

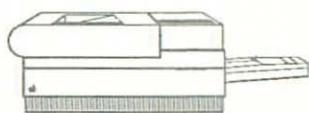
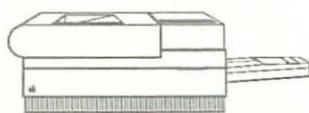
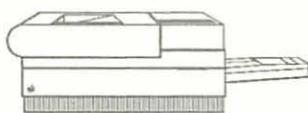
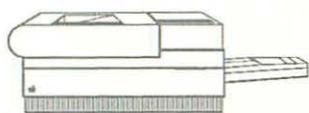


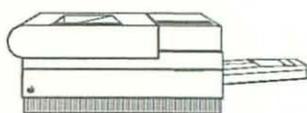
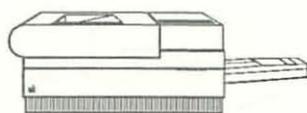
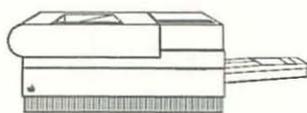
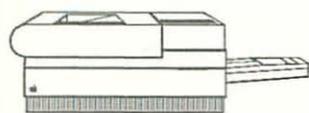
Apple®

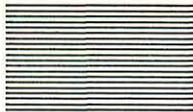
# LaserWriter® Reference



For LaserWriter, LaserWriter Plus, LaserWriter II<sub>NTX</sub>, and LaserWriter II<sub>TX</sub>







Apple® LaserWriter® Reference  
For the LaserWriter, LaserWriter  
Plus, LaserWriter IINT and IINTX



**Addison-Wesley Publishing Company, Inc.**

Reading, Massachusetts Menlo Park, California New York  
Don Mills, Ontario Wokingham, England Amsterdam Bonn  
Sydney Singapore Tokyo San Juan

🍏 APPLE COMPUTER, INC.

Copyright © 1988 by Apple Computer, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Computer, Inc. Printed in the United States of America.

© 1988 Apple Computer, Inc.  
20525 Mariani Ave.  
Cupertino, CA 95014  
(408) 996-1010

Apple, the Apple logo, AppleTalk, ImageWriter, LaserWriter, and Macintosh are registered trademarks of Apple Computer, Inc.

IBM is a registered trademark of International Business Machines Corp.

Diablo is a registered trademark of Diablo Systems Incorporated, a Xerox company.

Helvetica, Palatino, and Times are registered trademarks of Linotype Co.

ITC Avant Garde, ITC Bookman, ITC Garamond, ITC Zapf Chancery, and ITC Zapf Dingbats are registered trademarks of International Typeface Corporation.

LaserJet and LaserJet<sup>+</sup> are registered trademarks of Hewlett-Packard Corporation.

Microsoft is a registered trademark of Microsoft Corporation.

POSTSCRIPT is a registered trademark of Adobe Systems Incorporated. Adobe Illustrator is a trademark of Adobe Systems Incorporated.

Simultaneously published in the United States and Canada.

ISBN 0-201-19258-6  
ABCDEFGHIJ-DO-898  
First printing, March 1988

#### WARRANTY INFORMATION

**ALL IMPLIED WARRANTIES ON THIS MANUAL, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.**

Even though Apple has reviewed this manual, **APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL,** even if advised of the possibility of such damages.

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED.** No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.



# Contents

Figures and tables viii

Radio and television interference x

## **Preface** About This Book xi

What this book contains xii

How to use this book xiii

Visual cues and conventions xiv

Where to look for more information xiv

## **Chapter 1** Introduction 1

Laser printer models 3

    The LaserWriter and LaserWriter Plus 3

    The LaserWriter IISC 3

    The LaserWriter IINT and LaserWriter IINTX 4

    LaserWriter ROM features 5

        Original LaserWriter, LaserWriter Plus, and LaserWriter  
        IINT ROM features 5

        LaserWriter IINTX ROM features 5

    LaserWriter fonts 6

        The original LaserWriter 6

        The LaserWriter Plus, LaserWriter IINT, and  
        LaserWriter IINTX 6

The printing environment 6

    How you communicate 7

    Operating modes 7

    Software environment 7

        PostScript 9

        QuickDraw 9

        Font Manager 10

        Printing Manager 10

        LaserWriter driver 11

        AppleTalk 11

Approaching your application 12

## Chapter 2 Working with Fonts 13

- Basic font concepts 15
  - Font terminology 15
  - Bit-mapped fonts 16
  - LaserWriter printer fonts 18
  - Obtaining the font list 20
  - Coordinating noncoordinated fonts 22
  - Matching bit-mapped and printer fonts 24
- Classifying fonts 25
  - Classifying fonts for LaserWriter drivers of version 3.0 or later 26
    - Style-mapping table 28
    - Character-set-encoding table 29
    - Style-name table 30
- Naming fonts 33
  - Font-naming conventions for later drivers 33
- Downloading fonts 34
  - Downloading with later printer drivers 34
  - Downloadable fonts for all printer drivers 35
  - Temporary downloading of fonts 35
  - Permanent downloading of fonts 36
    - Fonts loaded before loading the dictionary 36
    - Fonts loaded after loading the dictionary 36
  - Downloadable file format 37
- Downloading fonts from non-Macintosh hosts 39
  - Downloading fonts permanently 39
  - Downloading fonts temporarily 40
  - Accessing downloaded fonts 40
  - Checking for downloaded fonts 40
  - Determining available room for fonts 41
  - Making room for more fonts 41
- LaserWriter IINTX font cache and disk file system 41
  - Disk and file-system overview 42
  - The Sys/Start file 43
    - Working with the Sys/Start file 44
  - Finding fonts on the disk file system 46
  - LaserWriter IINTX disk and file-system operators 46

## Chapter 3 Working in the Printing Environment 53

- LaserWriter operation modes 54
  - Printer switch settings 54
- Using AppleTalk 57
- Accessing the LaserWriter directly 59
  - Working interactively 60
  - Working in batch mode 64
- Using the Diablo 630 emulator 66
  - Invoking the Diablo emulator 66
    - Invoking the Diablo emulator with software 66
  - Changing print parameters 68
  - Deviations from Diablo 630 protocol 70
    - Detecting the end of a document 70
    - Double-striking for bold text 71
    - Proportional fonts 71
    - Paper positioning 71
    - Unsupported features and commands 71
- Using the Hewlett-Packard LaserJet<sup>+</sup> Emulator 72
  - Deviations from LaserJet protocol 73
  - Character font selection 73
    - Printing orientation (portrait versus landscape) 73
    - Symbol set (Roman-8, Line draw, Math, and so on) 73
    - Print pitch (nonproportional fonts only) 73
    - Character height (point size) 74
    - Stroke weight (light, medium, and bold) 74
    - Typeface (Courier, Pica, and so on) 74
  - Clipping region 74
  - Paper-size interactions 75
  - Symbol set 75
  - Character widths 76
  - Typeface size 77
  - Line printer font 77
  - Transparent communications 77
  - Longevity of downloaded information 78
  - Laserjet<sup>+</sup> control codes 78

<b>Chapter 4</b>	<b>PostScript Language for the LaserWriter IInt and LaserWriter IIntX</b>	<b>83</b>
	Basic operation of the LaserWriter IInt and LaserWriter IIntX	84
	Page types	84
	Manual feed	86
	Timeouts	87
	Idle-time font scan-conversion	88
	System parameters	90
	Changing persistent parameters	90
	Persistent parameters	91
	Additional persistent parameters	101
	Volatile parameters	102
	PostScript language changes	106
	Packed arrays	106
	Immediately evaluated names	107
	Font-cache operation	109
	Device-resolution images	109
	New operators	110
	Showpage and cöppage	113
<b>Chapter 5</b>	<b>LaserWriter Functional Overview</b>	<b>115</b>
	The LaserWriter IInt/IIntX controllers	116
	The controller and print engine	117
	Video interface	120
	LaserWriter and LaserWriter IInt functional modules	122
	LaserWriter IIntX functional block diagram	123
<b>Chapter 6</b>	<b>LaserWriter IIntX Font-Expansion Card Specification</b>	<b>127</b>
	Font expansion card dimensions	128
	Interface connection	129
	Power consumption	131
	Timing	131
	Memory	131
	Environment	132

## **Appendix A Serial-Data Communication 136**

- Using the RS-422 port 138
- Using the RS-232C port 140
- Cable configuration 141
  - RS-232C cable configuration for IBM-compatible to LaserWriter 142
- Changing communication parameters 142
  - The operator setscbatch–LaserWriter and LaserWriter IINT 143
    - Channel 143
    - Baud 143
    - Parity 144
  - The setscinterative operator 145
- Controlling communication 145

## **Appendix B LaserWriter Technical Specifications 147**

## **Appendix C Laser Printer Controllers 151**

- Controller models 151

**Glossary 153**

**Index 163**

---

---

## Figures and tables

### Chapter 1 Introduction 1

- Figure 1-1 The LaserWriter printer 2
- Figure 1-2 The LaserWriter II printer engine 2
- Figure 1-3 Macintosh-LaserWriter software environment 8

### Chapter 2 Working with Fonts 13

- Figure 2-1 Font processing by the 64K ROM Font Manager 17
- Figure 2-2 Font processing by the 128K ROM Font Manager 18
- Figure 2-3 A printer font versus a bit-mapped font 20
- Figure 2-4 Structure of the 'FOND' resource 27
- Figure 2-5 Style mapping table 28
- Figure 2-6 Character-set-encoding table 30
- Figure 2-7 Style name table 31
- Table 2-1 Font classification 29
- Table 2-2 Macintosh-style bits 32
- Table 2-3 Style-code format for style mapping 32
- Table 2-4 Data types of downloadable font resources 37

### Chapter 3 Working In the Printing Environment 53

- Figure 3-1 Simple AppleTalk network 57
- Figure 3-2 Network sockets 58
- Table 3-1 LaserWriter operational-mode switch settings 55
- Table 3-2 LaserWriter IINT operational-mode switch settings 56
- Table 3-3 LaserWriter IINTX operational-mode switch settings 56
- Table 3-4 Interactive mode line-editing characters 60
- Table 3-5 Control characters for Diablo 630 emulation 67
- Table 3-6 Persistent parameters for the Diablo 630 emulator 68
- Table 3-7 The eescratch location assignments 69
- Table 3-8 LaserJet<sup>+</sup> control codes 79-82

<b>Chapter 4</b>	<b>PostScript Language for the LaserWriter IINT and LaserWriter IINTX 83</b>
Table 4-1	Font numbers 89
Table 4-2	setscbatch options 97
Table 4-3	Persistent parameters for the Diablo 630 emulator 101
<b>Chapter 5</b>	<b>LaserWriter Functional Overview 115</b>
Figure 5-1	The image transfer process 117
Figure 5-2	LaserWriter and LaserWriter IINT controller functional block diagram 118
Figure 5-3	LaserWriter IINTX controller functional block diagram 119
Figure 5-4	The paper path and the mechanical-component location (in the printer engine) 121
<b>Chapter 6</b>	<b>LaserWriter IINTX Font-Expansion Card Specification 127</b>
Figure 6-1	LaserWriter IINTX font expansion card dimensions 129
Figure 6-2	A simplified circuit design diagram for the font expansion card 133
Table 6-1	Pin description of the 96-pin DIN connector to the LaserWriter IINTX expansion board 130
<b>Appendix A</b>	<b>Serial Data Communication 136</b>
Figure A-1	The original LaserWriter and LaserWriter Plus serial connectors 137
Figure A-2	The LaserWriter IINT and LaserWriter IINTX serial connectors 138
Figure A-3	The 9-pin connector of the LaserWriter 139
Figure A-4	LaserWriter IINT and LaserWriter IINTX MINI DIN-8 connector 140
Table A-1	9-pin RS-422 port pin assignments 139
Table A-2	LaserWriter IINT and LaserWriter IINTX MINI DIN-8 RS-422 port pin assignments 139
Table A-3	LaserWriter RS-232C port pin assignments 140

---

---

## Radio and television interference

The equipment described in this manual generates and uses radio-frequency energy. If it is not installed and used properly—that is, in strict accordance with Apple's instructions—it may cause interference with radio and television reception.

This equipment has been tested and complies with the limits for a Class B computing device in accordance with the specifications in Subpart J, Part 15, of FCC rules. These rules are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that the interference will not occur in a particular installation, especially if a “rabbit-ear” television antenna is used. (A rabbit-ear antenna is the telescoping-rod type usually found on television receivers.)

You can determine whether your computer is causing interference by turning it off. If the interference stops, it was probably caused by the computer or its peripheral devices.

If your computer system does cause interference to radio or television reception, you can try to correct the interference by using one or more of the following measures:

- Turn the television or radio antenna until the interference stops.
- Move the computer to one side or the other of the television or radio.
- Move the computer farther away from the television or radio.
- Plug the computer into an outlet that is on a different circuit than the television or radio. (That is, make certain the computer and the radio or television are on circuits controlled by different circuit breakers or fuses.)
- Consider installing a rooftop television antenna with a coaxial cable lead-in between the antenna and the television.

