$\text{E}_c(s1S$	Selects italic as the primary font style.
Job Control	.reset	E_cE	Resets the printer.
Miscellaneous	.reset_all	$\text{E}_cE\text{E}_c\&f6X\text{E}_c^*cF$	Resets the printer, then clears all the fonts and macros.

Functional Category	Macro Name	Escape Sequence Sent	Description
Page Control	.clear_margins	E_c9	Defaults the horizontal margins.
	.landscape	$\text{E}_c\&l10$	Sets the logical page orientation to landscape.
	.portrait	$\text{E}_c\&l0$	Sets the logical page orientation to portrait.
	.set_66_lines_per_page	$\text{E}_c\&l14c1e7.64c66F$	Sets top margin 0.29" from top of page, vertical spacing to 6.28 lines per inch, and page length to 66 lines per page.
Printing Text	.underline_on	$\text{E}_c\&d\#D$	Enables underlining.
	.underline_off	$\text{E}_c\&d@$	Disables underlining.
Device Control	.flush 0 (Flush all complete pages)	$\text{E}_c\&r0F$	Suspends processing of the input stream until all complete pages currently in the device have been printed.
	.flush 1 (Flush all pages)	$\text{E}_c\&r1F$	Suspends processing of the input stream until all pages, including partial pages, are printed.

Functional Category	Macro Name	Escape Sequence Sent	Description
Status Readback	.mem	E_c^*s1M	Returns the current amount of free memory in the printer (in bytes)
	.inquire_entity #1 #2 #3 (see Values #1, #2, #3 below.)	$\text{E}_c^*s\#1T\text{E}_c^*s\#2U\text{E}_c^*s\#3I$	Sets the status readback location type to value 1, status readback location unit to value 2, and returns status for all entities of type value 3 in the current status readback location.
	Value (#1)	0 (default)	Invalid location
		1	Currently selected
		2	All locations
		4	Downloaded
		5	Cartridge
		6	User-installable ROM device
	Value (#2)	0 (default)	All
		1 to 32767	Unit number
	Value (#3)	0	Font
		1	Macro
		2	User-defined pattern
		3	Symbol Set

PJL Command Macros

The table below contains the PJL macros that have been implemented in SRTool. These macros are used by typing the macro name followed by a carriage return. The macro name should begin in the first column of the line and include the period. For example, to use the .variables macro you would send:

```
.variables<CR>
```

The following table lists the PJL macros supplied with SRTool.

Macro Name	Command Sent	Description
.uel	␣%-12345X	Universal Exit Language (UEL) Command
.config	␣%-12345X@PJL INFO CONFIG<LF>	Returns a list of configuration information
.status	␣%-12345X@PJL INFO USTATUS<LF>	Returns the current printer status
.variables	␣%-12345X@PJL INFO VARIABLES<LF>	Returns the current and possible values of environmental and personality-dependent variables.

Options

The *Options* menu contains the following menu items:

- Files
- Set Carriage Return
- Add Break Point
- Remove Break Point
- Check Input Buffer
- Pace Off Data
- Pace On Data
- Display Printer Data
- Display Inputfile Data
- Clear Windows
- Byte Counters
- Display Line Status

Each menu item is described in the following section.

Files

The choices in the Files menu are:

- Input
- Output
- Record

Each menu choice is described in the following paragraphs.

Input Files

SRTTool allows you to create an input file for testing PJJ and PCL commands. Once your file is created, you can tell SRTTool to read from it by selecting *Input file* from the Options menu. If you wish to send only one file, select *Single*. A window appears at the top of the screen and you are prompted for the file name.

As soon as you enter the filename, SRTTool begins reading from it. If you wish to send several files to the printer, select *Multiple* and enter the path name when requested. You will be given a list of files in that directory; press the Enter key

to select a file name. When you have selected all the files you wish to send, press Enter next to *Begin Sending*, which is the first line in the list of file names. The tool then sends the files you specified to the printer.

Note



When an input file is sent to the printer with either of these options, SRTool assumes an 80-character line and checks for special characters described in the following sections. If you need to send a file with longer lines, such as raster graphics data, and do not want the input lines manipulated in order to find special characters, use *scopy* or *copy2* to send the file.

• **Using Loops in Input Files**

One of the powerful features of input files is that they allow you to use loops. These loops may be nested up to three levels. When creating loops, you have three variables to work with: i, j, and k.

When a variable is referenced it must begin with a “%”. Variables may be used in controlling a loop and may be assigned values outside loops. The format of an assignment statement is:

```
%<variable> = <value>
```

The format of a loop is:

```
^for %<variable><start value>, <stop value> [, step value]
```

```
{loop body}
```

```
^endfor %<variable>
```

If the step value is omitted it defaults to 1.

The *loop body* can consist of DOS commands, PCL macros, and any other data that you want to send to the printer.

- **Using Pauses in Input Files**

If your application is receiving data from the printer so quickly that you can't read it all, you can add a break point to your file. Insert a break point by placing a question mark at the beginning of the line following the point at which you would like SRTool to pause; this question mark is not sent to the printer. (If a question mark is found anywhere else in a line, however, it is sent to the printer.)

When SRTool encounters a question mark at the beginning of a line, it quits sending data to the printer and a window appears at the top of the screen. In this window you are given a choice of browsing the printer window, continuing with the input file, or quitting the input file. To make a selection, press the letter of the choice you would like. If you choose to browse the printer window, you will be able to use the arrow keys to scroll the printer window up and down until you are finished browsing. Then press the Esc key and you will once again be given choices at the top of the screen.

If you choose to continue with the input file, SRTool quits reading the input file and returns you to the Options menu. If you decide later that you want to finish reading this file, select *Input file* from the Options menu. SRTool will ask if you would like to continue with the input file. When you press Enter, the tool begins reading where it left off. If you are finished with the file that is currently open and want to begin reading from a new one, press N to abandon the current file. In response, the tool asks you if you want to open a new one. Press Enter and you will be prompted for the file name. (See the "Break Points" discussion for another way to pause files.)

- **Using DOS Commands Within Input Files**

DOS commands may be invoked from a file by adding a dollar sign (\$) immediately before the command name. The dollar sign should be the first character on the line and a carriage return should follow the command. While executing a DOS command, you will not see a DOS prompt; how

ever, the screen clears while it is executing the DOS command, so you may see a blank screen while the tool is reading an input file. The screen is restored when the command has finished executing.

- **Using PCL Macros Within Input Files**

PCL macros are executed from a file in the same way they are used from the keyboard. Make sure the macro begins the line and that there is a carriage return following it—otherwise the tool has no way of knowing where the macro ends. A period (.) encountered anywhere but the beginning of a line is interpreted as a regular character and sent to the printer.

- **Specifying Output Files From Within Input Files**

Output files can also be specified from an input file. The syntax for this command is:

```
outfile = <filename>
```

All of the data received from the printer after this line has been processed is sent to the specified file until the tool is exited or another output file is named.

- **Comments Within Input Files**

If you wish to include comments in your input file that will not be sent to the printer, begin the line with “.*”. This causes any data on that line to be ignored.

- **Sending Instructions to the Operator**

When running interactive tests, it may be useful to give the operator of the tests instructions via SRTool, rather than using a written procedure. To do this, begin your instructions with the following line:

```
. comment
```

Place your instructions on the lines following the .comment, then end them with the following line:

```
. end_comment
```

For example:

```
.comment  
This is a comment.  
.end_comment
```

All lines between `.comment` and `.end_comment` are displayed in an instruction window within SRTTool. No data is sent to the printer while this window is displayed, although any data that is received from the printer is processed.

When the operator is finished with the instructions, he/she may press any key to continue processing the input file.

Output Files

At some point you will probably want to get a hard copy of the data sent back by the printer in order to verify it. When you want to start saving this data, select *Output* from the *Files* menu. A window appears at the top of the screen and prompts you for a file name. Once you enter a file name, all data received from the printer is sent to the file as well as being displayed on the screen. If an output file has already been specified, you are asked to verify that you want to open another file. If you answer yes, the file that is already open will be closed before opening the specified file.

You also have the option of specifying the format of the data in the file. The *parsed* option converts all unprintable characters to a more readable format. The *raw* option sends the data to the output file exactly as it is received from the printer.

Record Files

It is sometimes helpful to have a record of the data that has been sent to the printer. If you wish to capture data as it is sent to the printer, choose *Record* in the *Files* option. Once you enter the name of the file, all subsequent data that is sent to the printer is recorded in this file until either another record file is specified or the tool is exited.

Setting the Carriage Return

SRTTool allows you to specify the characters that you wish to be sent to the printer when you press the Enter key. The options are:

- Carriage return (default)
- Carriage return/line feed
- Line feed

Break Points

If you want SRTTool to stop sending data to the printer after a particular set of data has been sent, but you don't want it to stop every time you use this input file, you can do this without adding a question mark to the file. Instead, you may specify a string of characters within your file to trigger a break point by choosing *Add break point* in the Options menu. A window appears at the top of the screen and prompts you for the break point. Type in the character sequence after which you want the tool to stop sending data. You can specify up to 5 break points at a time.

Note



If you want the tool to pause after a PCL macro, you may type either the macro name or the escape sequence.

Make sure you do this before you begin reading from the input file. Every time this sequence is sent to the printer, the tool stops sending data to the printer and a screen appears at the top of the screen. You are given the same choices as when a question mark is encountered in your input file:

- Browse the printer window
- Continue the input file
- Quit the input file

You can remove a break point at any time by choosing *Delete break point* in the Options menu.

Checking the Input Buffer

In some situations it may be necessary to fill the buffer that receives data from the printer. To do this you can choose *Check input buffer* from the *Options* menu, then select *No*. The tool then stops reading the data sent by the printer until you exit the tool or command it to start reading again, which is accomplished by selecting *Yes* in the *Check input buffer* option.

Note



The reading of data should only be turned off if absolutely necessary; when the buffer is full, all of the data sent from the printer is lost.

Data Pacing

If you wish to indicate to the printer that you do not want it to send data back to the host, you may lower the DTR line by selecting *Pace Off Data* from the *Options* menu. To subsequently raise the DTR line, select *Pace On Data*.