If necessary, consult your authorized Apple dealer or an experienced radio-television technician for additional suggestions.

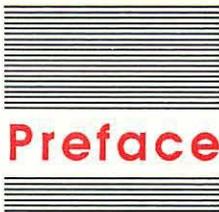
You may find helpful the following booklet, prepared by the Federal Communications Commission: “How to Identify and Resolve Radio-TV Interference Problems.” This booklet is available from the U.S. Government Printing Office, Washington, DC 20402.

---

### Important

This product was FCC-certified under test conditions that included use of shielded cables and connectors between system components. It is important that you use shielded cables and connectors to reduce the possibility of causing interference to radios, television sets, and other electronic devices. For Apple peripheral devices, you can obtain the proper shielded cable from your authorized Apple dealer. For non-Apple peripheral devices, contact the manufacturer or dealer for assistance.

---



## Preface

### About This Book

The *LaserWriter Reference* provides information for experienced computer users, system administrators, and programmers who want to take advantage of the features of the LaserWriter family of printers. To use this book, you should have some knowledge of a programming language, such as BASIC, Pascal, C, or assembly language, and you should be familiar with the computer with which you intend to use your LaserWriter printer.

You do *not* need to use this book when simply running packaged programs for your Apple® computer. However, this book can help you if you are running a program that can be configured to take advantage of the features of the LaserWriter, or if you are writing or modifying a program that uses the LaserWriter.

Instructions for connecting the LaserWriter to your computer, loading paper, changing toner cartridges, installing options, and other routine operating tasks are provided in the owner's guide that came with your printer.

This preface describes the contents of this book and the visual cues and conventions used throughout. It also lists some other books that you may wish to refer to when using this book.

---

---

## What this book contains

The *LaserWriter Reference* describes the operating modes and special capabilities of the LaserWriter, LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX as well as the hardware and software interfaces of these printers to the host computer.

- ❖ *Note:* Because the LaserWriter, LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX printers execute page descriptions in the PostScript® language, they are commonly dealt with in this book, with reference to specific models clearly indicated. Because the LaserWriter IISC processes images with QuickDraw and has no PostScript functionality, information on the LaserWriter IISC is provided in a separate reference.

The information in this book pertains to two releases of the LaserWriter driver:

- LaserWriter drivers with version numbers 3.0 and later
- LaserWriter driver enhancements provided in version numbers 4.0 and later

The following is a brief outline of the contents of this book.

- Chapter 1, "Introduction," provides a general description of the different LaserWriter features and discusses the software components that work together in the printing environment on the Macintosh®.
- Chapter 2, "Working with Fonts," provides an introduction to font terminology and structure for the Macintosh and LaserWriter. It also includes a section about the LaserWriter IINTX fixed-disk and file-system utilities.
- Chapter 3, "Working in the Printing Environment," describes the operational modes of the LaserWriter, LaserWriter IINT, and LaserWriter IINTX.
- Chapter 4, "PostScript Language for the LaserWriter IINT and LaserWriter IINTX," describes the features of version 4.7 of the Adobe PostScript Language in the LaserWriter IINT and LaserWriter IINTX.
- Chapter 5, "LaserWriter Functional Overview," provides an overview of the software and hardware interaction between the laser-printer controller and the print engine.
- Chapter 6, "LaserWriter IINTX Font Expansion Board Specification," provides the specification for designing a font expansion board for the LaserWriter IINTX.

- Appendix A, "Serial-Data Communication," describes the serial-communication ports on each of the LaserWriter printers. It also provides cable-configuration information and discusses the PostScript operators used to control serial-data transmission.
- Appendix B, "LaserWriter Technical Specifications," lists technical specifications for the LaserWriter.
- Appendix C, "Laser Printer Controllers" summarizes the design specifications for the Apple laser-printer controllers.

The table of contents provides a complete list of the subjects covered in each chapter. This book also contains a glossary of technical terms used in the book, and an index.

---

---

## How to use this book

This book provides information for programmers and developers with a variety of needs and levels of experience. The following guidelines can help you get the most out of this book.

- If you are not writing programs but are interested in learning more about laser printers in general, and about the LaserWriter IINT and LaserWriter IINTX printers in particular, read Chapters 1 and 3. Look through the rest of the book to get an idea of the kinds of tasks the LaserWriter printer can do. You might also want to read Chapter 2, Chapter 4, and Appendix A to learn about all the considerations involved in writing programs for your LaserWriter printer.
- If you are using this book to learn how to configure or customize a program to take advantage of the LaserWriter IINTX's features, read Chapter 4.
- If you have experience writing programs that use printers but are unfamiliar with the Apple laser printers, look through Chapters 2 through 4 to learn about the features and options available. Then go back to the detailed description of a command when you are ready to use it.
- If you are a hardware designer interested in building a font expansion board for the LaserWriter IINTX, read Chapter 6.

---

---

## Visual cues and conventions

Look for these visual cues throughout the book:

❖ *Note:* Notes like this contain supplementary information.

---

### Warning

Warnings like this direct your attention to something that could damage either software or hardware or result in loss of data.

---

Terms in **boldface** type are defined in the Glossary.

A special typeface is used for characters that you type or for lines of program code:

It looks like this.

In general, commands are sent to the printer with no spaces between characters in the command. Spaces between characters in sample commands in this book are for legibility only. When the space character is included as part of a command, it is shown as SPACE.

**Hexadecimal** numbers are preceded by a dollar sign (\$), except in tables where hexadecimal numbers are clearly labeled. For example, the hexadecimal equivalent of decimal 16 is written as \$10.

---

---

## Where to look for more information

The *LaserWriter Reference* assumes that you are familiar with the information presented in these documents:

- The *LaserWriter and LaserWriter Plus* owner's guide, which is shipped with every LaserWriter and LaserWriter Plus, explains how to set up a LaserWriter in the standard configuration as a network printer connected to the AppleTalk® network system. The manual gives basic operating information, such as how to load paper and how to change toner cartridges.

- The *LaserWriter IINT/NTX Owner's Guide*, which is shipped with every LaserWriter IINT and every LaserWriter IINTX, explains how to set up a LaserWriter IINT or LaserWriter IINTX in the standard configuration as a network printer connected to the AppleTalk network system. The manual gives basic operating information, such as how to load paper and how to change toner cartridges.
- The *PostScript Language Reference Manual* describes the programming language that tells the LaserWriter what and how to print. The manual is published by the Addison-Wesley Publishing Company and is available in bookstores.
- The *PostScript Language Tutorial and Cookbook* introduces the PostScript programming language and provides a collection of useful PostScript programming examples. The manual is published by Addison-Wesley.
- The "Printing Manager" chapter of *Inside Macintosh*, Volumes I, II, and IV, describes how to print from a Macintosh application using the generic printing-manager/printer-driver interfaces.

The "Font Manager" chapter of *Inside Macintosh*, Volumes I, IV, and V, describes fonts and their structures, and how the Printing Manager communicates with the Font Manager.

The "QuickDraw" chapter of *Inside Macintosh*, Volumes I, IV, and V describes how QuickDraw operates to create and to display text and graphics.

*Inside Macintosh* is published by Addison-Wesley.

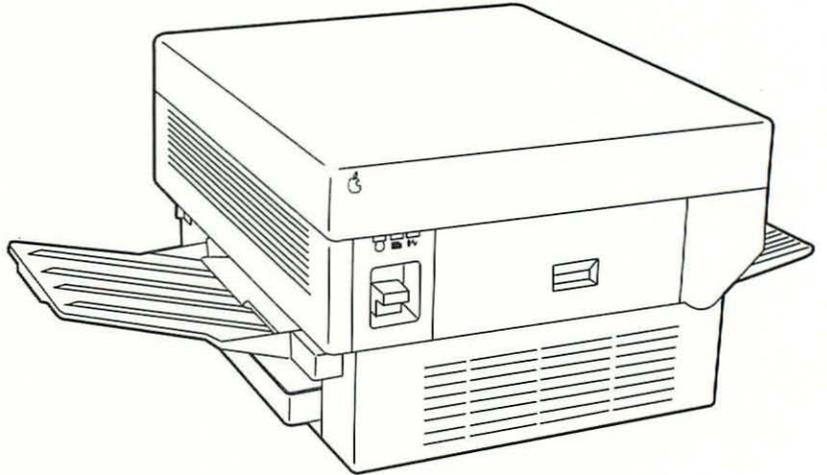


# Chapter 1

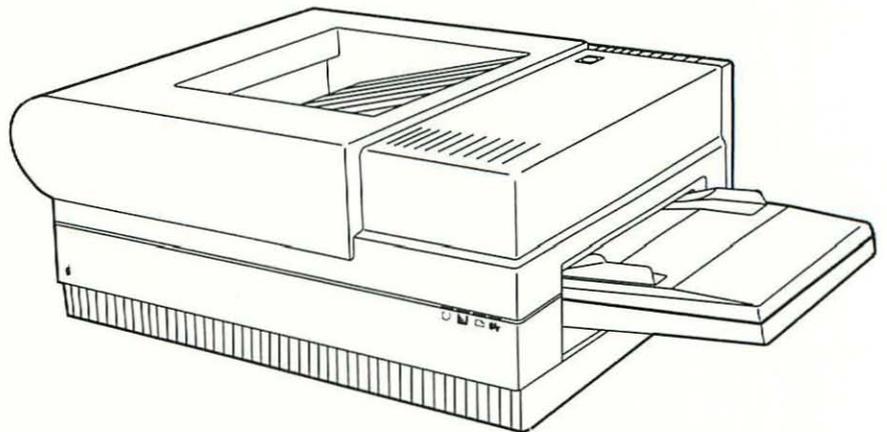


## Introduction

The Apple LaserWriter and LaserWriter II, shown in Figures 1-1 and 1-2, are laser-scanning, xerographic printers driven by powerful intelligent controllers. The LaserWriter and LaserWriter Plus printing engines are controlled by built-in controllers, whereas the LaserWriter IINT and LaserWriter IINTX laser printers consist of a printer engine configured with a plug-in controller. These printers print at a maximum rate of 8 pages per minute with a resolution of 300 dots per inch (dpi).



**Figure 1-1**  
The LaserWriter printer



**Figure 1-2**  
The LaserWriter II printer

---

---

## Laser printer models

Apple laser printers are available in five models:

- LaserWriter
- LaserWriter Plus
- LaserWriter IISC
- LaserWriter IINT
- LaserWriter IINTX

The LaserWriter IISC, LaserWriter IINT, and LaserWriter IINTX printers are part of a family of easily upgradable Apple laser printers. Because these printers share a common printing engine, built to accept plug-in controller boards, you can upgrade the LaserWriter IISC to a LaserWriter IINT or LaserWriter IINTX, and the LaserWriter IINT to a LaserWriter IINTX. Upgrades are available from Apple dealers.

---

### The LaserWriter and LaserWriter Plus

The LaserWriter and LaserWriter Plus printers are controlled by a Motorola MC68000 microprocessor. The LaserWriter has 0.5 megabytes of ROM and 1.5 megabytes of RAM. The LaserWriter Plus differs from the LaserWriter in that it has 1.0 megabyte of ROM, which contains additional built-in fonts.

---

### The LaserWriter IISC

The LaserWriter IISC differs from the LaserWriter, LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX in functional concept and controller design. Additionally, the LaserWriter IISC does not contain a PostScript interpreter in ROM for processing page descriptions, but uses QuickDraw for all image processing for printing (as discussed in the *Apple LaserWriter IISC Reference*).

---

## The LaserWriter IINT and LaserWriter IINTX

The LaserWriter IINT has the same functionality as the LaserWriter Plus. However, while a LaserWriter IINT contains a PostScript interpreter, it differs in hardware design. The LaserWriter IINT consists of a print engine configured with a LaserWriter IINT controller. The LaserWriter IINT controller uses a Motorola MC68000 microprocessor, like the LaserWriter and LaserWriter Plus, and contains 1 megabyte of ROM, and 2 megabytes of RAM.

Like the LaserWriter IISC and LaserWriter IINT, the LaserWriter IINTX printer contains a print engine with a LaserWriter IINTX controller board installed. The LaserWriter IINTX controller board uses a 16-MHz Motorola MC68020 microprocessor (the same processor in the Macintosh II), expandable ROM, and can be configured with 2 to 12 megabytes of RAM.

Both the LaserWriter IINT and LaserWriter IINTX execute page descriptions written in the PostScript language and produce printed output by using the same raster printing technology as the LaserWriter. And like the LaserWriter and LaserWriter Plus, the LaserWriter IINT and LaserWriter IINTX can be used as dedicated printers accessed by a RS-232 serial port or as shared printers on an AppleTalk local-area network.

The LaserWriter IINTX high-speed controller board provides faster processing through put of the PostScript page descriptions (approximately four times faster) than the original LaserWriter, and has a connection for an optional SCSI hard disk with a file system for font caching. The board also has a slot for installing a font expansion board. For information about the disk file system, see "LaserWriter IINTX Font Cache and Disk File System," in Chapter 3. For hardware developers interested in producing a font expansion board for the LaserWriter IINTX, see Chapter 6.

Both the LaserWriter IINT and LaserWriter IINTX support the PostScript language as specified in Adobe Systems's *PostScript Language Reference Manual*. Any PostScript programs written for the other LaserWriter models are fully functional in the LaserWriter IINT and LaserWriter IINTX environment.

The LaserWriter IINTX controller can be configured with 2 to 12 megabytes of RAM. The larger RAM sizes provide additional virtual memory used while constructing the page image and for a larger font cache; the additional memory increases printer throughput. LaserWriter IINTX RAM upgrade kits are available from Apple dealers.

The LaserWriter IINT and LaserWriter IINTX support serial-communication rates up to 57,600 baud. Appendix A, "Serial-Data Communication" contains both RS-232, RS-422, and MINI DIN-8 serial port specifications for LaserWriter.

❖ *Note:* In this book, the term *LaserWriter* refers to the LaserWriter, LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX. The terms *original LaserWriter*, *LaserWriter Plus*, *LaserWriter IINT*, and *LaserWriter IINTX* are used when necessary to distinguish between the models. The LaserWriter IISC printer and its software environment are discussed in a separate manual, the *LaserWriter IISC Reference*.

---

## LaserWriter ROM features

The original LaserWriter, LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX ROM firmware features are as follows.

### Original LaserWriter, LaserWriter Plus, and LaserWriter IINT ROM features

- Adobe Systems's PostScript page-description language
- diagnostic routines
- AppleTalk routines
- routines to emulate the Diablo 630 printer
- fonts

### LaserWriter IINTX ROM features

- all of the features just listed
- fixed-media disk support via the LaserWriter IINTX external SCSI interface (primarily for font cache)
- routines to emulate the Hewlett-Packard LaserJet<sup>+</sup> printer
- ROM expansion through a font expansion card

The RAM can contain additional fonts and PostScript code **downloaded** by the printer driver, your application, or the user.

---

## LaserWriter fonts

The original LaserWriter, LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX have certain built-in fonts, as described next.

### The original LaserWriter

The original LaserWriter has four built-in **font families**:

- Courier
- Helvetica®
- Symbol
- Times®

### The LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX

The LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX have a total of eleven built-in fonts. In addition to the four just listed, they have the following seven fonts.

- Helvetica Narrow
- ITC Avant Garde®
- ITC Bookman®
- ITC Zapf Chancery®
- ITC Zapf Dingbats®
- New Century Schoolbook
- Palatino®

You can add more built-in fonts to the LaserWriter IINTX through a font expansion board.

---

---

## The printing environment

LaserWriter printers execute PostScript programs, or jobs, sent to them from another computer. During normal operation, a LaserWriter continually cycles through three steps:

1. It sets up a clean initial environment in **virtual memory** (RAM) for executing the PostScript print job and constructing the page image.

2. It receives the job over an AppleTalk or serial-communication channel (see "How You Communicate," given next), simultaneously interpreting program code and executing the job.
3. Finally, when it encounters an end-of-job or error condition, it resets the virtual memory to its initial state in preparation for the next job.

---

## How you communicate

LaserWriter printers are connected to host computers via either an AppleTalk network or a point-to-point RS-232C serial link. (See Chapter 3, "Working in the Printing Environment," and Appendix A, "Serial Data Communication," for software and hardware configuration information relating to serial communication.) Multiple host computers can communicate over the AppleTalk network connection, but only a single host may use the RS-232C or RS-422 serial link. The connection between the host computer and a LaserWriter is called the **communication channel** (or simply **channel**). The host uses the channel to send data and PostScript programs to the LaserWriter. In a point-to-point configuration, the host device at the other end of the channel can also be either a terminal operated directly by a user or a Macintosh computer running MacTerminal (or another terminal-emulation program).

---

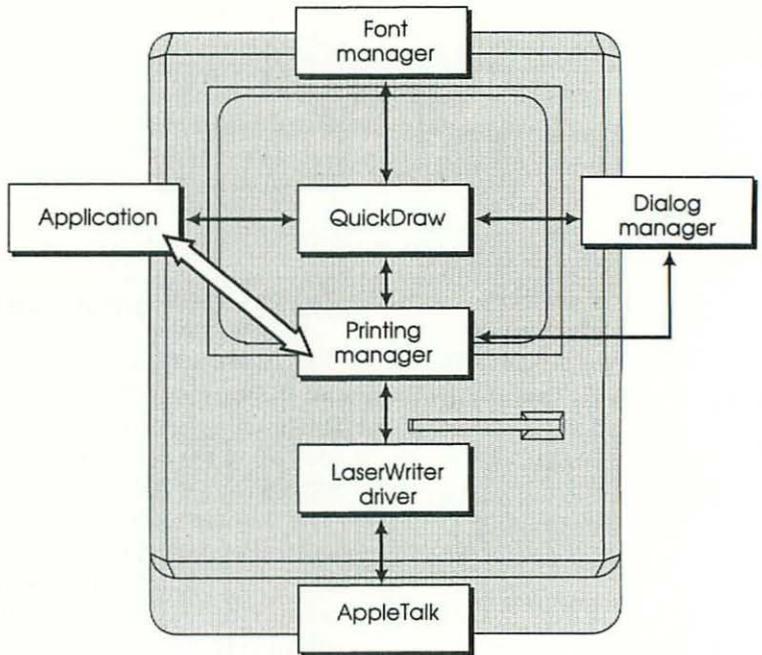
## Operating modes

The LaserWriter operates in one of three modes: batch, interactive, or emulation. (For a complete description of the LaserWriter operating modes, see Chapter 3.)

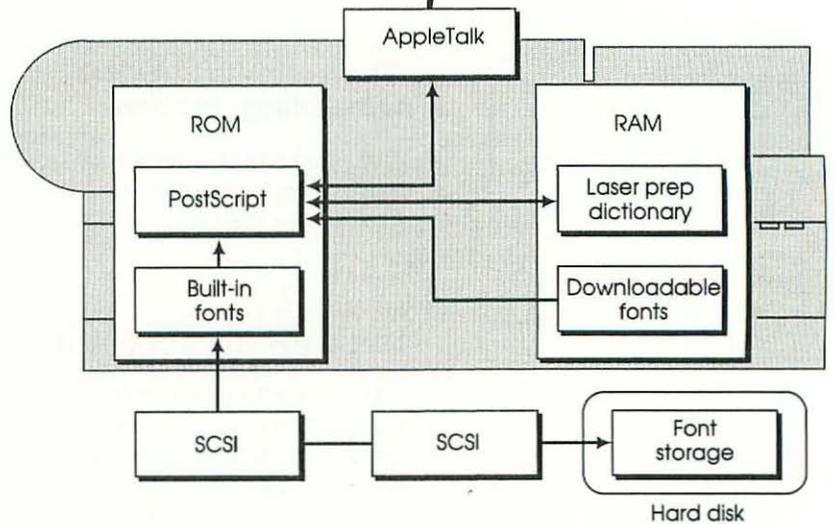
---

## Software environment

Many programs in the Macintosh and the LaserWriter interact to print a Macintosh file on the LaserWriter. Among these programs are your application, the PostScript interpreter, QuickDraw, the Font Manager, the Printing Manager, the LaserWriter driver, and AppleTalk code. Figure 1-3 shows some of these programs and their relationships.



As Figure 1-3 shows, your application can control the printing process at several levels if necessary. However, most applications only need the control provided by the Printing Manager (as indicated in the figure by the wide arrow).



**Figure 1-3**  
Macintosh-LaserWriter software environment

## PostScript

PostScript is a page-description language for describing text, graphic entities, and digitized images for printed pages. You can also use PostScript to control aspects of a printer's behavior. PostScript is designed for use with high-resolution printers and typesetting equipment. The LaserWriter executes PostScript page descriptions and commands to construct the image to be printed.

**Page description:** PostScript programs are typically generated by application programs running on a host computer. For example, when normally printing with the Macintosh Printing Manager, an application uses QuickDraw commands to describe the image to be printed. The LaserWriter driver then translates the lower-resolution (72 dpi) QuickDraw text and graphics descriptions into higher-resolution (300 dpi) PostScript programs to print on the LaserWriter.

The PostScript language allows you to achieve effects not available through QuickDraw, such as printing rotated text. You can also send PostScript programs to the LaserWriter for execution by using commercially available downloading programs.

**Printer control:** Normally, PostScript programs are executed to print pages. However, you can use PostScript to inquire about or to change the values of some parameters in the LaserWriter itself, such as communication modes (baud or parity settings, for example), and to initiate printer emulation modes (Diablo 630 or Hewlett-Packard LaserJet<sup>+</sup>). You can also use PostScript to perform some computation whose results are sent back over the communication channel. (For more information about the PostScript language, see the Preface for a list of books by Addison-Wesley.)

## QuickDraw

QuickDraw performs all Macintosh graphic operations (including text depiction), both on and off screen. The Macintosh's QuickDraw interface is a set of ROM-resident general-purpose routines that are accessed by your application and by other parts of the Macintosh toolbox, such as the Printing Manager. QuickDraw uses mathematical constructs to create a wide variety of graphics for output onto the screen or other devices, such as printers. Before drawing text, QuickDraw must call the Font Manager to obtain information about character sizes and styles.

Because the LaserWriter executes PostScript, you must translate the Macintosh QuickDraw commands, which describe the graphics and text, before the LaserWriter can execute them. This translation is the job of the LaserWriter driver, described later.

## Font Manager

Whenever QuickDraw needs to draw text, it calls the Font Manager. The Font Manager controls specific information about how to draw text characters, including character widths, style-implementation algorithms, and kerning details. It also ensures that the LaserWriter uses the most appropriate font in a particular situation.

The Macintosh-LaserWriter environment divides fonts into two basic categories: bit-mapped fonts and outline fonts. **Bit-mapped fonts** reside within the Macintosh System file and are designed to provide the best appearance on the Macintosh display screen. Bit-mapped fonts also provide high-quality printed output on non-PostScript printers such as the ImageWriter, ImageWriter II, ImageWriter LQ, and LaserWriter IISC. PostScript uses outline fonts to describe characters printed on the LaserWriter. The Macintosh screen has a resolution of only 72 dpi, whereas the LaserWriter prints 300 dpi. Consequently, outline fonts are usually defined differently than bit-mapped fonts. (For more information about fonts and the Font Manager, see Chapter 2, "Working with Fonts," in this book and the "Font Manager" chapters of *Inside Macintosh*, Volumes IV and V.)

## Printing Manager

Because of the wide variety of current printing technologies, a general-purpose printer driver would be either extremely limiting or impossibly large. Nevertheless, to be useful, a system must provide a standard printing interface for a number of printers. The Printing Manager is that standard interface for printers connected to the Macintosh. The Printing Manager is a set of routines that allows an application to use QuickDraw to print on a printer without regard to device-specific details.

The Printing Manager performs the functions common to most print jobs. For example, it calls the Dialog Manager to allow the user to specify such parameters as page ranges, the number of copies, and the paper source. The Printing Manager also supervises the transfer of files to the printer.

The Printing Manager selects the appropriate printer driver for the printer to be used. Printer drivers provide the necessary translation of application output into control codes for the selected printer device. A printer driver provides communications between the Printing Manager and the printer, thereby providing device independence. For example, the LaserWriter driver translates QuickDraw pictures into PostScript programs. (For more information about the Printing Manager and printer drivers, see the "Printing Manager" chapter of *Inside Macintosh*, Volume II.)

### **LaserWriter driver**

The LaserWriter driver is a Macintosh interface to the printer. It performs such functions as monitoring the status of print jobs, ensuring the necessary fonts are available on the printer, and translating QuickDraw into PostScript commands.

The Printing Manager usually calls the LaserWriter driver to perform printing and to control the printer. However, your application can access the driver directly to alter the normal printing process.

### **AppleTalk**

The AppleTalk network provides for efficient communication between the Macintosh and the LaserWriter. Some AppleTalk code resides in both the Macintosh and the printer. Everything that an application sends to the LaserWriter is normally sent through AppleTalk. (For more information about AppleTalk, see Chapter 3, "Working in the Printing Environment" and the *AppleTalk Reference*.)

---

---

## Approaching your application

If you are developing a Macintosh application, you should make every effort to use the Printing Manager when possible. The Printing Manager is designed to work with other Macintosh Toolbox managers to serve as a simple, general-printer interface. It enables your application to use a wide variety of printer devices, and helps you to avoid the pitfalls of device dependence. By using this standard interface, you help ensure the longevity and usability of your product. You also improve its upgrade path to new Macintosh hardware and software products. For example, many applications that use the Printing Manager to print on the ImageWriter require no revision to print on all of the other Apple printers.

Unfortunately, the gains of using a standard interface are achieved at some loss of flexibility. This loss of flexibility presents no problem in most cases. However, occasionally, you may have to modify or to bypass parts of the standard interface to take full advantage of the unique capabilities of PostScript in the LaserWriter printers. For example, you could alter or add to the standard printing dialogs, or even generate your own PostScript instructions to be passed to the LaserWriter.