Displaying Data Received from the Printer

If you are not interested in the data the printer is sending to the host, or if you are saving this data in a file and would like to eliminate the time it takes to display this data on the screen, you may choose to turn the printer window off. To do this, select *Display Printer Data* from the *Options* menu, then select *No*. To resume displaying this data, select *Display Printer Data*, then select *Yes*.

Displaying Data Sent to the Printer

Similarly, it is possible to turn off the displaying of data sent to the printer from an input file. To do this, select *Display Inputfile Data* from the *Options* menu, then select *No*. To again display this data, select *Display Inputfile Data*, then select *Yes*.

Clearing the Windows

If you wish to clear the input window or the printer window, you may select *Clear Windows* from the *Options* menu. Select *Input* to clear the input window, *Printer* to clear the printer window, or *Both* to clear both windows.

Byte Counters

If you would like to know how many bytes of data you have sent or received, you may select *Byte Counters* from the Options menu, then select *On*. A small window is displayed above both the input window and the printer window, displaying the number of bytes sent to and received from the printer. You may turn the counters off by selecting *Byte Counters* and then selecting *Off*. You may reset the counters by selecting *Byte Counters* and then selecting *Reset*.

Displaying Line Status

When running tests it is sometimes helpful to be able to monitor the lines to the printer. To do this, select *Display Line Status* from the *Options* menu, then select *On*. For the BiTronics interface, the SELECT, BUSY, PERROR, and NFAULT lines are monitored. For serial I/O, the DTR line is monitored. The status of the lines is displayed at the top of the screen, directly below the pull-down menu bar. To stop displaying this status, select *Display Line Status, Off*.

Contents

Introduction.	G-1
Pair Kerning	G-2
Sector Kerning	G-3
Track Kerning.	G-6

Introduction

This appendix describes the methods used for kerning characters. The information described here provides more detailed information related to the kerning tags documented in Chapter 6.

There are basically three methods used for kerning text:

- Using data which kerns specific character pairs (pair kerning)
- Reducing the white space between each character by an amount indicated by sector kern data (sector kerning).
- Uniformly reducing the white space between all characters (track kerning)

Each kerning method provides a different result. *Pair kerning* only reduces space between predetermined character pairs. *Sector kerning* analyzes all adjacent characters to calculate and adjust the correct distance between each character. *Track kerning* removes the same amount of space from between all characters.

Pair Kerning

A kern pair table provides convenience and speed when printing text using the default character escapements. Kern pair tables are used to indicate which two character combinations require special handling in order to maintain a consistent appearance while setting type.

Predefined kern pairs do not normally use the maximum kern values possible. Instead they use values which the type designers have set to visually balance the use of the character pairs with the characters that are not kerned.

Users are frequently familiar with the use of kern pairs. Any two characters may form a kern pair; however, for a given language, certain pairs of characters appear more frequently than others. The following table shows some of the most common kern pairs in the English language.

COMMON KERN PAIRS		
AC AL AN AO AT AV AW AY	LA LI LL LO LS LT LV LW	sy st
Av Aw	LY	TA TC TE TO TS TW TY
ac af ao at au av aw ax ay	Ma	T, T.
CA CO CT CY	mu	Ta Te To Tr Tu Tw Ty
Co Ce	NT	VA VO VY
DY	nu	V, V. Va Ve Vo
du	OA OT OV OW OY	WA WO WV WY
ew ex ey	PA PE PO PR	W, W. Wa We Eh Wi Wo Wr
FA FG FO	P, P. Pa Pe Po Pr	wa we w, w.
F, F. Fa Fe Fo Fu	Qu	YA YO YS
GY	RA RO RV RY	Y, Y. Ya Ye Yo
KE KO	ra rc re ro	ya ye yo ys y, y.
ke ko ku	SA ST SY	ZA

Sector Kerning

Sector kern information is assigned by the font designer to aid in maintaining the aesthetic quality of the type design while it is being set with the aid of computers. Sector kern information allows the maximum kern distance to be determined for any two character combinations in the font.

To assign sector kern information, the designer first determines how close adjoining characters should be without kerning. A vertical line is drawn to represent those bounds. The vertical dimension of the font is then broken into sectors. These sectors may be aligned next to each other, or they can be overlapped or separated from each other.

The number of sectors and their break up of the character is determined by the font designer. For the example shown in Figure G-1, five sectors are defined. The font designer determines how close adjoining characters may come to the target character for each of the sectors. The distance from the vertical reference line to the maximum kern line is measured in design units, and that value is assigned to the sector.

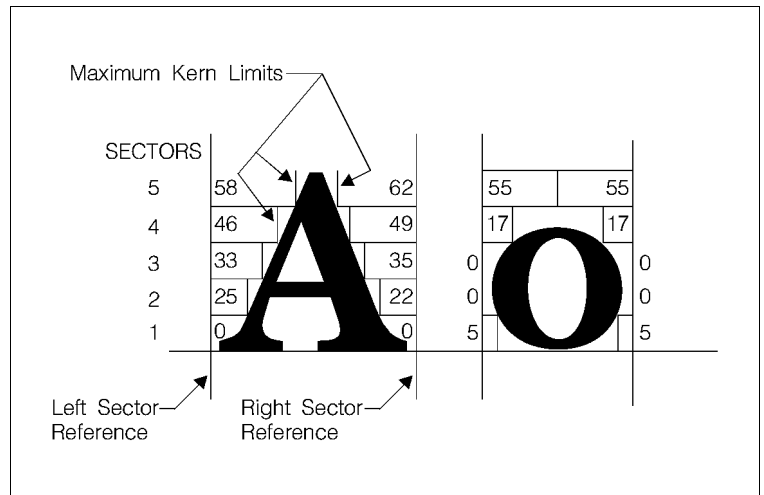


Figure G-1. Maximum Kern Limits

To determine how close two characters may come, the values of adjoining sectors are added and the results are compared. The smallest sum represents the maximum extent to which the two characters may be kerned. Figure G-2 shows the sectors defined for the two characters T and A.

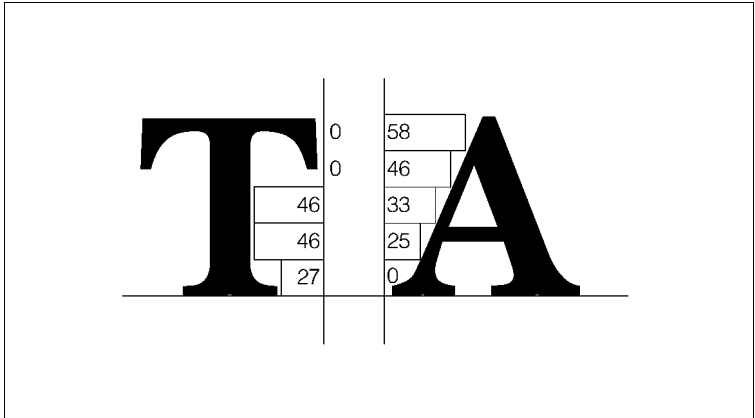


Figure G-2. The Sectors Defined for Letters T and A

If A is set relative to T using only the horizontal escape-ment value of the T (135), then the result would be that shown in Figure G-3 with no kerning.

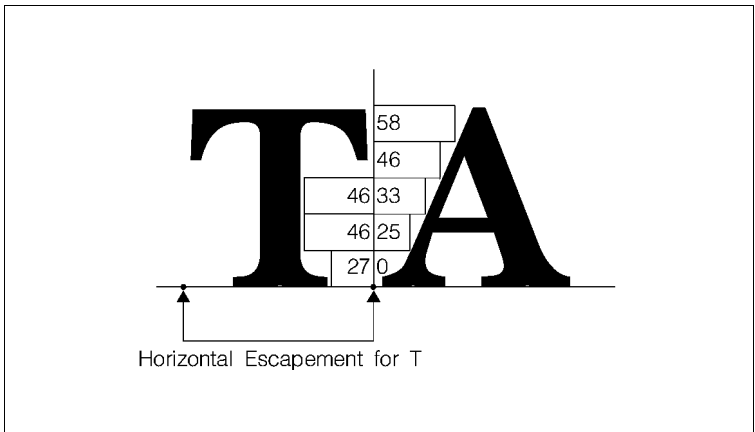


Figure G-3. The Letters T and A with No Kerning

To determine the maximum amount to which T and A can be kerned, the values of the sectors are first summed.

Sector Values Summed: 1 = 27
 2 = 71
 3 = 79
 4 = 46
 5 = 58

The smallest value is 27. To adjust for maximum kerning, 27 is subtracted from the horizontal escapement of T (135), with the result being 108. Setting the A after the T with an escapement of 108 results in a kerned pair with maximum kerning. Figure G-4 shows TA with maximum kerning applied.

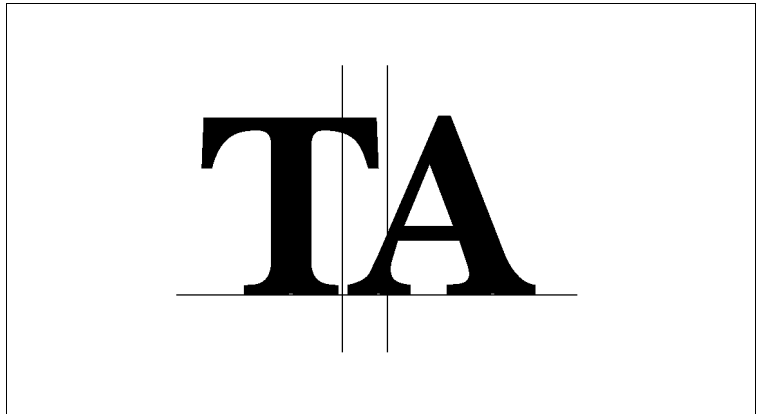


Figure G-4. The Kerned Letters T and A

As a general rule, kern pairs can be derived for any two characters by using the sector kern information and applying a factor of 0.5 to the resulting maximum kern value. The result is subtracted from the escapement value of the first character.

Track Kerning

Track adjusting (kerning) is based on an algorithm that uses information provided by the typeface designer. Track adjusting is used to reduce the white space between all characters uniformly. The degree to which text is adjusted is dependent upon the track being used and the point size of the typeface. Each track covers a range of point sizes and contains 4 values which define the point size and kern range. The tighter the track kerning, the larger the minimum point size is.