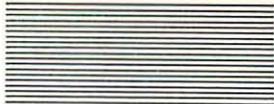
---

**Warning** There are very few operations that require bypassing the Printing Manager. If your application implements device-dependent features, it may be incompatible with future Apple printer products.

---

Read the font-description information provided in Chapter 2 if your application manipulates text to be printed on the LaserWriter or if you are developing fonts for the LaserWriter.

Read Chapter 3 if your application generates files to print on a Diablo 630, Hewlett-Packard LaserJet<sup>+</sup>, or compatible printers, or if you are simply interested in the environment in which the LaserWriter performs its job.



## Chapter 2



# Working With Fonts

This chapter provides an in-depth discussion of fonts. First, it describes basic font concepts and terminology for both the Macintosh and the LaserWriter, and then it describes the font-structuring techniques used specifically for the LaserWriter. Also included is a section that provides a synopsis of the PostScript operators necessary to implement a disk-based file system on disk drives attached to the LaserWriter IINTX for use as additional virtual memory for font cache and other user utilities.

Like traditional publishing systems, the Macintosh-LaserWriter desktop publishing system uses fonts to display text. The rich variety of fonts available for both the Macintosh and the LaserWriter is one of the most important features of Apple's desktop publishing system.

Fonts for the Macintosh and the LaserWriter are implemented as software entities that describe how characters are formed. Fonts designed to be displayed on the Macintosh screen are known as *bit-mapped fonts*. Fonts designed for printing on the LaserWriter are known as **printer fonts**. The LaserWriter driver matches bit-mapped and printer fonts to provide a *what-you-see-is-what-you-get* (WYSIWYG) publishing system.

In addition to its built-in fonts, the LaserWriter uses downloadable fonts. These are fonts stored externally (on the host disk and not in printer ROM) and downloaded to the LaserWriter as they are needed. A single document can use both built-in and downloadable fonts. However, the process of downloading fonts can be complex, especially when the LaserWriter driver must match those fonts to the bit-mapped fonts chosen by the user. The driver must match several characteristics of the two fonts. Those font characteristics are described in this chapter.

To provide a richer variety of fonts, the 128K ROM Font Manager processes fonts differently from the 64K ROM Font Manager. This chapter notes the differences throughout. Additionally, LaserWriter drivers with version numbers prior to version 3.0 and LaserWriter drivers with version numbers 3.0 and later use slightly different methods of processing fonts. If you are creating a downloadable font for the LaserWriter or if your application downloads fonts, you need to understand these methods.

---

---

## Basic font concepts

To work with fonts in your application, you need to understand some basic font concepts. These concepts include the structure of a bit-mapped font, the structure of a printer font, and how the LaserWriter driver matches the two types of fonts. You also need to understand some basic font terminology. (The "Font Manager" chapter of *Inside Macintosh*, Volumes I, IV, and V gives a detailed introduction to fonts.)

---

### Font terminology

The use of terms that relate to fonts varies somewhat between the typewriter, traditional typeset printing, and computer industries. The following definitions are those used in this manual and *Inside Macintosh*.

- **Font:** A specific set of characters in a single size and style. For example, 12-point Geneva italic is a font. A particular font may be intrinsic or derived.
- **Font family:** A complete set of characters for one font, including all styles and sizes of the characters in that font. For example, the Geneva font family includes 9-point to 36-point characters in italic, bold, outlined, and other styles.
- **Intrinsic font:** A font whose characteristics are entirely defined in a 'FONT' or 'NFNT' resource. The plain-style, or Roman, font of any family is an intrinsic font. Other styles may or may not be intrinsic. QuickDraw or the LaserWriter can use an intrinsic font without modification.
- **Derived font:** A font whose characteristics are partially determined by modifying an **intrinsic font**, described earlier. A derived font might be one whose characters either are scaled from an intrinsic font to achieve a desired size or are slanted to achieve an italic style.
- **Leading:** The space between lines of text. The term *leading* (pronounced *LED-ing*) is derived from the lead strips that typesetters used to separate rows of metal type.
- **Size:** The vertical measurement of a font in points, equal to the height of the font rectangle (the coordinate system that encloses the entire character) plus the leading. One point is  $\frac{1}{72}$  of an inch.

- **Style:** The characteristics other than size that uniquely define the fonts of a single font family. Geneva bold and Geneva italic are two styles of the Geneva font family.

---

## Bit-mapped fonts

Any black-and-white image can be displayed as a collection of black-and-white dots. These dots are called **picture elements**, or **pixels**. Fonts for electronic publishing are data structures that specify arrangements of pixels. The Macintosh displays pixels on its screen as tiny squares at 72 pixels per linear inch. This size coincides with the definition of a printer's point— $\frac{1}{72}$  inch. In contrast, the LaserWriter can print 300 pixels per inch.

Fonts that appear on the Macintosh screen are logically defined as **bit maps**: a bit image of each character in the font is stored in memory. When the user types a character, each pixel of the character is drawn on screen as specified by the corresponding bit in memory. The bit-map approach works well for a screen display, but storing a bit image for a single font can require as much as 30K or more bytes. Enabling the user to specify different sizes of characters freely requires either a bit map for each size or a mechanism for enlarging and reducing bit images. (In reality, the Macintosh uses both methods for screen displays.)

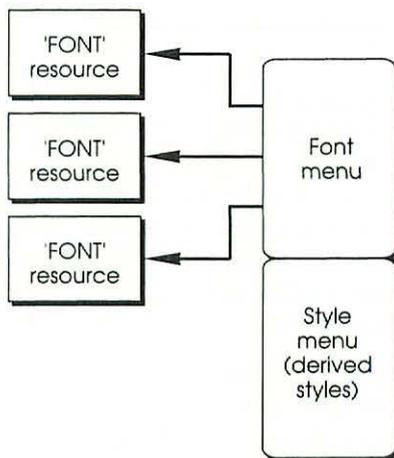
The Macintosh provides a wide variety of font families with a well-defined set of styles that is uniformly available across all families. For each plain bit-mapped font, the user may select up to seven distinct styles provided by QuickDraw:

- bold
- italic (or oblique)
- underline
- outline
- shadow
- condensed
- extended

Often the plain style is the only style available as an intrinsic font. QuickDraw derives the other styles, just listed. However, styles other than the plain style can be intrinsic rather than derived.

Each intrinsic font is defined in a resource of type 'FONT' or 'NFNT'. The two resources have the same format, but are processed differently. All fonts in 'FONT' resources appear in the Font menu (within the restrictions of the applications). For example, if you have an intrinsic Times plain font and an intrinsic Times italic font defined in 'FONT' resources in your system resource file, they both appear in the Font menu.

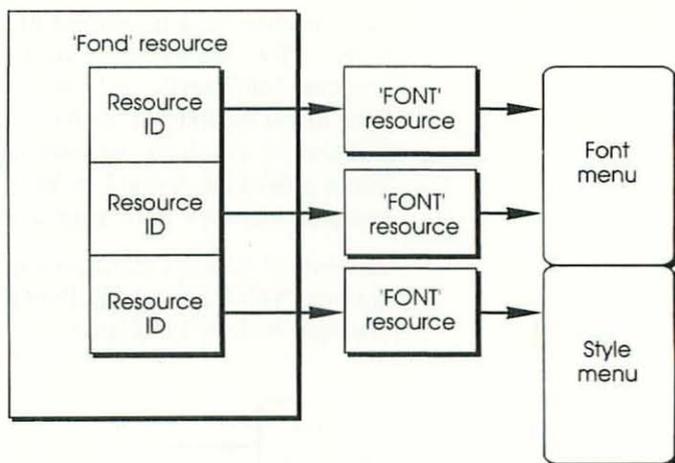
The 64K ROM Font Manager can process only 'FONT' resources. It ignores 'NFNT' resources. Font processing by the 64K ROM Font Manager is shown in Figure 2-1.



**Figure 2-1**  
Font processing by the 64K ROM Font Manager

If you have the same Times plain font in the 'FONT' resource and the Times italic font defined in an 'NFNT' resource in your System file, the Times italic font does not appear in the Font menu. However, when you select italic from the Style menu, the Font Manager uses the intrinsic Times Italic font defined in the 'NFNT' resource. This method of processing fonts allows you to use a number of intrinsic fonts without cluttering up your menus.

A font family is defined in a 'FOND' resource. The 128K Font Manager uses this font family resource to identify all the fonts in a family, whether they are specified in a 'FONT' resource or an 'NFNT' resource. Part of this 'FOND' resource is a font association table. The font association table contains the identification of each 'FONT' resource and 'NFNT' resource that contains a font in the family. Basic font processing by the 128K ROM Font Manager is shown in Figure 2-2.



**Figure 2-2**  
Font processing by the 128K ROM Font Manager

If the 128K ROM Font Manager finds a 'FOND' resource, it uses the resource exclusively to determine which fonts are available. It is crucial that every new font have an associated 'FOND' resource. The 'FOND' resource format is defined in "Classifying fonts" in this chapter.

The LaserWriter driver uses the font information from the 'FOND' resource to find a printer font that matches the size and appearance of the bit-mapped font and gives the best-quality printed copy.

---

## LaserWriter printer fonts

The Adobe fonts in the LaserWriter are not defined as bit maps. The image of a character is defined as a series of *Bézier curves*, or **B-splines**. These curves are stored as mathematical constructs that form the outline of the character. The LaserWriter processes the constructs to draw the character outline, and then simply fills it in.

This type of character definition offers several advantages:

- Drawing the image of the character takes much less time than constructing the image from a bit map.
- The sizes of the curves are easily reduced or enlarged (scaled) to produce a clear image of the character, regardless of its size.

- Because one definition specifies all sizes of a character, less memory is required to store many sizes of a font. Character definitions are stored as 1-point fonts.
- The definition is device independent. It can be reproduced on any PostScript printer. The resolution of the printer determines the quality of the printed image.

The LaserWriter has a number of built-in printer font families in its ROM, including several intrinsic bold and italic fonts. In addition, the LaserWriter can use new fonts that are downloaded to it either on a once-per-document basis, which is called *temporary downloading*, or for use on any document until the printer is switched off, which is called *permanent downloading*. The user can have as many downloadable fonts on the LaserWriter as its virtual memory allows. Depending on the size of the downloadable font files, the limit is usually between two and five fonts.

At the beginning of every document, the LaserWriter driver asks the LaserWriter to list all of the fonts it has. (For an explanation of this process, see the “Obtaining the Font List” in this chapter.) The driver stores this information in a temporary font cache. Whenever the driver encounters a new font in a document, it checks this cache. If the desired font is in the cache, the driver switches the printer to that font. If the desired font is not in the cache, the driver searches the disk or disks on the system for a font file to download to the printer. If the driver finds an appropriate printer font, it downloads the font to the printer and enters the font name in the cache. (For information about using a disk attached to the LaserWriter IINTX for additional font cache, see “LaserWriter IINTX Font Cache and Disk-File System” in this chapter.)

If the driver does not find a printer font, it gets a bit-mapped version of the font and downloads that to the printer. At this point, the differences between bit-mapped and printer fonts become obvious. If the user selects a character in a size that is not defined in a 'FOND' resource or a 'FONT' resource, QuickDraw attempts to resize the character by a method called *scaling*. The result of this process can sometimes be less than spectacular. The character can be significantly distorted, with rough curves and jagged edges. This distorted image is sent to the LaserWriter, which accurately reproduces the bit map on the printed page. Figure 2-3 shows the difference in quality between a true printer font and a scaled bit-mapped font. As in the case of intrinsic fonts, the driver enters the font name in the temporary font cache.

**This is an example of 18-point Times  
as displayed on the Macintosh screen.**

This is an example of 18-point Times  
as printed on a LaserWriter IINT.

**Figure 2-3**

A printer font versus a bit-mapped font

When requesting the font inventory from the printer, the driver receives information about permanently downloaded fonts as if they were built-in fonts. It uses a permanently downloaded font in the normal manner—provided the font is defined and classified correctly, and is matched through an appropriate 'FOND' resource to a corresponding bit-mapped font on the Macintosh (see “Matching Bit-mapped and Printer Fonts” later in this chapter).

If you are creating a downloadable font, you do not have to use a Bézier form. You can arbitrarily define a font as long as it conforms to Macintosh and PostScript conventions. Bit-mapped fonts are just one example of fonts that can be defined and downloaded. (For detailed information about fonts, see the “Font Manager” chapters of *Inside Macintosh*, Volumes I, IV, and V. For details on the downloadable font structure, see the *PostScript Language Reference Manual*.)

---

## Obtaining the font list

At the beginning of a print job, the LaserWriter driver asks for the list of all known fonts on the printer. The query begins with the string

```
%?fontlist
```

followed by the necessary PostScript code, and then the string

```
%?end
```

Here is an example of the query:

```
%?fontList
md begin
lsf
end
%?end
```

The driver expects the printer to respond with a list of fonts. The anticipated response differs according to the version of the driver used. Fonts in the font list can be either **coordinated** or noncoordinated. A font with a proper class assignment and font name is called a *coordinated font*. A font without both of these attributes is called a *noncoordinated font*.