The first two values assigned would be the minimum and maximum point size to which this track adjustment information applies. Next are the minimum and maximum values. This can be determined from the sector kern information. If the point size of the font is less than the minimum point size specified then the minimum kern (usually zero) value is used. If the point size for the font is greater than the maximum point size then the maximum kern value is used. Inside this range the following formula is applied:

$$X = ((K_{max} - K_{min}) / (P_{max} - P_{min})) * P_{curr} + (K_{min} * P_{max} - K_{max} * P_{min}) / (P_{max} - P_{min})$$

P_{curr} = Current point size.

P_{max} = Maximum point size to which track applies.

P_{min} = Minimum point size to which track applies.

K_{max} = Maximum kern value.

K_{min} = Minimum kern value.

X = Amount by which to kern (This value is subtracted from the horizontal escapement of the preceding character. The larger the point size, the tighter the characters can be placed together.)

Track kerning can also be used in conjunction with sector kern information to achieve customized results as shown below.

If:

$P_{max} = 20 \text{ pt.}$

$P_{min} = 5 \text{ pt.}$

$K_{max} = 5 \text{ pt.}$

$K_{min} = 0 \text{ pt.}$

Using the formula above, and given the current point size (P_{curr}), the kerning distance for each point size shown (X) is:

P_{curr}	4	5	10	18	20	21
X	0	0	1.667	4.333	5	5

HP MSL Character Number Table

H

The Character Table (Table H-1) lists all of the symbols in the HP Master Symbol List. For each character in the table, the Agfa (CG) character code number is cross-referenced to the HP Master Symbol List number and the Character Name description.

Table H-1. Character Table (HP Master Symbol List)

CG Char. Code	HP MSL No.	Character Name	CG Char. Code	HP MSL No.	Character Name
1	86	t	35	42	l
2	81	o	36	49	P
3	74	h	37	36	C
4	80	n	38	55	V
5	79	m	39	38	E
6	78	l	40	59	Z
7	84	r	41	37	D
8	73	g	42	35	B
9	75	i	43	52	S
10	82	p	44	58	Y
11	69	c	45	39	F
12	88	v	46	57	X
13	71	e	47	34	A
14	92	z	48	56	W
15	70	d	49	43	J
16	68	b	50	54	U
17	85	s	51	50	Q
18	91	y	52	44	K
19	72	f	53	6	ampersand
20	90	x	85	1	exclam
21	67	a	86	32	question
22	89	w	87	66	quoteleft
23	76	j	88	8	quoteright
24	87	u	96	188	guillemotleft
25	83	q	97	190	guillemotright
26	77	k	98	121	exclamdown
27	53	T	99	122	questiondown
28	48	O	106	221	doubleexclam
29	41	H	172	148	AE
30	47	N	173	1091	OE
31	46	M	174	152	ae
32	45	L	175	1090	oe
33	51	R	176	1047	ij
34	40	G	177	159	germandbls

Table H-1. Character Table (HP MSL) (Cont'd.)

CG Char. Code	HP MSL No.	Character Name	CG Char. Code	HP MSL No.	Character Name
231	137	agrave	367	114	Yacute
232	133	aacute	368	115	yacute
233	141	adieresis	386	1096	lslash
234	129	acircumflex	392	1095	Lslash
235	163	atilde	423	1093	guilsingright
236	149	aring	424	1092	guilsingleft
238	118	ccedilla	444	1017	quotedblleft1
239	138	egrave	445	1018	quotedblright
240	134	eacute	480	1036	prescription
241	142	edieresis	493	1104	copyrightlarge
242	130	ecircumflex	497	1105	trademarksans
243	328	dotlessi	541	1106	Zcaron
244	150	iacute	542	172	Scaron
245	154	igrave	551	101	Egrave
246	158	idieresis	558	155	Odieresis
247	146	icircumflex	563	153	Adieresis
248	120	ntilde	564	145	Aring
249	135	oacute	565	161	Aacute
250	139	ograve	566	99	Agrave
251	143	odieresis	567	100	Acircumflex
252	131	ocircumflex	568	162	Atilde
253	151	oslash	569	160	Ocircumflex
254	136	uacute	570	157	Eacute
255	140	ugrave	571	102	Ecircumflex
256	144	udieresis	572	103	Edieresis
257	132	ucircumflex	573	167	Igrave
258	171	otilde	574	166	Iacute
276	117	Ccedilla	575	104	Icircumflex
291	147	Oslash	576	105	Idieresis
349	1107	IJ	577	119	Ntilde
363	164	Eth	578	169	Ograve
364	177	Thorn	579	168	Oacute
365	165	eth	581	170	Otilde
366	178	thorn	582	174	Uacute

Table H-1. Character Table (HP MSL) (Cont'd.)

CG Char. Code	HP MSL No.	Character Name	CG Char. Code	HP MSL No.	Character Name
583	111	Ugrave	1303	1094	opensquare
584	112	Ucircumflex	1307	1110	encirclemedium
585	156	Udieresis	1323	297	greaterthanorequalto
597	173	scaron	1324	298	lessthanorequalto
599	1031	zcaron	1335	295	intersection
720	176	ydieresis	1374	293	phi1
721	175	Ydieresis	1404	189	closedbox
790	1067	quotesinglbase	1406	1111	embulletsolid
791	1103	registersans	1407	1109	encircleopen
812	0	thinspaceinverse	1408	331	bullet
813	1117	enspaceinverse	1451	653	emboxopen1
814	1118	emspacereverse	1565	15	period
839	1068	perthousand	1566	13	comma
1094	1019	quotedblbase	1567	27	colon
1215	283	Gamma	1568	28	semicolon
1219	98	Delta	1569	9	parenleft
1228	285	Sigma	1570	10	parenright
1231	289	Theta	1571	60	bracketleft
1234	288	Phi	1572	62	bracketright
1237	290	Omega	1573	14	hyphen
1240	281	alpha	1574	326	endash
1241	282	beta	1575	325	emdash
1242	291	delta	1576	16	slash
1243	304	eta	1591	332	nsuperior
1246	180	mu	1592	18	one
1248	284	pi	1593	19	two
1250	286	sigma	1594	20	three
1254	294	epsilon	1595	21	four
1261	287	tau	1596	22	five
1267	301	approximate	1597	23	six
1273	296	equivalence	1598	24	seven
1282	1100	register2	1599	25	eight
1283	1101	copyright2	1600	26	nine
1286	303	radical1	1601	17	zero

Table H-1. Character Table (HP MSL) (Cont'd.)

CG Char. Code	HP MSL No.	Character Name	CG Char. Code	HP MSL No.	Character Name
1602	4	dollar	1719	3	numersign
1603	128	cent	1722	1028	ellipsis
1619	200	onesuperior	1725	125	yen
1620	197	twosuperior	1727	310	minute
1621	198	threesuperior	1728	311	second
1622	1001	foursuperior	1751	1023	thinspace
1623	1002	fivesuperior	1752	1021	enspace
1624	1003	sixsuperior	1753	1020	emspace
1625	1004	sevensuperior	1807	232	Pt
1626	1005	eightsuperior	1825	123	currency
1627	1006	ninesuperior	1827	61	backslash
1628	1000	zerosuperior	1846	1412	ogonek
1656	324	fraction	1847	1407	hungarumlaut
1673	185	onehalf	1851	1411	cedilla
1674	184	onequarter	1852	1400	acute
1675	182	threequarters	1853	1401	grave
1684	302	centerperiod	1854	1403	dieresis
1700	124	sterling	1855	1402	circumflex
1702	5	percent	1856	1404	tilde
1703	127	florin	1857	1408	ring
1704	12	plus	1858	1405	caron
1705	183	minus	1859	1410	macron
1706	191	plusminus	1860	1406	breve
1707	30	equal	1861	1409	dotaccent
1708	201	multiply	1873	1425	Ogonek
1709	202	divide	1874	1420	Hungarumlaut
1711	186	ordfeminine	1878	1424	Cedilla
1712	187	ordmasculine	1879	1413	Acute
1713	116	degree	1880	1414	Grave
1714	312	dagger	1881	1416	Dieresis
1715	327	daggerdbl	1882	1415	Circumflex
1716	126	section	1883	1417	Tilde
1717	181	paragraph	1884	1421	Ring
1718	11	asterisk	1885	1418	Caron

Table H-1. Character Table (HP MSL) (Cont'd.)

CG Char. Code	HP MSL No.	Character Name	CG Char. Code	HP MSL No.	Character Name
1886	1423	Macron	7259	1113	arrowtab
1887	1419	Breve	7260	1114	hardspace
1888	1422	Dotaccent	7274	113	connector
1904	179	catalan	7275	314	connectordouble
1905	1427	Catalan	7276	233	logicalnotreverse
3171	1060	careofscript	7277	228	logicalnotflopped
4662	1112	carriagereturn	7278	31	greater
4757	1040	fi	7279	29	less
4758	1041	fl	7280	63	circumflex1
4759	1042	ff	7281	94	arrowvertex1
4760	1043	ffi	7282	64	ruleunderline
4761	1044	ffl	7283	96	asciitilde
4767	196	registered	7284	375	asciigrave
4768	193	copyright	7286	192	brokenbar
4770	313	trademark	7287	309	apostrophen
4771	1034	slogan	7291	306	Lcatalan
4793	1099	carriagereturn1	7295	307	lcatalan
7002	231	trianglesoliddown	7297	370	lira
7004	230	trianglesolidup	7304	322	Caronspacing
7005	219	trianglesolidleft	7305	330	Cedillaspacing
7006	218	trianglesolidright	7306	318	Gravespacing
7009	661	emboxsolid	7307	317	Acutespacing
7018	194	logicalnot	7308	320	Dieresisspacing
7248	308	liter	7309	319	Circumflexspacing
7249	33	at	7310	321	Tildespacing
7250	93	braceleft	7311	1085	Macronspacing
7251	95	braceright	7312	323	Ringspacing
7252	305	enboxsolid	7313	107	gravespacing
7253	1108	enboxopen	7314	106	acutespacing
7254	1102	trademark	7315	109	dieresisspacing
7255	329	quotesingle	7316	108	circumflexspacing
7256	2	quotedbl	7317	110	tildespacing
7257	1115	hyphensoft	7318	1084	macronspacing
7258	1116	embox	7319	316	ringspacing

Table H-1. Character Table (HP MSL) (Cont'd.)