- ❖ *Note:* Each font string transmitted by AppleTalk is encapsulated in a single PAP packet by executing the PostScript `flush` operator on the printer. Use this convention for all strings the spooler sends to the driver. This convention helps the driver distinguish between status and data in the responses it receives.

LaserWriter drivers prior to version 3.0 expect the response to the query to be a list of coordinated fonts. Drivers with version numbers of 3.0 or greater accept both coordinated and noncoordinated font names as valid responses. Apple suggests that your application's documentation recommend the use of LaserWriter drivers with version numbers 3.0 or later. The later drivers recognize the difference between the two types of font names by the use of the vertical bar character (|) at the beginning of coordinated names. For example, the response to a font-list query might be as follows:

```
% Sample List for LaserWriter Plus
NewCenturySchlbk-Bold
Times-BoldItalic
Palatino-Italic
Bookman-Light
Helvetica-Narrow-Oblique
Helvetica-Bold
Helvetica-Narrow
Courier-Oblique
Times-Bold
Times-Roman
NewCenturySchlbk-BoldItalic
Palatino-Bold
Palatino-Roman
|_____Courier      % A sample preinitialized font
ZapfChancery-MediumItalic
ZapfDingbats
|_____Seattle
```

Courier-Bold  
AvantGarde-Book  
Palatino-BoldItalic  
Helvetica  
Bookman-LightItalic  
Times-Italic  
Courier-BoldOblique  
NewCenturySchlbk-Italic  
NewCenturySchlbk-Roman  
Helvetica-Narrow-BoldOblique  
Bookman-Demi  
Helvetica-Narrow-Bold  
Helvetica-BoldOblique  
Helvetica-Oblique  
AvantGarde-Demi  
Bookman-DemiItalic  
Symbol  
AvantGarde-BookOblique  
AvantGarde-DemiOblique

You can define both the coordinated and noncoordinated names of a font in the dictionary. If your application provides a print spooler, it should be able to distinguish between the two types of names, and it should never respond to a font-list query with the noncoordinated name of a font that has already been coordinated. When the LaserWriter driver receives noncoordinated font names in response to the font-list query, the driver attempts to coordinate them before continuing to process the document body. (Guidelines for naming fonts are presented in “Naming Fonts” in this chapter.) For more information about print spooling, see the *Print Spooling in an AppleTalk Network* reference manual.

---

## Coordinating noncoordinated fonts

One of the major new features of the LaserWriter driver with version numbers greater than 3.0 is the ability to configure fonts as necessary for any given document. If configuring fonts requires downloading or coordinating a font, the driver initiates the process. Two conditions signal the driver to coordinate a font. First, the font must be present (along with its 'FOND' resource information) on the Macintosh that initiates the print job. Second, in the font-list query, the name of that font must flag it as noncoordinated.

When the driver determines from the font-list response that it must coordinate a font, it generates the appropriate PostScript text to coordinate the given font, rename it, and make it semipermanent on the printer. The driver also coordinates fonts to be downloaded with a document. In this case, the font information is part of the document and, thus, is transparent to any intercepting print spooler. To coordinate the fonts in the font-list example provided earlier, the driver generates the following PostScript code:

```
serverdict begin 0 exitserver
md begin
% General format is...
% /Coordinated-Name /Previous-Name MacEncoding-Vector-flag rf
/|_____Times-Roman /Times-Roman T rf
/|_____Times-Bold /Times-Bold T rf
/|_____Times-Italic /Times-Italic T rf
/|_____Times-BoldItalic /Times-BoldItalic T rf
/|_____Helvetica /Helvetica T rf
/|_____Helvetica-Bold /Helvetica-Bold T rf
/|_____Helvetica-Oblique /Helvetica-Oblique T rf
/|_____Helvetica-BoldOblique /Helvetica-BoldOblique T rf
/|_____Courier-Bold /Courier-Bold T rf
/|_____Courier-Oblique /Courier-Oblique T rf
/|_____Courier-BoldOblique /Courier-BoldOblique T rf
/|_____Symbol /Symbol F rf
/|_____AvantGarde-DemiOblique /AvantGarde-DemiOblique T rf
/|_____AvantGarde-BookOblique /AvantGarde-BookOblique T rf
/|_____AvantGarde-Demi /AvantGarde-Demi T rf
/|_____AvantGarde-Book /AvantGarde-Book T rf
/|_____Bookman-DemiItalic /Bookman-DemiItalic T rf
/|_____Bookman-LightItalic /Bookman-LightItalic T rf
/|_____Bookman-Demi /Bookman-Demi T rf
/|_____Bookman-Light /Bookman-Light T rf
/|_____Helvetica-Narrow-BoldOblique /Helvetica-Narrow-BoldOblique T rf
/|_____Helvetica-Narrow-Oblique /Helvetica-Narrow-Oblique T rf
/|_____Helvetica-Narrow-Bold /Helvetica-Narrow-Bold T rf
/|_____Helvetica-Narrow /Helvetica-Narrow T rf
/|_____NewCenturySchlbk-BoldItalic /NewCenturySchlbk-BoldItalic T rf
/|_____NewCenturySchlbk-Italic /NewCenturySchlbk-Italic T rf
/|_____NewCenturySchlbk-Bold /NewCenturySchlbk-Bold T rf
/|_____NewCenturySchlbk-Roman /NewCenturySchlbk-Roman T rf
/|_____Palatino-BoldItalic /Palatino-BoldItalic T rf
/|_____Palatino-Italic /Palatino-Italic T rf
/|_____Palatino-Bold /Palatino-Bold T rf
/|_____Palatino-Roman /Palatino-Roman T rf
/|_____ZapfChancery-MediumItalic /ZapfChancery-MediumItalic T rf
/|_____ZapfDingbats /ZapfDingbats du fe
% Redefine encoding vector for above font
```

```
% encoding-position, name, operand
128 /a89 ce
129 /a90 ce
130 /a93 ce
131 /a94 ce
132 /a91 ce
133 /a92 ce
134 /a205 ce
135 /a85 ce
136 /a206 ce
137 /a86 ce
138 /a87 ce
139 /a88 ce
140 /a95 ce
141 /a96 ce
nf
```

Font coordination depends greatly on the nature of the font. The driver also performs coordination dynamically.

---

## Matching bit-mapped and printer fonts

An essential font-related task of a desktop publishing system is to match bit-mapped fonts displayed on screen with the appropriate printer fonts. Several factors complicate this task.

For example, in traditional typesetting, a font family may contain from 1 style to approximately 50 different styles. With QuickDraw, the Macintosh can provide more than 64 styles, but not necessarily the same styles as in typeset font families. For example, the Macintosh has only one bold weight per family, whereas some typeset families have more than four weights (light, demi, bold, heavy, ultra, and so on).

The large number and wide variety of characters that can exist in a font further complicate font matching. For example, fonts for Japanese Kanji typically have more than 3000 intricate characters, and fonts for Arabic include ligatures and characters whose forms change depending on their context.

Additionally, some styles may be derived in some fonts but may be intrinsic in others. To make matters worse, a derived style in one font family may be implemented differently than the same derived style in a different font family. This difference is true particularly for outline and shadow styles.

To handle all of these variations correctly, the driver uses **font classes**. Each font class uses a different method of implementing font styles. Different classes can also use different character-set-encoding schemes to define the characters that are available in the font. In addition to font classes, the printer driver uses **font names** to match bit-mapped and printer fonts.

For a downloadable font to be usable, it must be assigned to a font class and must have a correctly formatted name.

---

---

## Classifying fonts

As mentioned earlier, a font class provides the LaserWriter driver with two important types of information—the style-implementation method and the character-set-encoding scheme.

A **style-implementation method** is used to obtain the font style that the user specifies. The method may indicate that either the style is implemented in an intrinsic font, or the way in which the driver modifies an intrinsic font to achieve the derived style. Every font in a given class implements its styles according to the same methods.

The **character-set-encoding** scheme defines the name of each character and the location of that character within the font. The need for the character-set-encoding scheme arises because of the difference between the Macintosh character-set vector for the bit-mapped fonts and the Adobe Standard Encoding vector for the printer fonts.

With PostScript, you can arrange characters within encoding sets at your discretion to match other sets, such as the Macintosh character set. However, you cannot swap characters between character sets. For example, the characters ®, ©, ™, and Ⓜ are available in the Macintosh encoding set for the Symbol font. Because these characters are unavailable in the Adobe standard encoding scheme for Times and Helvetica fonts, the printer driver must switch automatically to the Symbol font to print these characters whenever it encounters them in a document. Fonts that require this type of character borrowing are called **reencoded fonts**.

- ❖ *Note:* Although reencoded printer fonts do not contain these characters, the associated bit-mapped font must contain them. Also, when a font switch like the one just described occurs, the LaserWriter driver does not alter the traits of the reencoded characters borrowed from the switched font to match those of the original font family. For example, serif Symbol characters remain serif characters, even if the original font is **sans serif**.

A font can also incorporate other encodings that are in neither the Adobe character set nor the Macintosh character set. A font with its own character coding defined intrinsically presents no problem. In such a case, the font creator makes the screen font to match the printer font in each character position, and can assign character positions within the font arbitrarily.

However, there are some fonts—such as Zapf Dingbats—that contain additional characters not assigned to any particular character position. (PostScript allows a font to contain an arbitrary number of characters—even more characters than the 256 available encoding positions.) The driver can assign these characters to normally unassigned positions in the encoding vector or to character positions already defined. To download such fonts, the driver must have access to the encoding information whenever it is ready to make the new character-set-encoding assignments. This chapter shows the character-set-encoding table in a figure later.

---

## Classifying fonts for LaserWriter drivers of version 3.0 or later

LaserWriter drivers, version 3.0 or later, use a style-mapping table (part of the 'FOND' resource) to match screen and printer fonts. The structure for the 'FOND' resource is shown in Figure 2-4.

- ❖ *Note:* If you are familiar with the 'FOND' resource format, be aware that it has been extended to include an entry for the font-bounding-box (FontBBox) table. The FontBBox is an array of four numbers in the user-coordinate system giving the lower-left  $x$ , lower-left  $y$ , upper-right  $x$ , and upper-right  $y$  of the font-bounding box. The FontBBox describes the smallest rectangle enclosing the shape that would result if every character in a given font style were placed one on top of the other with their origins coincident. The PostScript interpreter uses the FontBBox coordinates to make decisions about character caching and estimating line layout to ensure that font-kerning details are not clipped out of the image area during the printing process.

'FOND' resource	Bytes
Flags for font family (word)	2
Font family ID number (word)	2
ASCII code of first character (word)	2
ASCII code of last character (word)	2
Max. ascent expressed for 1 pt. font	2
Max descent expressed for 1 pt. font	2
Max. leading expressed for 1 pt. font	2
Max. width expressed for 1 pt. font	2
Offset to width table	4
Offset to kerning table	4
Offset to style-mapping table	4
Style property information	24
Int'l field of family record (reserved)	4
Version number (\$02)	2
Font-association table	2
Number of offsets minus 1	2+6*(#fonts)
Offset to FontBBox table	4
FontBBox table	n
Font-width table	p
Font style-mapping table	q
Font-kerning table	r

The FontBBox table must have an entry for each style available in the font family

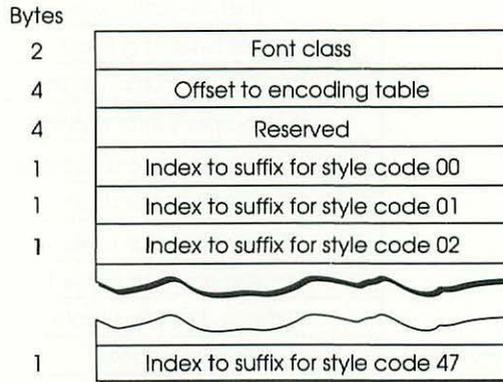
Bytes	Format of FontBBox table
2	# of FontBBox entries minus 1
10	First FontBBox entry
10	Second FontBBox entry
10	Third FontBBox entry

Bytes	Format of FontBBox entry
2	Style word
2	Lower-left x coordinate
2	Lower-left y coordinate
2	Upper-left x coordinate
2	Upper-left y coordinate

**Figure 2-4**  
Structure of the 'FOND' resource

## Style-mapping table

The style-mapping table provides a more flexible way to assign font classes and to specify character-set encoding. The table contains the font-class identification, character-encoding information, and a mechanism for obtaining the name of the appropriate printer font. That font can be a built-in printer font, or a temporarily or permanently downloaded font. Figure 2-5 shows the structure of the style-mapping table.



**Figure 2-5**  
Style-mapping table

The font-classification method of the Macintosh 128K ROM Font Manager provides a greater variety of font classes. However, with the 128K ROM method, neither the LaserWriter nor the LaserWriter driver provides the classification. Font classification is a property of the font itself, specified in the style-mapping table. This classification must be available to the printer driver through the 'FOND' resource either at the time the font is downloaded or at the time the font is accessed.

- ❖ *Note:* The Macintosh Font Manager maintains and controls 'FOND' and 'FONT' resources. Your application should never perform a release resource, a detach handle, or a purge handle operation on any 'FOND' or 'FONT' resource. However, applications may call `FMSwapFont` or `SetFontLock` with other Font Manager calls to avoid potential Font Manager problems.

The font class is available as the first entry in the style-mapping table in the form of a 16-bit integer. Table 2-1 shows the bit assignments of this integer.

**Table 2-1**  
Font classification

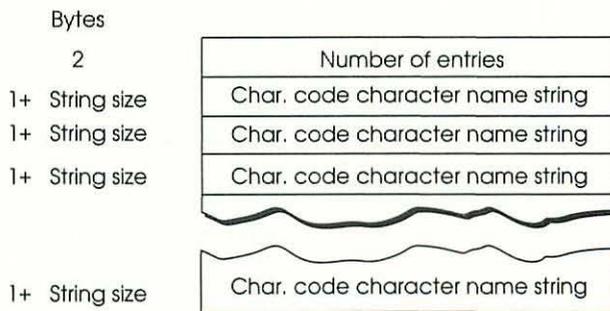
Bit set	Meaning
0	Font name needs coordinating
1	Macintosh vector reencoding scheme is required
2	Font has outline property by changing <code>PaintType</code> to 2
3	Disallows outlining simulation by smear and whiteout
4	Disallows emboldening by smear technique
5	Emboldening is simulated by increasing point size
6	Disallows obliquing for italic
7	Disallows automatic simulation of condensed style
8	Disallows automatic simulation of expanded style
9	Requires reencoding other than Macintosh vector encoding
10	Signifies the font family should have no additional intercharacter spacing other than the space character
11–15	Reserved

If no class is defined, the class definition defaults to class 0, which has default settings that indicate that the LaserWriter driver should derive the bold, italic, condensed, and expanded styles from the plain font. Intrinsic fonts are assigned classes that prevent the derivations from occurring (by using bits 2 through 8). Thus, the distinction between derived and intrinsic styles is apparent. Bits 11 through 15 of the class-definition integer are reserved; your application should not use them.

### Character-set-encoding table

If the font class indicates that the font must be reencoded, the second and third words of the style-mapping table are used as an offset to a **character-set-encoding table**. If the encoding for the font differs from the conventional Macintosh character set, if it does not contain its own font encoding, or if it needs to be altered in any other way, the driver derives the new encoding vector from this table. If the offset in the style-mapping table is empty, there are no extra encoding requirements.

The encoding table itself contains an integer length field, followed by an entry for every character position that the driver must reassign. Each character-position entry consists of a single-byte character code and a Pascal string that contains the PostScript character-name key (excluding the normal preceding slash). Figure 2-6 shows the format of the character-set-encoding table. Fonts that require reencoding must also be flagged as requiring coordination as well.



**Figure 2-6**  
Character-set-encoding table

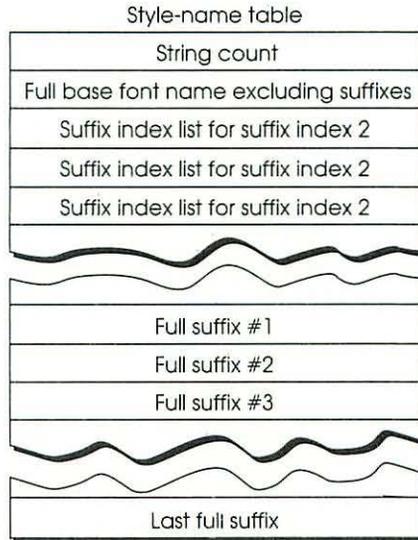
- ❖ *Note:* The offset to the character-set-encoding table is a relative offset from the beginning of the style-mapping table, and not from the beginning of the 'FOND'.

### Style-name table

Because there are so many more font styles available on the printer than on the Macintosh, all printer font families are first subdivided into subsets. There may be one or more subsets per family, including the subset that encompasses the entire family. Each subset is given a unique font name, which is usually a derivative of the font family name.

The user selects a screen font from the Macintosh font and style menus. For each selectable screen font, each subset contains a single font into which the LaserWriter driver will map that Macintosh font. Subsets can vary in size and can overlap within a family, but they can never overlap another font family, and they can never exceed more than one font per Macintosh font.

For every font that is available in a given style, there is a nonzero entry in the corresponding position in the table. That entry is an index that points to the **style-name table**, from which a unique printer font name is derived. Figure 2-7 shows the format of the style-name table.



**Figure 2-7**  
Style-name table

On the LaserWriter, the driver can generate many derived styles automatically by modifying existing fonts, as determined by the font class. Underline, shadow, and outline styles are derived this way and are relatively easy to generate dynamically. On the other hand, bold, italic, and bold italic styles are often not so easy to generate. Thus, these styles are usually obtained from intrinsic fonts. Some font families have *missing fonts*: fonts for which there is no supplied style and for which no derivation is required. For missing fonts, the corresponding style-name index entry in the style-mapping table is zero. If the user selects such a font, the plain style is used instead, without modification.

The style-name table contains a string list of the information necessary to produce the font name for that style. Entry number 1 in the string list contains the base font name to which 0, 1, or more suffixes—which are also contained in the table—may be appended to derive the full font name. Following the base font name are index strings that provide access to font-name suffix strings. Therefore, the string list contains both index lists (in the form of strings) and suffix strings.

Style mapping occurs by initially matching the selection from the Macintosh font menu to a 'FOND' resource that contains the mapping table for the given subset of the printer font family. The selected style is matched with a style code.

The normal Macintosh style bits are as shown in Table 2-2. However, style mapping omits the bit assignment for underline to provide the style-code format, as shown in Table 2-3.

**Table 2-2**  
Macintosh-style bits

Bit	Meaning
0	Bold
1	Italic
2	Underline
3	Outline
4	Shadow
5	Condense
6	Extend

**Table 2-3**  
Style-code format for style mapping

Bit	Meaning
0	Bold
1	Italic
2	Outline
3	Shadow
4	Condense
5	Extend

The bit list in Table 2-3 shows the numeric order of style entries in the style-mapping table; the list does not represent any actual data quantity. Because it resembles the actual Macintosh-style assignment in Table 2-2, you should notice the differences. It has been modified solely to optimize storage space.

For each style code, an entry in the style-mapping table provides the offset (in the style-name table) of an index number. The index number points to a numbered string in the table. This first string is itself an ordered list of index numbers that point to actual suffix strings in the same string list. The actual suffix strings are appended in the given order to the base font name to produce the full font name.

When a driver must derive a style, as specified by the font class, it is derived from the font specified in the style-mapping table for that style. Thus, the entry for the outlined italic style may contain the font name for the italic style, and the driver derives the outlining, if so specified in the class, from the italic style. As described earlier, when there is no entry (zero entry) for a given style in the style-mapping table, that style will be derived from, or replaced by, the plain style entry in the table. (Therefore, the plain style entry must never be empty or zero, and must always point to some name string.) Different style entries can point to the same font name.

---

---

## Naming fonts

The name that a user sees when selecting a font from the Font menu in an application is the name of either the 'FONT' or 'FOND' resource. The driver uses this name to match the font name on the printer. When a font is classified and named properly, it is considered to be coordinated. A font that is not properly classified or is not properly named (even though it may be syntactically correct to PostScript) is considered to be noncoordinated. Whether or not a font should be coordinated depends on the method and version of the driver you use to download it to the LaserWriter.

---

## Font-naming conventions for later drivers

Coordinated font names in LaserWriter drivers of version 3.0 and later are prefixed by the seven-character prefix

| \_ \_ \_ \_ \_ \_ \_

These font names retain the vertical bar, but have changed the hyphen characters to the underscore (hexadecimal 5A) and do not include the class number. The underscore character positions, though still valid for *B*, *I*, *O*, and *S*, are reserved for use by Apple. The application can never change these characters.

The name after the prefix is the name of the font derived from the style-mapping and style-name tables. The name is the normal name for the font on the printer, including all suffixes. Hyphens are considered to be separate suffixes, and spaces are not allowed. The name does not necessarily match the name given in the Macintosh Font menu, nor does it necessarily match other font names in the same font family. The name in the Font menu is used to access the appropriate 'FOND' resource. The 'FOND' resource contains the style-mapping table, which in turn points to the actual font name. For example, if the user selected the Courier font in both bold and italic styles, the coordinated name would be

```
|_____Courier-BoldOblique
```

---

---

## Downloading fonts

As mentioned earlier, the LaserWriter driver maintains a list of fonts that are available on the printer. If the driver discovers a font in a document that is not in that list, the driver attempts to download the correct font to the printer. The process of downloading fonts differs between the versions of the LaserWriter driver. To make things as simple and consistent as possible for your application, you should recommend that users install LaserWriter drivers of version 3.0 or later.

---

## Downloading with later printer drivers

Your application can load downloadable fonts for LaserWriter drivers of version 3.0 and later on either a temporary basis (automatically) or a permanent basis. The fonts need not be coordinated because the driver coordinates them from the information in the 'FOND' resource for the screen font. Every font that is intended for downloading must have a corresponding 'FONT' and 'FOND' resource on the Macintosh.

---

## Downloadable fonts for all printer drivers

For a font to be usable with all printer drivers, the font must be initially noncoordinated and must have a font name that does not contain any suffixes other than **Bold**, **Italic**, or **Oblique**. The font's class must be class 1 for drivers before version 3.0, and class 95 (hexadecimal \$005F) for drivers of version 3.0 and later. The font must be loaded before the Laser Prep dictionary for use with drivers before version 3.0; for version 3.0 and later, the font can be loaded permanently before or after the dictionary, or it can be temporarily downloaded as required.

---

## Temporary downloading of fonts

In addition to permanent downloading of a printer font, drivers of version 3.0 and later support temporary downloading. After encountering a new font in a document, the driver scans the printer font cache for that font. If it is found, a switch is made to that font on the printer.

If the font is not available on the printer, the Font Manager searches in the 'FOND' resource for that font. If there is no 'FOND' resource, QuickDraw creates the bit-mapped version of the font. If the 'FOND' resource is found and if it contains an entry in its style-mapping table for the appropriate style, the Font Manager searches the root directories of all on-line volumes for a filename that corresponds to that style entry. If the filename is found, the driver downloads the font; if not, a bit-mapped version is created.

The downloadable font file is a resource file whose name is derived from the first five characters (after the four characters, *ITC-*, if present) of the full base font name in the style-mapping table of the 'FOND' resource. These five letters are followed by the first three letters of every suffix required for that style of the font. In the font name, capital letters indicate the base name and suffix positions. For example, the downloadable filename for the font Helvetica-Heavy Italic would be *HelveHealta*. This method of using particular characters in a filename is used to limit filename lengths.

**Warning**

---

Because of the restrictions of the hierarchical file system, the font name must never generate a filename that is longer than 31 characters.

---

---

## Permanent downloading of fonts

As discussed earlier, your application may download fonts permanently (that is, until the LaserWriter is turned off) either before or after the Laser Prep dictionary is loaded. Both methods are described next.

### Fonts loaded before loading the dictionary

You can download fonts on a permanent basis by using utilities provided by Adobe, such as `PSDump`, if the font definition is preceded by the string

```
password serverdict begin exitserver
```

However, for the font to be used by the LaserWriter driver, there must be a 'FONT' resource and a 'FOND' resource that points to that font on the printer. When the driver encounters the font in a document, the entry in the driver cache is noted and the 'FOND' resource provides proper font classification. The LaserWriter driver may still use a printer font and Macintosh bit-mapped font pair without a corresponding 'FOND' resource, but the Macintosh font name in the font menu must agree with the printer font name (with spaces in place of hyphens), and the downloaded font must be coordinated. Any style for such a font is derived as though it were a bit-mapped font.

If you use `PSDump` to download fonts permanently, the fonts must be in PostScript text form, as opposed to the resource-file form of temporary downloadable fonts. It is possible for the user to download the resource form of the font using the Apple LaserWriter font utility, in place of `PSDump`.

### Fonts loaded after loading the dictionary

Unlike drivers before version 3.0, later drivers download a font after loading the Laser Prep dictionary in exactly the same way as they download a font before loading the dictionary. No changes to the font are required.

---

## Downloadable file format

A permanent downloadable file must be in PostScript text if you use PSDump or an equivalent utility for downloading. The name and font definition must conform to the specifications described earlier. This requirement does not preclude the development of other methods for downloading. However, for compatibility with the LaserWriter driver, the naming conventions and 'FOND' structure must be followed.

A temporary downloadable font file is an unbundled resource of type 'LWFN' whose creator is 'LWRT'. The file is made up of several resources of type 'POST'. The IDs for these resources begin with the number 501 (decimal) and increase by 1 for each resource in the file. The number of resources is unlimited. Each resource begins with a 2-byte data field that contains the data type in the first byte and a binary zero in the second. The rest of the resource is the data to be downloaded.