CG Char. Code	HP MSL No.	Character Name	CG Char. Code	HP MSL No.	Character Name
7320	199	cedillaspacing	7402	245	lowerrghtbxcrndbl
7321	315	caronspacing	7403	242	rightboxsidedouble
7322	1097	hungarumlautspacing	7404	259	middleboxbottomdbl
7323	1045	Hungarumlautspacing	7405	260	middleboxtopdouble
7345	1086	brevespacing	7406	261	leftboxsidedouble
7346	1087	Brevespacing	7407	262	cntrboxbarhorizdbl
7347	1089	Dotaccentspacing	7408	272	boxintersngltodble
7348	1088	dotaccentspacing	7409	256	lftboxsiddbltosngl
7349	1098	ogonekspacing	7410	243	centerboxbarvertdbl
7350	1030	Ogonekspacing	7411	239	rightboxsidesngdbl
7378	254	thinintersctinglines	7412	267	midtopsingletodble
7379	236	thinverticalline	7413	265	midbotdbletosingle
7380	237	rightmiddleboxside	7414	268	lwrleftcrndbltosngl
7381	250	middleboxbottom	7415	271	upprlftdbletosngl
7382	251	middleboxtop	7416	240	upprrghtboxcrner
7383	252	leftmiddleboxside	7417	246	lwrrghtbxsnngltodble
7384	253	centerboxbarhoriz	7418	276	solidfillcharacter
7385	248	upperrightboxcorner	7419	277	solidfillbottomhalf
7386	274	lowerrightboxcorner	7420	280	solidfilltophalf
7387	249	lowerleftboxcorner	7421	278	solidfilllefthalf
7388	275	upperleftboxcorner	7422	279	solidfillrighthalf
7389	273	boxinterdbletosngle	7423	222	thickunderliningrule
7390	264	midbotsngletodble	7424	235	seventyfivegraytint
7391	266	midtopdbletosngle	7425	234	twentyfivegraytint
7392	238	rightboxsiddblesingl	7426	97	error
7393	255	leftboxsidesngltodbl	7427	205	heart
7394	247	lwrrghtboxdbltosngl	7428	206	diamond1
7395	269	lwrleftsngletodble	7429	207	club
7396	270	uprllftsngletodble	7430	208	spade
7397	241	uprrghtboxdbltosngl	7431	203	smileface
7398	263	boxintersectiondbl	7432	204	darksmile
7399	257	lowerleftbxcrndble	7433	215	eighthnote
7400	258	uprleftboxcrndble	7434	216	doublesixteenthnote
7401	244	uprrghtboxcrndble	7435	214	female

Table H-1. Character Table (HP MSL) (Cont'd.)

CG Char. Code	HP MSL No.	Character Name	CG Char. Code	HP MSL No.	Character Name
7436	213	male	7472	508	Lambda
7437	217	compass	7473	509	Xi
7438	209	embulletsolid1	7474	510	Pi
7439	211	emcircleopen	7475	511	Sigma
7440	210	inversecenterbullet	7476	512	Upsilon
7441	212	inverselargecircle	7477	513	Phi
7442	292	infinityserif	7478	514	Psi
7444	299	integraltp	7479	515	Omega
7445	300	integralbt	7480	516	gradient
7446	225	arrowdown	7481	517	partialdiff
7447	224	arrowup	7482	518	sigma1
7448	227	arrowleft	7483	519	notequal
7449	226	arrowright	7484	520	summationbottomrule
7450	220	northsoutharrow	7485	522	alpha
7451	223	nrthstharrowperp	7486	523	beta
7452	229	arrowboth	7487	524	gamma
7453	628	northwestarrow	7488	525	delta
7454	625	northeastarrow	7489	526	epsilon
7455	626	southeastarrow	7490	527	zeta
7456	627	southwestarrow	7491	528	eta
7457	554	arrowdblleft	7492	529	theta
7458	552	arrowdblright	7493	530	iota
7459	551	arrowdbldown	7494	531	kappa
7460	553	arrowdblup	7495	532	lambda
7461	556	arrowdblboth	7496	533	mu
7462	555	doublearrowupdown	7497	534	nu
7463	500	radical	7498	535	xi
7464	501	isproportionalserif	7499	536	omicron
7466	502	baseofnaturallog	7500	537	pi
7467	503	Epsilon	7501	538	rho
7468	504	therefore	7502	539	sigma
7469	505	Gamma	7503	540	tau
7470	506	Delta	7504	541	upsilon
7471	507	Theta	7505	542	phi

Table H-1. Character Table (HP MSL) (Cont'd.)

CG Char. Code	HP MSL No.	Character Name	CG Char. Code	HP MSL No.	Character Name
7506	543	chi	7540	583	functionalcomposite
7507	544	psi	7541	584	circlelarge
7508	545	omega	7542	586	perpendicularleft
7509	546	theta1	7543	585	perpendicularright
7510	547	phiopenalternate	7544	587	integral
7511	548	pialternate	7545	588	contourintegral
7512	549	approximatelyequal	7546	589	angle
7513	550	notidenticalwith	7547	590	emptyset
7514	557	rghtarrwoverleftarrw	7548	591	infinity
7515	558	reversiblereaction	7549	592	Beth
7516	559	vectorsymbol	7550	593	Gimel
7517	560	summationtoprule	7551	594	cblackletter
7518	561	universal	7552	595	iblackletter
7519	562	existential	7553	596	Rfraktur
7520	563	perpndiclarupsidedwn	7554	597	zblackletter
7521	564	perpendicular	7555	598	bracketlefttp
7522	565	union	7556	599	bracketleftbt
7523	566	element	7557	600	bracelefttp
7524	567	suchthat	7558	601	braceleftmid
7525	568	notelement	7559	602	braceleftbt
7526	569	propersubset	7560	603	contourintegcenter
7527	570	propersuperset	7561	604	summationtoplftcrner
7528	571	notsubset	7562	605	centerlinedbl
7529	572	doesnotcontain	7563	606	summatnbottmlftcrner
7530	573	reflexsubset	7564	607	summationlowerdiag
7531	574	reflexsuperset	7565	608	bracketrighttp
7532	575	circleplus	7566	609	bracketrightbt
7533	576	sun	7567	610	bracerighttp
7534	577	circlemultiply	7568	611	bracerightmid
7535	578	abstractminus	7569	612	bracerightbt
7536	579	abstractdivide	7570	613	braceverticalpiece
7537	580	logicaland	7571	614	histogram
7538	581	logicalor	7572	615	radicalcomposite
7539	582	exclusiveor	7573	616	summationtopright

Table H-1. Character Table (HP MSL) (Cont'd.)

CG Char. Code	HP MSL No.	Character Name	CG Char. Code	HP MSL No.	Character Name
7574	617	summationcenterpiece	7609	656	qrtrcrlbottomright
7575	618	summationbottomright	7610	657	productbottom
7576	619	summationupperdiag	7611	658	producttop
7577	620	minusplus	7612	659	brackrghtdbltop
7578	621	obtuselessthan	7613	660	brackleftdblbottom
7579	622	obtusegreaterthan	7615	662	diamond
7580	623	masksymbol	7616	663	suchthatrotated
7581	624	congruent	7617	664	asterisk
7582	629	triangleup	7618	665	arrowextndrsnglhoriz
7583	630	triangleright	7619	666	arrowextndrdblhoriz
7584	631	triangledown	7620	667	revsummationcenter
7585	632	triangleleft	7621	521	because
7586	633	muchlessthan	7622	333	homeplate
7587	634	muchgreaterthan	7623	668	rightanglesymbol
7588	635	identical			
7589	636	definedas			
7590	637	Digamma			
7591	638	plancksconstant			
7592	639	lagrangian			
7593	640	powerset			
7594	641	weierstrass			
7595	642	summation			
7596	643	bracketleftdouble			
7597	644	bracketdoublemiddle			
7598	645	bracketrightdouble			
7599	646	quartercirtopleft			
7600	647	qrtrcrlbottomleft			
7601	648	unioncompositemiddle			
7602	649	intersection			
7603	650	union			
7604	651	brackleftdbltp			
7605	652	brackleftdblbottom			
7607	654	lozenge			
7608	655	quartrcirtopleft			

Index

- !**
- 600 dpi 1-10, 13-3
 - Ec?DC1 Command A-16
 - EcE Reset 2-11
- A**
- Abbreviated Font Selection Commands
 - Problems With 5-3
 - Absolute Character Size (SI) 10-10, 10-18
 - Absolute Cursor Positioning 4-6 - 4-7
 - Absolute Direction (DI) Command 10-15
 - Absolute Units
 - HP-GL/2 10-11
 - Accessing Special Characters
 - Example 5-19
 - ACG Character Numbers D-3, H-1
 - Actual Printable Area 3-4
 - Adaptive compression 9-5, 9-12, 13-10, 14-2, E-6
 - Example 9-14
 - Addressable Area
 - See Logical Page
 - Adjusting Line Spacing
 - Example 5-17
 - Adjusting Line Spacing For Point Size 5-17
 - AG.C
 - printCharacterMetrics 6-71
 - printGeneralInfo 6-69
 - printKerningInfo 6-72
 - printSymbolMSLInfo 6-71
 - printSymSetInfo 6-71
 - printTypefaceMetrics 6-70
 - AG.C (Sample TFM Implementation) 6-68
 - Agfa Compugraphic 7-14
 - Typeface Number C-14
 - Anchor Point
 - Picture Frame 10-4
 - See also Picture Frame Anchor Point
 - Angle
 - Printing Fonts at Any 10-15
 - Anisotropic Scaling
 - Relative vs. Absolute 10-20
 - Anisotropic Scaling of Fonts 10-18
 - Appearance width (Tag 414) 6-19, 6-70, 6-75 - 6-76, 8-3
 - Application Downloader 7-11
 - Arabic characters D-9
 - Array length 6-12
 - Array Length (refer Tag 404) 6-71
 - Ascent (Tag 425) 6-25, 6-70, 8-5
 - ASCII Generator 6-68
 - Aspect Ratio 10-11
 - Auto Macro Overlay, Disabled 3-9
 - Auto-Rotation
 - Effect on Memory A-11
 - Fonts 5-13
 - Raster Images 9-20
 - AutoFont Support 1-7, 6-1, 6-52, 13-9
 - End-User Considerations 6-57
 - Font Cartridges 6-48
 - AutoFont Support for TrueType 8-1
 - AutoFont Support Installer 6-1, 6-47, 6-54, 6-57, 6-79 - 6-80, 7-2, 7-6 - 7-7, 7-10, C-1, C-6 - C-7
 - Automatic Font Support 6-1
 - Automatic Forms Overlay, Example 12-4
 - Average width (Tag 419) 6-22, 6-70, 8-4
- B**
- Base Driver 6-66
 - Bezier curves 1-8, 1-10, 10-1, 10-6
 - Bi-directional communications 2-23
 - BIN format, converting to E-3
 - Bin Selection 2-32
 - Binary Transfer 9-5
 - Binary Transfer Character
 - Upper-Case Only A-15
 - BINS.ZIP E-1
 - Bitmap files 8-1
 - Bitmapped Fonts 5-2, 6-47 - 6-48
 - Compressed 1-9
 - Bitmaps 6-81
 - Blackwidth 6-23
 - Block size, raster graphics E-8
 - Bottom Margin 3-13
 - Establishing 3-16

Bottom of Page		Example	3-22
Placing Text at	4-15	Character table	H-1
Bottom-Most Position		Characters	
Moving Cursor To	4-14	Accessing Special	5-18
BR Command	10-1	choose_mode()	E-8
BUILDSYM		Clearing fonts from memory	2-35
Example	D-5	Clearing printer memory	2-35
using	D-5	Clipping	4-11
BUILDSYM kit	D-1	and Cursor Positioning	4-11
BUILDSYM operation	D-2	at HP-GL/2 Picture Frame Boundaries	10-20
BUILDSYM utility	D-1	Cell-Level	4-11
Bullet	7-10	Dot-Level	4-11
BZ Command	10-1	Example Demonstrating	3-7
		Graphics	14-4
		Outside Printable Area Boundary	3-5
		Outside Raster Image Area	9-4
		Solution for	14-1
		See Unprintable Region	
		Combining escape sequences	1-7, 13-1
		Font Selection	5-5
		Command Byte	
		Delta Row Compression	9-8
		Commands	
		Consolidating with Macros	12-1
		PCL Job setup	2-5
		Comment (Tag 402)	6-69
		COMMENT Command, PJI	2-4
		Comment information (Tag 402)	6-11, 8-2
		Common problems	14-1
		Compatibility Issues	A-10
		Compressed fonts	1-9
		CompressErgs()	E-8
		Compression	
		mode 0	E-5
		mode 1	E-5
		mode 2	E-6
		mode 3	E-6
		mode 5	E-6
		Raster Graphics	9-5
		Compression Mode Command	9-5
		Compression Mode Performance	9-15
		Compression Modes	
		Comparison	E-5
		Raster Graphics	9-2
		Compression, Lossless/Lossee	A-11
		CompressLJ3()	E-7
		Consolidating Commands	
		Using Macros	12-1
		Control panel	2-7, 2-9
		Overriding settings	2-37
C			
C Programming Language, Examples	B-1		
Calculating VMI	5-18		
call_program	C-12, C-16		
Calling a Macro	12-2		
CAP			
See Current Active Position			
Floating	A-13		
Starting Cursor Position	A-17		
CAP (Current Active Position)	9-17		
Capheight (Tag 423)	6-24, 6-70, 8-4		
Carriage return	2-4		
Cartridge fonts (glue file)	6-37		
Cartridge-based fonts	6-54		
Cartridges			
Macro	12-2		
Scalable Typeface	6-48		
Cell-Level Clipping	4-11		
CG character numbers	H-1		
CG-to-MSL conversion	6-97		
CG2HP.EXE	D-4		
Changing Character Spacing	3-21		
Changing Orientation	3-9		
Example	3-10		
Changing the print environment	2-9		
Chapter summary	1-1		
Character ascent (Tag 437)	6-29, 6-72, 8-6		
Character descent (Tag 438)	6-29, 6-72, 8-6		
Character descriptor	7-3		
Character Index	6-72		
Character numbers	D-3		
Character parameters	6-27		
Character Reference Point (CRP)	6-28		
Character Spacing			

Control Panel job offset setting	2-31
Control panel reset	2-11
Control Panel Settings	
Overriding	13-2
Control panel, Reset Menu	2-11
Controlling Right Margin	3-20
Example	3-20
Controlling the Left Margin	3-19
Controlling Top Margin	3-15
Conventions used in manual	1-6
Converting .TIF to .BIN format	E-3
Converting to HP MSL numbers	D-4
Coordinate System	
HP-GL/2 & PCL	10-7
Copyright (Tag 401)	6-69
Copyright Information (Tag 401)	6-11, 8-1
CR	2-4
Creating TFM files	6-81
Current Active Position	
See also Cursor Positioning	
Floating CAP	A-12
Raster Graphics	4-2
Rectangular Area Fill (Rules)	4-4
Saving/Restoring	4-10
Starting Position	A-17
Text	4-1
Vector Graphics (HP-GL/2)	4-3
Current Active Position (CAP)	4-1
Cursor	
Storing (Example)	12-3
Cursor Position	9-17
Saving	4-10
Starting	A-17
Cursor Positioning	
Absolute	4-7
Absolute vs. Relative	4-6
At Page Limits	4-11
See Current Active Position	
Dots vs. Decipoints vs. Columns/Rows	4-7
Raster Graphics	9-17
To Lowest Position	4-14
Units	4-7
Using Rows and Columns	4-9
Cyrillic characters	D-9

D

Data Compression	
Raster	9-2, 9-5
Data flow, TFM	6-85
Data Loss	4-11, A-11
See Unprintable Region	
Data loss and PJJ	2-34
Decipoints	
Using for Cursor Movement	4-9
Default	
HP-GL/2 Orientation	10-8
Default Symbol Set	A-14
Defaulting environment features	2-11
Delta Row Compression	9-5, 9-8, E-6
Command Byte	9-8
Example	9-11
Replacement Byte	9-8
Seed Row	9-8
Delta X	4-1
Descent	
Lower Case (Tag 428)	6-71
Descent (Tag 426)	6-25, 6-70, 8-5
Descent Distance	4-15
Design Unit Value	5-6
Design Units	6-11, G-3
Converting	6-64
Design Units (Tag 408)	6-15, 6-70, 8-3
Desired state	
Example (job setup)	2-15, 2-18, 2-26, 2-28
Desired state, setting features to	2-13
Desktop Publishing	4-9
Destination Image	11-1 - 11-2
Device data	6-35
Device dots	6-65, 8-8
Device-Independence	4-9
DI (Absolute Direction) Command	10-15, 10-17
Dingbats	C-18
Disk-based fonts	6-54
Display Functions Mode	A-16
Disproportionate Font Scaling	
See Anisotropic Scaling	
Distorted Scaling	
See Anisotropic Scaling	
Dot Placement Accuracy	4-16
Dot size	4-8
Dot-Level Clipping	4-11
Dots (see PCL Units)	4-7
Downloadable symbol sets	D-1
Downloading Fonts	5-9, 5-11

Example	5-11	Automatic Forms Overlay	12-4
Permanent vs. Temporary	5-12	Avoiding the Unprintable Region	3-17
Drivers		Changing Orientation	3-10
Reading Font Metric (TFM) Data	6-53	Character Spacing	3-22
Drop-Shadowed Type (Example)	11-12	Compatibility and Floating CAP	A-13
DT (Define Label Terminator) Command	10-17, 14-5	Controlling Right Margin	3-20
Dual Context Extensions	10-13	Delta Row Compression	9-11
Duplicate row command byte value	9-14	Downloading Fonts	5-11
		Drop-Shadow Effects Using Print Model	11-12
		Erasing fonts and macros	2-36
		Font Management	5-10
		Job Control and Page Setup	3-23
		Job Offset	2-32
		Job setup	2-26, 2-28
		Job setup without PJL	2-15
		Justifying Text	5-15
		Language switching	2-20
		LaserJet III Si job setup	2-18
		Merging Fixed-Pitch Text With Graphics	9-18
		Merging Proportional Text With Graphics	9-19
		Moving Cursor to Lowest Position	4-14
		PackBits (TIFF) Encoding	B-10
		Pattern-Filled Raster Graphics	11-8
		Pattern-Filled Raster Graphics on Black	11-9
		Pattern-Filled Type	11-11
		Placing Graphics at the Top-Most Position	4-11
		Placing Text at Page Bottom	4-15
		Placing Text at Top-Most Position	4-12
		Print Direction	B-1
		Print-and-Space Cursor Positioning	4-5
		Print-and-Space Formatting	3-18
		Printing Internal Scalable Typefaces	5-3
		Printing Rules	B-3
		Printing Two Orientations on a Page	3-11
		Rectangular Area Fill and	
		Pattern Transparency Mode	11-5
		Reverse and Pattern-Filled Type	11-10
		Right-Most Page Position	4-13
		Rotating Fonts at Any Angle	10-16
		Run-Length Encoding	9-7, B-10
		Sample .SYM file	D-7
		Saving/Restoring the Cursor Position	4-10
		Seeing the Printable Limits	3-7
		Selecting a Paper Source	3-1
		Selecting Fonts by Characteristic	5-5
		Selecting Legal-Size Landscape Page	3-4
		Setting the Left Margin	3-19
		Setting Up a Print Job	3-23
		Storing the Cursor When Using Macros	12-3
		TIFF (PackBits) Encoding	9-8
E			
Eastern European Latin characters	D-9		
Eighth Bit Shift Not Supported	A-14		
Ejecting Pages			
Solution for Unexpected	14-4		
Empty row command byte value	9-14		
End Graphics	9-4		
end parameter	D-10		
End raster graphics	9-16		
ENTER command, PJL	2-4, 2-18, 2-24, 2-34, 14-4		
Enter PCL Mode Command	10-17		
Effect on CAP	4-4		
Envelope feeder	3-1		
Envelopes, B5	1-10		
Environment			
Factory Default Environment	2-7		
Modified Print Environment	2-8		
Overlay environment	2-8		
PCL Current Environment	2-8		
User Default Environment	2-8		
Environment defaults	2-11		
Environment variables	2-14, 2-17, 2-24		
Environment, PJL Current	2-11		
EOJ command, PJL	2-8, 2-11 - 2-12		
Erasing fonts and macros	2-36		
Ergs_PCL	E-7		
Error 20, Solution for	14-1		
Error 21 (see page protection)			
Solution for	14-3		
and HP-GL/2	13-12		
Error Codes, Rambo	C-9		
Escape Sequences, Combining	13-1		
Escapement data, accessing	6-98		
Escapement value	6-28		
Example			
Accessing Special Characters	5-19		
Accessing the TFM Data	6-63		
Adaptive compression	9-14		
Adjusting Line Spacing	5-17		