The possible values for the first byte of the 2-byte data field are 0, 1, 2, 3, 4, and 5; their meanings are shown in Table 2-4.

**Table 2-4**  
Data types of downloadable font resources

Value	Meaning
0	Ignore the rest of the resource (thus, a comment).
1	The data is ASCII text.
2	The data is binary and is first converted to ASCII hex characters before being sent.
3	An AppleTalk end-of-file (EOF) will be sent (but the AppleTalk connection will be maintained). The rest of the resource data for this value is interpreted as ASCII text and will be sent after the AppleTalk EOF. If there is no additional ASCII text to be sent, the rest of the resource can be empty.
4	The data fork of the current resource file is to be opened and sent as ASCII text. This should be done only once.
5	The end of the resource file; the data in the resource itself may be empty or ASCII text.

The second byte of all the 2-byte fields is reserved and should always be set to 0.

Because resource data is always loaded into memory in its entirety, you should keep the resources to a size of less than 2K. It is not necessary for the data in a resource to begin and to end on any particular boundary, but you may not mix text and binary data in the same resource. For Rez and RMaker users, it may be helpful to set type 'POST' equal to type 'GNRL' when you construct these resources.

When the font data is downloaded, the driver does not interpret any of the data and does not include any prefix or suffix data. However, the driver does coordinate noncoordinated fonts following the downloading, if the font class so specifies. The data in the file is presumed complete and self-contained.

The resource should never contain the string

```
password serverdict begin exitserver
```

Each resource is downloaded in sequence, beginning with resource ID 501, and proceeds in numerical order either until there are no more resources available in the file or until a data type 5 is encountered.

If the font class specifies that the font be coordinated after downloading, the driver will redefine the font on the printer using the name given in the 'FOND' resource for that style, reencode it if required, and give it a new name prefixed with the string

```
| _____
```

as described in "Font-Naming Conventions for Later Drivers," given earlier. Thus, the name in the 'FOND' resource and the normal noncoordinated name for the font must agree.

The downloading process must define the noncoordinated font on the printer with the same name that the 'FOND' resource specifies. The driver subsequently assigns the coordinated name to the font. Adobe built-in and downloadable fonts are examples of noncoordinated fonts that must be coordinated before the driver can access them.

Downloadable fonts have a great potential for creating disaster. They are essentially in control of the entire printed document. While the LaserWriter driver does try to shield itself from misuse, it cannot cover every possible problem. For example, the driver cannot determine whether or not an error has occurred as a result of the downloading process and simply assumes that all is well. However, if an error does occur, the driver reports it in the status window on the Macintosh screen. Since the driver preserves any additional side effects that the font definition causes over the rest of the document, your application must treat such fonts with extreme caution.

Unfortunately, it is not easy to achieve dynamic unloading of fonts (removing a specific font at will) in PostScript except on a *LIFO* basis—which means that the last in is the first out or the last loaded is the first unloaded. Because of this limitation, the LaserWriter driver does not currently support dynamic unloading. Fonts downloaded permanently can be unloaded only by turning off the printer.

---

---

## Downloading fonts from non-Macintosh hosts

Applications running on non-Macintosh hosts can download fonts to the LaserWriter by issuing the correct PostScript code. In this environment, responsibility for managing fonts on the LaserWriter lies solely with the application.

---

### Downloading fonts permanently

You can download fonts permanently by sending the PostScript font file as a separate job. You should send the following line of PostScript code immediately prior to sending the font file:

```
serverdict begin 0 exitserver
```

The 0 in this line is the default password for the printer. Your application should be prepared to handle printers with other passwords.

You should send the LaserWriter a Command-D character sequence immediately after the font file as an end-of-job indicator. The font then remains in the printer until it is turned off.

---

## Downloading fonts temporarily

You can download a font temporarily by simply sending it to the printer with the document file. The font code must appear in the document file before the font is required.

---

## Accessing downloaded fonts

After a font has been downloaded, your application can access the font through PostScript just as with any other LaserWriter font. Your application should be prepared to recognize a variety of fonts and to ensure that they are available to the user as required.

---

## Checking for downloaded fonts

If your application does not maintain a current font list, it should check before sending each font to see if the font is already available in the printer. The following PostScript code initiates a response of `yes` from the LaserWriter if the font is in the printer, or a response of `no` if it is not. (Replace the variable *fontname* with the name of the font you are checking for.)

```
/scratch 100 string def
FontDirectory /fontname
  {(yes)}
  {(no)
   systemdict /filenameforall known
   {(fonts/fontname) {pop pop (yes)} scratch
    filenameforall} if
  } ifelse
print flush
```

%create a scratch string  
%search FontDirectory for *fontname*  
%if it's there put (yes) on the stack  
%else put (no) on the stack and  
%check for a file system at printer  
%if a file system is present, search  
%for *fontname*; if found, replace  
%top of stack with (yes)  
%pop stack to buffer and flush to host

You can get a list of all fonts in the printer with the following PostScript code:

```
/scratch 100 string def
FontDirectory {pop == } forall flush
systemdict /filenameforall known
  {(fonts/*) {dup length 5 sub 5 exch
   getinterval =} scratch filenameforall
  flush} if
```

%create a scratch string  
%send names in FontDirectory to host  
%check for file system at printer  
  
%if file system present, send  
%names of fonts present to host

This code transmits the list of fonts back to the host as ASCII text, with one font name per line.

---

## Determining available room for fonts

A limited number of fonts can reside in the LaserWriter at a time. Before downloading a font, your application should check that there is enough room in RAM for the font. The following PostScript code returns the number of bytes available in the LaserWriter RAM:

```
vmstatus exch sub = flush pop
```

If the number returned is greater than the number of bytes in the font file, there should be enough room to download the font.

---

## Making room for more fonts

Currently, there is only one way to recover memory in the printer. This method uses the PostScript command `save` to store the state of the virtual memory before your application downloads a font and the command `restore` to return the virtual memory to its previous state after the printer uses the font. You may want to use this method with each font, especially if the application allows many fonts to be in use within a single document.

---

### Warning

Use the PostScript commands `save` and `restore` with caution: `restore` can erase some definitions.

---

---

---

## LaserWriter IINTX font cache and disk file system

LaserWriter IINTX PostScript firmware provides support for a simple disk-based file system for font cache and other user utilities. This section describes the file-system features and provides a synopsis of the PostScript operators necessary to implement the disk file system. The material in this section is for advanced programmers.

Apple has developed an application called the *LaserWriter Font Utility* that implements the PostScript operators described in this section. Therefore, the information here applies primarily to two types of advanced programmers who are already familiar with the PostScript language: those who are developing non-Apple CPU

applications, or those who wish to include these utilities in their Macintosh applications or possibly to develop new Macintosh Desk Accessories.

The LaserWriter Font Utility provides most of the necessary utilities for Macintosh users of LaserWriter IINTX printers equipped with or without a hard disk. The application performs four basic operations:

- initializing the hard disk file system
- downloading fonts to the LaserWriter IINTX disk or RAM
- showing available fonts on a disk attached to the LaserWriter IINTX, and in LaserWriter IINTX RAM and ROM
- printing a list of available fonts

---

## Disk and file-system overview

The LaserWriter IINTX PostScript file system uses one or more devices for data storage. In the LaserWriter IINTX, there are two potential types of devices: the disk device and the cartridge device.

Each instance of a device has a unique name; this name can be part of the parameter *filename* to operators that deal with files. For example,

```
(%cartridge1%name) (r) file
```

will return a filestream to a file named *name* on the device whose name is *%cartridge1%*.

In addition, *file* will accept a filename without a device name and will search all devices for a file of that name.

```
(logo1) (r) file
```

will return a filestream to a file named *logo1* if such a file exists on any valid device. If more than one file with that name exists, *file* will return a filestream to one of them, but it is not defined how to determine which.

Additionally, a device may consist of a single unnamed stream of data, which can be referenced only by the device name. For example:

```
(%cartridge1%) (r) file
```

will return a filestream to an unstructured cartridge device whose name is *%cartridge1%*.

You can configure the LaserWriter IINTX controller with one or more fixed media disks, connected via LaserWriter IINTX's external SCSI interface. The disk or disks, if present, form one logical disk device for the PostScript file system. The system does not distinguish between individual disk drives: for example, you cannot reformat a single drive of a multiple-drive disk device; reformatting the disk device requires reformatting all drives that make up the disk device. The name of the disk device, if present, is *%disk%*.

You can also configure the LaserWriter IINTX controller with an optional font expansion board that provides an area of read-only memory referred to in PostScript terms as a *cartridge device*. The name of the cartridge device, if present, is *%cartridge1%*. For hardware-specific information about the optional font expansion board, see Chapter 6.

Future versions of the LaserWriter IINTX may have new types of devices, and may allow multiple devices of a single type, such as several cartridge devices that share the additional ROM.

To support fixed disks, LaserWriter IINTX's PostScript implementation has been extended to include a simple file system that provides

- nonvolatile storage for a cache of character bit maps
- a virtual memory for expanding display-list buffers for complex pages
- a user-accessible file-storage facility

The file system provides a storage facility that is less volatile than RAM, but not as stable as a backed-up file system. Therefore, you should ensure that all data on the disk is recoverable from another source of storage.

---

## The Sys/Start file

When you turn on the LaserWriter IINTX controller, PostScript checks the disk device for file-system integrity (possibly updating the bit table for allocated pages) and the disk-based font cache for internal consistency. If the check of the file-system integrity fails, the disk device is reinitialized. If the check of the internal consistency of the disk-based font cache fails, the information in the font cache is discarded, and the corresponding data structures and files are restored to their initial state.

After successful completion of the start-up diagnostics, PostScript then searches the disk for a file named *Sys/Start*. If this file is present, and if the value of `dosysstart` is true, the file is executed just as if that file were the first job sent to the controller. (For information about `dosysstart`, see "System Parameters" in Chapter 4.) The *Sys/Start* file can be used to set volatile parameters and to load special items into virtual memory (VM) before any other jobs are executed. If the *Sys/Start* file is not present, the server loop merely waits for the first job to arrive via the selected communication channel.

If a bad *Sys/Start* file is present on the disk-file system, you can prevent the file from being executed by turning on the LaserWriter IINTX with the disk that contains the bad *Sys/Start* file turned off, and then setting the value of `dosysstart` to false. When the value of `dosysstart` is false, the *Sys/Start* file is not executed the next time the LaserWriter IINTX is turned on. Then, the LaserWriter IINTX can be turned on with the disk that contains the *Sys/Start* file turned on, and the file can be repaired or deleted if desired.

If the *Sys/Start* file tries to send output back through the communication channel (for example, by using the `print` operator), the effects vary according to the current communications mode. If AppleTalk is selected, all communication output is lost, because no other host has yet made a connection. If serial communication is selected, the 25-pin output channel is used according to its setup via `setscbatch` unless it has been disabled. If the 25-pin channel has been disabled by the 0 baud parameter to the `setscbatch` operator, then the 8-pin channel is used instead.

### **Working with the *Sys/Start* file**

To protect the system, you cannot read, write, delete, or rename any file named *Sys*. These restrictions mean that any jobs accessing *Sys* files must execute an `exitserver` first. To access the *Sys/Start* file, you must store it on the disk from outside of the server loop.

Because the *Sys/Start* start-up file executes from within the server loop as if it were the first incoming job, its behavior must be the same as that of a normal job. If exiting the server loop is required for some operations performed by a job, then the same sequence of PostScript operations is required when the job runs from the disk as the start-up job. Doing this allows for testing the file by

executing the file as a batch job through a communication channel in the normal manner. Once the job works satisfactorily, you may then store it on the LaserWriter IINTX disk unchanged for its role as the start-up job.

Store the start-up job under the name *Sys/Start* outside the server loop. However, it is normally undesirable to do a lot of work outside of the server loop because you can not reclaim any VM consumed outside the server loop until the next power-on. For this reason, storing the start-up file in a two-step process is recommended. The first step stores the start-up job into a temporary file without exiting the server loop. This step can consume VM to perform the file storage efficiently; this VM is reclaimed when the job is complete. The second step exits the server loop and renames that temporary file *Sys/Start*.

As the first step, to store the start-up job inside the server loop, concatenate the following preamble of PostScript code onto the beginning of the file to be stored:

```
/StartTempFile (StartTemp) (w) file def
/buffer 1024 string def
/storefile
(
  {currentfile buffer readstring
   pop dup length 0 eq
   {pop StartTempFile closefile exit}
   {StartTempFile exch writestring}
   ifelse
  } loop
) def
storefile
<<insert text of the start-up job here>>
```

This PostScript code reads all of the lines following the line *storefile*, writes them into a file named *StartTemp*, and terminates when the input file ends.

The second step exits the server loop and renames the file with the following PostScript job:

```
serverdict begin 0 exitserver
statusdict begin
(Sys/Start) status {clear (Sys/Start) deletefile} if
(StartTemp) (Sys/Start) renamefile
end
```

This job exits the server loop (the 0 in the first line represents the password), deletes the `Sys/Start` file if it exists, and renames `StartTemp` to `Sys/Start`. Note that these two PostScript jobs do not actually execute the start-up job themselves; they merely store it on disk.