TIFF Encoding	B-10	Font Cartridges	
User-defined patterns	11-14	Scalable	6-48
Using BUILDSYM	D-5	Screen Fonts for	7-13
Using Primary and Secondary Fonts	5-8	Font class (glue file)	6-44
Using Rambo (#1)	C-8	Font Descriptor	C-1
Using Rambo (#2)	C-8 - C-9	Font entry (glue file)	6-38
Using TFM Values in Calculations	6-64	Font file (glue file)	6-42
Using the Page Length Command	3-4	Font file formats	7-1
Using the Page Size Command	3-3	Font header	6-50
Executing a Macro	12-2	Font Height (Point Size)	
Expanded I/O	A-14	and Scalable Typefaces	5-6
		Font ID Number	5-11
		Font Management	
		and Macros	A-16
		Example	5-10
		General	5-9
		Font metrics	1-7
		Benefits of TFM Support	6-50
		Contained in TFM Data Structure	6-56
		Font metrics, reading	6-94
		Font parameters	6-23
		Font Selection	
		Eighth Bit Shift Not Supported	A-14
		Example	5-5
		Exceptions	5-5
		Using Shift In/Shift Out	5-8
		Font Selection by ID Number	5-3
		Font Selection Characteristics	
		Priority	5-4
		Font Spacing	
		Font Products Using the Same	5-18
		Font Support	
		Intellifont Integration	7-5
		Tips for Efficient	13-9
		Font support history	6-49
		Fonts	
		AutoFont Support	6-1
		Clearing from memory	2-35
		Compatibility	13-9
		Deleting with Reset	14-3
		Downloading	5-9, 5-11
		Drop Shadow (Example)	11-12
		Dynamic Font Metric Support	6-53
		Effect of ?E on	2-35
		Erasing downloaded	2-38
		Filling with Patterns/Gray Shades	11-1
		Justifying Text	5-14
		Mirror Image (Example)	B-6
		Mirrored	10-1
		Pattern-Filled (Example)	11-10, B-3
F			
Factory Default Environment	2-7, 2-11		
FAIS files	6-81, 7-1, 7-10, 8-1		
FASST	E-1		
Data size	E-10		
Integration	E-5		
Kit	E-1		
Modifying code	E-9		
Program structure	E-5		
Shell	E-1, E-3		
Shell (major functions)	E-6		
Testing data	E-11		
Testing results	E-12		
Feature comparison	A-1		
Feature Support Matrix	A-1		
Features			
Locking out	2-13, 2-16, 2-24		
File structure			
Job setup commands	2-5		
Print data	2-5		
See Print file structure			
Filenames			
TFM	6-67		
Filling images with user-defined patterns	11-13		
Filling Images/Fonts With Patterns			
See Print Model			
Filling polygons	1-8		
Filling Rectangles With Patterns	11-5		
First Character Index	6-72		
First code parameter	D-9		
Fixed pitch	6-18		
Floating CAP	A-12		
Avoiding Problems (Example)	A-13		
Floating Underline	5-19		
Font Auto-Rotation	5-13		
and Memory Use	5-13		

Permanent	2-35, 5-11
Primary and Secondary	5-8
Print Model (Example)	B-3
Printing at Any Angle	10-15
Printing on 45-Degree Angle (Example)	B-6
Resident	6-48
Reverse Type (Example)	11-10
Reverse Type (Print Model)	11-1
Rotated at 30-Degree Increments (Example)	B-7
Rotating at any Angle	10-1
Scaling	5-13
Scaling Anisotropically	10-18
Selecting	5-4
Selecting by Characteristic	5-3
SIMM-based	5-2
Special Effects with HP-GL/2 (Example)	B-6
Special HP-GL/2 Effects	10-15
Tips for Efficient Use	13-8
Unbound scalable	C-1
Using Shift In/Shift Out	13-8
Working With	5-14
FORMLINES (text length)	2-29
Forms	
Automatic Overlay (Example)	12-4
Using Macros for	12-1
Forms Applications	
Skew	4-16
Frame	
PCL Picture Frame	10-3
FT Command	10-1
FT22	10-2

G

General Font Management	5-9
getstring	C-12
Glue file	6-37, 6-80
format	6-37
Glue file, sample	6-45
Graphics	
Clipping	14-4
When to Use Vector vs. Raster	10-6
Graphics compression	
Optimizing	E-1
Graphics Limits	
Vector	10-9
Greek characters	D-9

H

Half Line Feed	3-17, 4-1
Hard-Clip Limits	10-9
Hard-Coding TFM Data	6-52
Header bytes, user-defined pattern	11-15
Headlines	
Line Spacing For	5-17
Hebrew characters	D-9
Hewlett-Packard Graphics Language (HP-GL/2)	10-1
HMI	
Default After Font Changes	A-15
Defaulted	3-9
See Horizontal Motion Index	
HMI (Horizontal Motion Index)	3-19
Horizontal Escapement	4-1, 4-13, 5-14, 6-28, 6-64, 6-72, 6-98, 8-6, G-4
Horizontal Motion Index	3-21, 4-1, 4-9
Default	3-19
Defaulting After Font Changes	A-15
Example	3-22
Horizontal Motion Index (HMI)	3-19
How to Use This Manual	1-1
HP MSL Character Table	H-1
HP MSL numbers	D-3
HP-defined patterns	11-1
HP-GL/2	1-8, 10-1
and Error 21	13-12
and Macros	12-7
Aspect Ratio	10-11
Basic Steps for Creating Plots	10-5
Basic Steps for Importing Plots	10-4
C Examples	B-9
CAP	4-3
Clipping Graphics Unexpectedly	14-4
Coordinate System	10-7
Default Orientation	10-8
Font Effects (Example)	B-6
I/O Data Transfer	10-6
Images Not Printing Properly	14-5
Labels (Text)	10-11
Line Joins/Ends	13-11
Memory Usage	10-12
Merging Graphics with Text	10-14
Page-Size Independent Plots	10-10
PCL Picture Frame	10-3
Picture Frame Anchor Point	10-4
Picture Frame Scaling Factor	10-3
Picture Presentation Directives	10-2
Plot Size	10-3, 10-11

Plotter Units	4-10	Importing HP-GL/2 Plots	
Printing Pie Chart (Example)	B-9	Basic Steps	10-4
Scaled Mode	10-10	IN (Initialize) Command	10-16
Scaling Factor & Picture Frame	10-10	Incorporating Intellifont	7-5
Special Font Effects	10-15	Index parameter	D-8
Stick Font	B-7	INFO command, PJJ	A-16
Syntax	14-5	Initialization	2-38, 13-2, 13-5
Tips for More Efficient Plots	13-11	LaserJet 4	13-8
Units	10-11	LaserJet III Si	13-7
User Units	4-10	non-PJJ	13-7
Vector Graphics Limits	10-9	INITIALIZE command, PJJ	2-7, 2-11
When to Use vs. Raster	10-6	INQUIRE command, PJJ	2-26
HP-GL/2 & PCL Orientation Interactions	10-7	Integrating FASST	E-5
HP-GL/2 and page protection	2-33	Integrating Intellifont without AutoFont installer	7-7
HP-GL/2 Commands	10-13	Intellifont	5-6, 6-48, 7-10, 8-1
(SS) Select Standard Font	10-17	Adding	7-5
DI (Absolute Direction)	10-15, 10-17	Availability	7-10
DT (Define Label Terminator)	10-17, 14-5	Bullet	7-10
Enter PCL Mode	10-17	Code	7-7
IN (Initialize)	10-16	Explanation	7-10
LO (Label Origin)	10-17	FAIS files	7-1
LT (Line Types)	10-11	Integration	7-1
PE (Polyline Encoded)	13-11	Requirements for adding	7-5
PU (Pen Up)	10-17	Inter-word Spacing (Tag 421)	6-70, 8-4
RO (Rotate)	10-7	Interface	
SC (Scale)	10-16	Video	A-14
SC (Scaled Mode)	10-10	Internal Fonts	6-48
SD (Standard Font Definition)	10-17	Internal fonts (glue file)	6-37
SI (Absolute Character Size)	10-10, 10-18	Internal Units	4-7, A-15
SI (Absolute Size)	B-6	Interword spacing (Tag 421)	6-24
SP (Select Pen)	10-16		
SR (Relative Character Size)	10-10, 10-18	J	
WU (Pen Width Selection Mode)	10-11	JOB command, PJJ	2-8, 2-11 - 2-12
HP-GL/2 Coordinate System	4-4	Job Control Example	3-23
HP-GL/2 in macros	12-1	Job Offset	2-31
HP-GL/2 Mode	4-4	Job Offset, Example	2-32
HP-GL/2 Plot Size	10-3	Job setup	2-1, 2-13, 13-2
		For PJJ printers	2-5
I		LaserJet 4 printer	2-22, 2-26
I/O, Expanded	A-14	LaserJet III Si printer	2-16
I/O Data Transfer		Non-PJJ printers	2-6, 2-13
and HP-GL/2	10-6	Tips	13-2
I/O Improvement	5-1	Job setup commands	
I/O, modular	A-14	Number of copies	2-5
Image fill		Reset	2-5
See Print Model		Justifying Text	5-14
Imagesetter	4-9	Example	5-15
		Using Relative Positioning Commands	4-6

M			
Macro		HP-GL/2	10-12
Automatic Overlay	12-3	Print protection	2-33
Call	12-2	menu	C-12
Execution	12-2	Menu reset, control panel	2-11
Macro cartridges	1-9, 4-16, 12-2	Merging Text with HP-GL/2 Graphics	10-14
Macro Control Command	12-4	Merging Text With Raster Graphics	9-17
Macro Management	12-4	Metric units	6-11
Macro SIMMs	12-2	Microsoft Windows	7-13
Macros	2-8, 4-10, 12-1	Minimum Kern Value (refer Tag 441)	6-73
and Display Functions Mode	A-16	Minimum Point Size	6-73
and Font Management Commands	A-16	MIO	A-14
Cartridges	4-16, 12-2	Mirroring Fonts (Example)	B-6
Creating	12-2	Misalignment	
General Management	12-4	Paper Registration Tolerance	3-8
Macro cartridges	1-9	Mode 0 compression	E-5
Nesting	12-7	Mode 1 compression	E-5
Permanent vs. Temporary	12-4	Mode 2 compression	E-6
Priority	12-6	Mode 3 compression	E-6
SRTool	F-7	Mode 5 compression	9-5, 9-12, E-6
Summary of Rules	12-6	Mode2Out()	E-9
Tips for Efficient	13-12	Mode3Out()	E-8
Use of HP-GL/2 in	1-8	mode_0_2_3_PCL()	E-7
using HP-GL/2 in	12-1	mode_0_2_PCL()	E-7
When to Use	12-1	mode_0_PCL()	E-7
Manual Feed	3-1	Modified Print Environment	2-8, 12-2
Manual Organization	1-1	Modular I/O	A-14
Margins	3-1, 3-13	Moving to the Right-Most Position	
Bottom	3-13	Example	4-13
Defaulted	3-9	MSL (see HP MSL)	D-3
Left	3-19	Multi-User Environment	5-3
Right	3-20	and Downloading Fonts	5-12
Right (Example)	3-20	Multiple copies	2-30
Top	3-15	Multiple Orientations on a Page	
Translated With Print Direction	3-11	Example	3-11
Marie	C-1	Multiple Print Directions	1-8
Master Symbol List (HP MSL) table	H-1		
MasterType Font Cartridges	6-48	N	
Math characters	D-9	Nesting Macros	12-7
Matrix		Networks	2-35
LaserJet Feature Comparison	A-1	NEXT_NAME	C-15
LaserJet Feature Support	A-1	NEXT_NUM	C-15
Maximum Kern Value	6-73	Nominal point size (Tag 407)	6-14, 6-70, 8-2
Maximum Point Size	6-73	Non-zero Winding Fill Type	1-10, 10-1
Maximum width (Tag 420)	6-23, 6-70, 8-4	Number of Characters	6-70
Memory		Number of copies	2-13, 2-16, 2-24, 2-30
and Performance	A-10	Number of copies command	1-10
Effect of Auto-Rotation on	A-11	Number of Kern Pairs	6-72
Memory usage	2-33, 2-35	Number of Sector Kern Characters	6-72

Number of Sectors per Character	6-72	Definition of Unsupported	A-17
Number of Symbol Sets (Tag 404)	6-70	Example	3-4
Number of Tracks	6-72	See also Page Size Command	
		Page protection	2-34, 10-12, 13-3
		LaserJet 4	13-12
		Page Setup	3-1
		Example	3-23
		Tips for Effective	13-4
		Page Setup Commands	2-5, 2-15
		Margins	2-5
		Page size	2-5
		Paper size/Paper Length	2-5
		Paper source	2-5
		Text length	2-5
		Page Size	
		Command	3-2
		Defaulted using Page Length Command	3-2
		Selecting	3-2
		Setting Using Page Length Command	3-3
		Page size command	2-15, 2-18, 2-20, 2-27, 2-29
		Definition of Unsupported	A-17
		Example	3-3
		Recommended	3-2
		Using	3-2
		Page-Size Independent Plots	10-10
		Pair kerning	6-30, 6-97, G-2
		PANOSE Information (Tag 443)	8-7
		PANOSE numbers (Tag 443)	6-36, 6-88
		PANOSE.IF file	6-88
		Paper Path and Cursor Placement	4-16
		Paper Registration Tolerance	3-8
		Paper Size	3-1
		Physical Page	3-4
		Paper Source	3-1
		Example	3-1
		Selecting	3-1
		Partial Pages	
		Solution for Printing	14-4
		Pattern	11-1
		for Print Model	11-1
		Pattern reference point	11-15
		Pattern Transparency Mode	11-1 - 11-2, B-3
		Example	11-5
		Pattern-Filled Raster Graphics, Example	11-8 - 11-9
		Pattern-Filled Type, Example	11-11
		Patterning Images	11-7
		Patterns, Filling Images With	11-7
		Patterns, User-Defined	10-2, 11-13
		PCL bitmap files	6-81, 8-1
		PCL char	D-8
O			
Offset, Job	2-31		
Opaque			
Pattern Transparency Mode	11-4		
Source Transparency Mode	11-3		
option_list	C-14		
Organization, Manual	1-1		
Orientation	2-15, 2-18, 2-20, 2-27, 2-29, 3-1		
Affect on Fonts	5-13		
Changing	3-9		
Effects on Margins	3-11		
Example (Changing)	3-10		
Four Choices	3-9		
Printing Multiple Orientations on Same Page	3-11		
Sending the Command at Page Start	3-9		
Orientation (glue file)	6-43		
Orientation Interactions, PCL & HP-GL/2	10-7		
Orientations, rotating	1-8		
Output Bin Selection	2-31 - 2-32		
output_dir	C-8		
output_file	C-8, C-17		
Overlying a Macro	12-3		
Overriding control panel settings	2-13, 2-16, 2-24, 2-37		
Overview of Manual	1-1		
P			
P1/P2	10-10		
PackBits (TIFF) Encoding	9-7		
Example	9-8		
PackBits Compression (Example)	B-10		
Page			
Logical	3-4		
Physical	3-4		
Page Boundaries			
Crossing with Raster Graphics	A-18		
Page Control Commands			
See Page Setup			
Page Eject			
Unexpected (Solution for)	14-4		
Page Length			
Defaulted	3-9		
Using to Set Page Size	3-2		
Page Length Command	3-3		

PCL Coordinate System	4-4	PJL	2-1, 2-3, 2-5, 13-4
PCL Job setup commands	2-5	and perishable data	2-34
PCL Mode	4-4	Command listing	A-9
PCL num	D-8	Command prefix (@PJL)	2-26
PCL Picture Frame	10-3, 10-7, 10-9	Commands	2-4
PCL Printer Language	1-6	ENTER command	2-18, 2-24, 2-34
PCL soft fonts (glue file)	6-37	ENTER LANGUAGE command	14-4
PCL status readback	5-10, F-10	INFO ID command	A-16
PCL Units	1-10, 4-7 - 4-8, 8-8, A-15	INITIALIZE	2-7, 2-11
Using for Cursor Movement	4-8	INQUIRE command	2-26
PCL value	6-79	Job setup	2-5
PCL2BIN.EXE	E-2	JOB/EOJ commands	2-8
pclwrite()	E-6	SET command	2-8 - 2-9, 2-23, 2-26, 2-34, 13-3, 14-3
PE (Polyline Encoded) Command	13-11	Status readback	F-13
Pen Location	4-3	Support	1-9 - 1-10
Pen Up (PU) Command	10-17	Syntax	2-4
Pen Width Selection Mode		USTATUS command	13-4
Relative vs. Metric	10-11	PJL COMMENT Command	2-4
Perforation Skip Mode	3-7, 4-5	PJL Current Environment	2-11
Perforation Skip Region	3-16 - 3-17, 4-5	PJL ENTER command	2-4
Performance		PJL-specific problems	14-3
and Memory	A-10	Plot Size, HP-GL/2	10-11
Font Support Tips	13-9	Plotter Units	4-10
Increase Using HP-GL/2 PE Command	13-11	HP-GL/2	10-11
Increase with Macros	12-1	PLU4-10	
Tips for Font Selection	13-8	Point (Tag 406)	8-2
Tips for Raster Graphics	13-9	Point Size	6-75
Vector-Processed Fonts	10-15	Adjusting Line Spacing For	5-17
Perishable data	2-34	and Scalable Typefaces	5-6
Permanent Fonts	5-11	Command	5-2
Downloading	5-11	Glue file	6-43
Permanent Fonts vs. Temporary Fonts	5-12	Tag 406	6-70
Personality	2-4	Point, exact size (Tag 406)	6-14
PERSONALITY variable	2-24	Portrait Orientation	3-9
Physical Job Offset	2-31	Positioning Units	4-7
Physical Page	3-1, 3-4	PostScript	2-1
Definition	3-4	Posture	6-76
Physical Page Size	3-2	Power cycling the printer	2-11
Picture Frame	10-3, 10-7	Presentation Mode, raster graphics	3-9, 9-13
Anchor Point	10-4, 10-7	Primary and Secondary Fonts	5-8
HP-GL/2	10-2	Print data	2-5
Scaling	10-18	Print Direction	
Scaling Factor	10-3	Defaulted	3-9
Units	10-11	Example	B-1
Picture Presentation Directives	10-2	No Affect on HP-GL/2 Orientation	10-8
Pie Chart	B-9	Print Direction Command	
Pitch	5-6, 6-74	Example Using	3-11
and Scalable Typefaces	5-6	Using	3-11
Pixels	6-65, 11-1, 11-3	Print directions, Multiple	1-8

Run-Length Encoding	9-6	RESET command, PJJ	2-11
Simple Binary Transfer	9-5	RESET MENU	2-7, 2-11
TIFF (PackBits) Compression (mode 2)	9-5	Resets	2-16
TIFF (PackBits) Encoding	9-7	Resident Fonts	6-48
Uncompressed	9-5	Scalable	6-48
Raster Data Compression	9-2	Resolution	1-10, 13-3
Performance	9-15	600 dpi	13-3
Raster graphics		Resolution enhancement	13-3
Across Page Boundaries	A-18	Resource (glue file)	6-40
Adaptive Compression	9-12	Resources for adding Intellifont	7-10
and Print Model (Example)	B-10	Return codes, Rambo	C-9
Auto-Rotation	9-20	Return Model Number Command	A-16
Compression (Examples)	B-10	Reverse Landscape Orientation	3-9
Cursor Positioning	9-17	Reverse Portrait Orientation	3-9
Data compression	1-8	Reverse Type	
Introduction	9-1	Example	11-10
Merging With Text	9-17	Right extent (Tag 436)	6-28, 6-72, 8-6
Presentation Mode	3-9	Right Margin	
Tips for Efficient	13-9	Controlling	3-20
Using	9-3	Right Side Sector Values	6-73
When to Use vs. Vector	10-6	RO (Rotate) Command	10-7
Raster Graphics Picture	9-1	Rotating Fonts	
Raster Graphics System	9-1	Effect on Memory	A-11
Raster graphics, end	9-16	Rotating Fonts (Example)	B-7
Raster Height	9-3	Rotating Fonts at Any Angle	
Raster Height Command	9-1	Example	10-16
Raster Presentation	9-3	Rotating Orientations	1-8
Raster Resolution	9-3	Rounding	4-9
Raster Width	9-3	Rules	
Raster Width Command	9-1	for Macros	12-6
Rasterizing the page	2-33	See Rectangular Area Fill	
READER.EXE	6-55, 6-68, 6-102	Run length encoding	E-5
File Size	6-55	Run-Length Compression	9-5
Recommended line spacing (Tag 422)	6-24, 6-70, 8-4	Run-Length Encoding	9-6
Rectangular Area Fill		Example	9-7
Example	11-5	Run-Length Encoding (Example)	B-10
Rectangular Area Fill (Rules)			
Using	11-5		
Relative Character Size (SR)	10-10, 10-18	S	
Relative Cursor Positioning	4-6	Sample glue file	6-45
Relative Plot		Saving the Cursor Position	4-10
See Page-Size Independent Plots		Saving/Restoring the Cursor Position	
Relative Scaling	10-20	Example	4-10
Replacement Byte		SC (Scale) Command	10-16
Delta Row Compression	9-8	Scalable disk-based fonts	6-54
requirements parameter	D-9	Scalable Font File	
Reset	2-8, 2-15, 2-19, 2-27, 2-29, 2-37	Making (M.C)	C-12
Overriding the control		Scalable Fonts	1-7, 7-3
panel settings	2-13, 2-16, 2-24	Unbound	C-2
Reset and the UEL command	2-5		

Scalable Internal Typefaces	6-48	Shift In/Shift Out	5-8
Scalable Typeface Products		for Performance Increase	13-8
Screen Fonts for	7-14	SI (Absolute Size) Command	10-18, B-6, B-8
Scalable Typefaces	5-2, 6-47 - 6-48	SI Command	10-11
Example	5-3	SIMM-based macros	12-2
Unbound	5-2	Skew	4-16
Scaled Mode		Slant (glue file)	6-43
HP-GL/2	10-10	Slant (Tag 413)	6-18, 6-70, 6-75, 8-3
Scaling Factor		soft fonts (glue file)	6-37
Picture Frame	10-3	Soft-Clip Limits	10-9
Scaling Factor and Picture Frame		Source Image	11-1 - 11-2
HP-GL/2	10-10	Source Transparency Mode	11-1 - 11-2, B-3
Scaling Fonts	3-12, 5-13	SP (Select Pen) Command	10-16
Anisotropically	10-18	Spacing (Tag 412)	6-18, 6-70, 6-74, 8-3
Scaling Fonts (Distorted)		Special Characters	
See Anisotropic Scaling		Accessing	5-18
Scaling Mechanism	10-11	SR (Relative Character Size)	10-10
Scanned Images	10-6	SR (Relative Character Size) Command	10-18
Screen Font Formatter	7-11	SR (Relative Size) Command	B-8
Screen font licensing	7-14	SRTool	F-1
Screen Fonts	7-13 - 7-14	Break points	F-23
SD (Standard Font Definition) Command	10-17	Byte counters	F-25
Second Character Index	6-72	Carriage return settings	F-23
Sector kern information (Tag 440)	6-31, 8-7	Checking the input buffer	F-24
Sector kerning	G-3	Clearing windows	F-24
sectorKernChar	6-72	Configuration	F-2
Seed Row	9-8	Data pacing	F-24
How Y Offset Affects	9-10	Display windows	F-5
Select Standard Font (SS) Command	10-17	Displaying line status	F-25
Selecting a Page Size	3-2	Displaying printer data	F-24
Selecting a Paper Source	3-1	DOS command line	F-7
Selecting Fonts	5-2	Error handling	F-6
Selecting Fonts by Characteristic	5-3	Exiting	F-9
Example	5-5	Files menu	F-18
Selecting fonts using TFM data	6-73	Initialization	F-2
Selecting Legal-Size Paper		Installing	F-2
Landscape Orientation	3-3	Macros	F-7
Selection string	6-35	Menus	F-8
Self Test		Options menu	F-18
and Automatic Reset	A-16	Output files	F-22
Semi-graphic characters	D-9	PCL macros	F-14
Serif style (Tag 415)	6-20, 6-70, 8-3	PCL status readback	F-10
SET command, PJI	2-8 - 2-9, 2-23, 2-26, 2-34, 13-3, 14-3	PJI macros	F-17
Setting up a print job	2-1	PJI status readback	F-13
Example	3-23	Record files	F-22
Setup		Requirements	F-1
Tips for Efficient Job	13-2	Running	F-2
Shell		Scopy	F-7
the Rambo Shell (M.C)	C-12	User interface	F-5
		User-defined escape sequences	F-12

Using mouse	F-9
SS (Select Standard Font) Command	10-17
Standard Font Definition (SD) Command	10-17
Start Graphics	9-3
Status readback	2-22 - 2-23, 5-10
Status Readback Tool (SRTool)	F-1
Stencil	11-2
Stroke weight	6-78
Stroke weight (glue file)	6-43
Stroke weight (Tag 411)	6-17, 6-70, 8-3
Structure	6-77
Structure, print file	
See Print file structure	
Style	6-75
Subfile type (Tag 400)	6-11, 8-1
Supporting TFM	
Benefits of	6-50
SV22	10-2
Switching printer languages	2-4
.SYM files	D-6
SYM file parameters	D-8
Symbol map (Tag 403)	6-12, 6-71, 8-2
Symbol set	2-15, 2-19 - 2-20, 2-27, 2-29, 6-74, 6-83
Abbreviations	C-12
Default	A-14
Definition File	D-2
Definition file, creating	D-6
Directory (Tag 404)	6-12, 8-2
Downloadable	D-1
Files	7-11
Glue file	6-42
Index Array	6-71
Index Array Offset	6-12
Maps	7-7, 7-11, C-9
Name (Tag 404)	6-71
Name Offset	6-12
Selection String	6-71
Selection String Offset	6-12
Symbol set directory (Tag 404)	8-2
symbol_set_dir	C-8
symbols parameter	D-10
SymSet	C-7
Syntax	
HP-GL/2	14-5

T

Table, Character number (HP MSL)	H-1
Tag descriptions, TFM	6-11
Tagged Font Metric files (TFM files)	1-7, 6-1
Tagged Image File Format (TIFF) Compression	9-5
Tagged Image File Format (TIFF) Encoding	9-7
Temporary Fonts vs. Permanent Fonts	5-12
Terms and Conventions, Manual	1-6
Testing data, FASST	E-11
Text Area	
Managing	3-13
Text Justification	5-14
Text Length	3-16
Defaulted	3-9
Defaulted with Top Margin Command	3-16
Text length (FORMLINES)	2-29
Text Readability	5-17
TFM	
adding new tags	6-89
glue file	6-37
TFM Data	3-20
Accessing Example	6-63
Printed to ASCII File	6-56
Using in Calculations	6-64
TFM data flow	6-85
TFM Data Structure	6-56
Accessing the	6-62
Examples	6-63
Printing to ASCII File	6-68
Reading	6-68
Typeface Sub-structure	6-62
TFM data types	6-9
TFM directory structure	6-6
TFM file structure	6-2
TFM files	6-1, 6-55, 7-5, 7-10
Creating	6-81
Naming Convention	6-67
Supplied	6-66
TFM files, major format version	6-4
TFM header structure	6-3
TFM path (glue file)	6-43
TFM Reader	6-1, 6-52, 6-54, 6-56
Data Flow	6-59
Example Implementation	6-55
for Hard-Coding Data	6-52
for Horizontal Escapement	5-14
Modifying the	6-61
READER.EXE	6-55
Sample TFM Implementation	6-68

Units of Movement	4-7
Universal Exit Language	13-2
Universal Exit Language (UEL command)	2-11
Unprintable Region	3-5, 4-11
Avoiding (Example)	3-17
Unsolicited status	13-4
Unsupported Page Length/Size Commands	A-17
Uppercase accent height (Tag 431)	6-26, 6-71, 8-5
User Default Environment	2-11
User Units	
HP-GL/2	10-11
User-Defined Patterns	10-2, 11-1, 11-13
Example	11-14
User-defined symbol sets	D-1
Using Primary and Secondary Fonts	
Example	5-8
Using Rambo	
Example #1	C-8
Example #2	C-8
Example #3	C-9
Using Rows and Columns	4-9
Using This Manual	1-1
USTATUS command, PJL	13-4

V

valid_symset	C-12, C-17
Vector Graphics	
See HP-GL/2	
Tips for More Efficient	13-11
When to Use vs. Raster	10-6
Vector Graphics Limits	10-9
Ventura Publisher	7-13
Vertical escapement (Tag 434)	6-28, 6-72, 8-6
Vertical Motion Index	3-22, 4-9
Calculating	5-18
Example	5-17
Video Interface	A-14
VMI	3-3, 3-22
Calculating	5-18
Defaulted	3-9
VMI (Vertical Motion Index)	2-15, 2-18, 2-20
VMI Command	3-2

W

Well-formed jobs	2-2
White Rule (Example)	B-3
Width	6-76
Word processing	2-30
Working With Fonts	5-14
WU (Pen Width Selection Mode)	10-11
WYSIWYG	7-1

X

x-height (Tag 424)	6-24, 8-4
Xerox Ventura Publisher	7-13
Xheight (Tag 424)	6-70

Y

Y Offset	9-4
Y Offset Command	9-3
For Performance Increase	13-9

Z

Zapf Dingbats	D-9
---------------	-----