---

## Finding fonts on the disk file system

The PostScript operator `findfont` has been extended for LaserWriter IINTX to make use of the disk file system for font storage. In PostScript printers without disks, the `findfont` operator searches `FontDirectory` for fonts that are available in ROM or that have been downloaded into RAM. (For a detailed description of the `findfont` operator, see the *PostScript Language Reference Manual*.)

In a LaserWriter IINTX configured with a disk, users may download fonts as files both into the disk-based file system and into RAM. Therefore, `findfont` operates differently. For example, when the PostScript interpreter encounters the input

```
/Palatino findfont
```

it first searches `FontDirectory` for the name *Palatino*. If that search fails, it then searches the disk file system for the file named *fonts/Palatino*. If that search succeeds, PostScript executes the file, thereby loading the font into virtual memory and registering the name *Palatino* in `FontDirectory`. As with the fonts downloaded directly into RAM over a communication channel, you must use the `save/restore` mechanism carefully to ensure VM does not become full.

---

## LaserWriter IINTX disk and file-system operators

This section provides detailed overview of the file system, and contains the definitions of disk and file-system operators that are extensions of the standard set of PostScript operators.

As indicated in the overview of the disk file system at the beginning of this section, this file system does not attempt to provide the mechanisms typically found in the file system of a multiuser operating system, such as file protection, hierarchical directory structure, and so on. Hence, you must follow this set of conventions when accessing the file system:

- Filenames are case-sensitive, consist of at most 100 characters, and do not begin with %. A hierarchical naming convention is followed by PostScript and is strongly suggested for users.
- All system filenames are of the form *dir/name*.
- Do not choose names that begin with these prefixes: *L Sys/*, *FC/*, and *DB/* (except for the special start-up file named *Sys/Start\_*). PostScript uses names with these prefixes for the disk directory, font cache, and display-list buffering, respectively.

There are two types of operations on the file system:

- PostScript file operators that are already part of the PostScript language (*read*, *write*, *file*, *run*, and so on)
- extensions to the standard file operators that are generally only available for PostScript products that include disk-based file systems

These operators are all defined in `systemdict`, with the exception of those in the following list, which are defined in `statusdict`:

```
diskstatus
initializedisk
setuserdiskpercent
userdiskpercent
```

Brief descriptions of the LaserWriter IINTX disk-based file-system and font expansion card operators are provided next.

## **cartstatus**

### **Syntax**

*string* cartstatus *false*  
*string* cartstatus *type id filetype true*

### **Definition**

The operator `cartstatus` takes a device name string or device-and-filename string, *string*, and returns status information for that device or file. A result of *false* indicates that no font expansion card is plugged in, or that no such device exists. Here, *type* is an integer that indicates the purpose of the font expansion card, with the font expansion card being of type 4; *id* is a unique font expansion card identifier; *filetype* is an integer that indicates the purpose of the file, with font files being type 4. If *string* designates a device but not a file, or if no file by the specific name is present, *filetype* is 0.

### **Errors**

undefinedfilename, limitcheck, typecheck

## **deletefile**

### **Syntax**

*namestring* deletefile

### **Definition**

The operator `deletefile` deletes the file, specified by *namestring*, from the disk file system.

(This operator is defined in `systemdict`, and not in `statusdict`.)

### **Errors**

ioerror, stackunderflow, typecheck, undefinedfilename.

## **devdismount**

### **Syntax**

*device* devdismount

### **Definition**

The operator `devdismount` takes a device name string, *device*, from the stack and "dismounts" that device. Before beginning each PostScript job, the PostScript server loop always mounts the font expansion card and any disks, if present. Therefore, it is generally unnecessary for a PostScript program to mount or dismount a font expansion card or disk.

(This operator is defined in `systemdict`, and not `statusdict`.)

### **Errors**

undefinedfilename (invalid device name), limitcheck, typecheck

## devmount

### Syntax

*device* devmount *bool*

### Definition

The operator `devmount` takes a device name string, *device*, from the stack and "mounts" that device. It is necessary for a device to be mounted before it can be used. Before beginning each PostScript job, the PostScript server loop always mounts the font expansion card or any disks, if present. Therefore, it is generally unnecessary for a PostScript program to mount or dismount a font expansion card or disk. The operator returns a boolean, *bool*. True indicates that the device was already mounted or has been mounted successfully. False indicates that there was no module inserted.

(This operator is defined in `systemdict`, and not `statusdict`.)

### Errors

undefinedfilename (invalid device name), limitcheck, typecheck

## devstatus

### Syntax

*device* devstatus *free size*

### Definition

The operator `devstatus` takes a device name string, *device*, from the stack and pushes the integers *free* and *size*. For a font expansion card, both integers are 0.

(This operator is defined in `systemdict`, and not `statusdict`.)

### Errors

undefinedfilename (invalid device name), limitcheck, typecheck

## diskonline

### Syntax

diskonline *bool*

### Definition

The operator `diskonline` returns a boolean, *bool*. True indicates that the device is attached and online. False indicates that the disk is not attached or online.

(This operator is defined in `statusdict`, and not in `systemdict`.)

### Errors

undefinedfilename (invalid device name), limitcheck, typecheck

## diskstatus

<b>Syntax</b>	<code>diskstatus <i>free total</i></code>
<b>Definition</b>	The operator <code>diskstatus</code> returns the number of pages, <i>free</i> , that are currently free in the file system and the maximum number of pages available, <i>total</i> . (A page is 1024 characters.)
<b>Errors</b>	<code>stackoverflow</code>

## initializedisk

<b>Syntax</b>	<code>pages <i>action</i> initializedisk</code>
<b>Definition</b>	The operator <code>initializedisk</code> formats the disk or disks making up the file system and initializes the file system. If <i>pages</i> is 0, then the entire disk system is selected. A value of <i>pages</i> greater than 0 specifies the number of pages affected by the operator (a page is 1024 characters). If <i>action</i> is 0, the file system is initialized. If <i>action</i> is 1, first the disk or disks are formatted and then the file system is initialized. If the value of <i>action</i> is <i>n</i> , with <i>n</i> greater than 1, then the disk or disks are formatted; the disk surfaces are tested <i>n</i> -1 times (to search for bad pages); and the file system is initialized.
<b>Errors</b>	<code>ioerror, rangecheck, stackunderflow, typecheck</code>

## filenameforall

<b>Syntax</b>	<code>pattern <i>proc scratchstring</i> filenameforall</code>
<b>Definition</b>	The operator <code>filenameforall</code> sequences through the file directory and, for each file whose name matches the pattern string, <i>pattern</i> , it executes the procedure, <i>proc</i> , passing it <i>scratchstring</i> with the filename as its contents. The string, <i>pattern</i> , may contain two forms of wildcard characters: * matches any substring and ? matches any single character. The string <i>scratchstring</i> must be large enough to contain the largest allowable filename (currently 100 characters). The order of enumeration is not specified and files that <i>proc</i> creates or deletes may or may not be encountered in the enumeration. The returned filenames will not include the device name, unless one or more leading characters of the device name are present in the pattern. For example, a pattern of (*) will return filenames without the device names <i>file1</i> , <i>file2</i> , and <i>file3</i> , but a pattern of (%*) will return filenames with device names <i>%cartridge1%file2</i> , <i>%cartridge1%file3</i> , <i>%disk%file1</i> , and <i>%disk%file4</i> .  (This operator is defined in <code>systemdict</code> , and not in <code>statusdict</code> .)
<b>Errors</b>	<code>stackoverflow, stackunderflow, typecheck</code>

## renamefile

- Syntax** `oldnamestring newnamestring renamefile`
- Definition** The operator `renamefile` changes the name of the file, *oldnamestring*, to a new name, *newnamestring*.  
(This operator is defined in `systemdict`, and not in `statusdict`.)
- Errors** `ioerror`, `stackunderflow`, `typecheck`, `undefinedfilename`

## setuserdiskpercent

- Syntax** `percent setuserdiskpercent`
- Definition** The operator `setuserdiskpercent` takes the percentage, *percent*, of the disk portion of the file system's storage to be reserved for users' files; *percent* is an integer. The remainder of disk storage is used for font caches and file-system overhead. If the requested space is not available, `setuserdiskpercent` will delete the least-recently used cached bit maps from the disk file system until sufficient free space is available.
- Errors** `rangecheck`, `stackunderflow`, `typecheck`

## status

- Syntax** `stringname status pages bytes access creation true`
- Definition** The operator `status` returns the number of pages, *pages*, allocated to the file specified by the *stringname*; the number of characters, *bytes*, actually written into the file; and the times of most recent access and creation of a given file in *access*, and *creation*. If the file is found, the operator returns *true*; if the file is not found, it returns *false*.  
The notion of time, as implemented in this file system, is merely a monotonically increasing integer value with no calendar-related interpretation. The primary use for time in the system is to implement a least-recently used strategy for recovering space in the disk-based font cache.
- ❖ *Note:* There is a `status` operator in `systemdict` in the standard PostScript language. The behavior just described is an extension to the standard semantics of `status`, invoked when the operand is a string rather than a file object.
- Errors** `stackoverflow`, `stackunderflow`, `typecheck`

## **userdiskpercent**

**Syntax** userdiskpercent *percent*

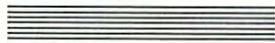
**Definition** The operator `userdiskpercent` returns the percentage, *percent*, of disk space reserved for user files.

**Standard value** 1

**Errors** `stackoverflow`



## Chapter 3



# Working in the Printing Environment

This chapter discusses the LaserWriter printing environment, which includes the LaserWriter modes of operation, serial-communication parameters and configuration, direct communication, and printer-emulation features.

---

---

## LaserWriter operation modes

The LaserWriter printers are connected to the outside world by two serial ports. The original LaserWriter and LaserWriter Plus use a 9-pin AppleTalk RS-422 connector and a DB25 RS-232C connector. The LaserWriter IINT and LaserWriter IINTX use an 8-pin miniature DIN AppleTalk RS-422 connector and a DB25 RS-232C connector. Through these ports, or communication channels, the LaserWriter communicates bidirectionally with a computer or terminal. The LaserWriter can send and receive data. Communication over these channels is **asynchronous**, except for AppleTalk which is **synchronous**. In asynchronous mode characters can be sent and received simultaneously. In synchronous mode transmitted and received packets of data are interleaved.

How the LaserWriter processes the information it receives over the channel is determined by its current operating mode:

- **Batch mode** is the LaserWriter's normal operating mode. In batch mode, a job consists of executing a single file that contains a PostScript program. When the end-of-file indicator is received, the job is finished. The only data that the LaserWriter transmits to the host is generated specifically by a PostScript operator, such as `print` or `prompt`, or by an error. An application that prints on the LaserWriter uses batch mode.
- **Interactive mode** allows the user to communicate directly to the LaserWriter either from a Macintosh running MacTerminal or from another terminal. The LaserWriter assumes the role of a powerful computer, which it really is, while still serving as a printer. Interactive mode is useful both for experimenting with PostScript and for using the LaserWriter as a general-purpose computer. If you need to test parts of your application, this mode can be useful.

- **Emulation mode** is a special mode in which the LaserWriter runs built-in code to emulate the Diablo 630 daisy wheel printer, which is widely supported by personal computer applications. If your application is designed to print on a printer compatible with the Diablo 630 all-purpose interface, selecting this mode enables the LaserWriter to interpret the control codes in the file to be printed. Emulation mode on the LaserWriter IINTX provides access to built-in code for either a Diablo 630 or Hewlett-Packard LaserJet<sup>+</sup> printer emulators. Later, this chapter provides a table of the LaserWriter IINTX printer emulator switch settings.

---

### Printer switch settings

The LaserWriter and LaserWriter Plus have a rotary four-position mode switch on the back panel. Together with some PostScript operators, this switch controls the mode of operation and the communication protocol. Table 3-1 lists the LaserWriter and LaserWriter Plus switch positions. The LaserWriter IINT has a two-position DIP switch on the left-side panel. Table 3-2 lists the LaserWriter IINT switch settings. The LaserWriter IINTX has a six-position DIP switch on the left-side panel. Table 3-3 lists the LaserWriter IINTX switch settings and mode definitions.

Changing the switch setting changes the mode immediately. If a print job is in progress, it is terminated.

**Table 3-1**  
LaserWriter operational-mode switch settings

Switch setting	Mode
AppleTalk	Batch mode. This is the normal mode for an application communicating to the LaserWriter over AppleTalk.
Special	Emulation mode. This mode provides serial communication using previously set parameters. (The default parameters are 9600 baud, with parity ignored.)
9600	PostScript batch mode. This mode provides serial (RS-232 or RS-422) communication using the current communication parameters. (The default parameters are 9600 baud, with parity ignored.) Because these parameters can be reset under software control, setting the switch to the 9600 position may select a data-transmission rate other than 9600.
1200	PostScript batch mode. This setting provides serial communication through either of the connectors, at 1200 baud, with parity ignored.

**Table 3-2**  
LaserWriter II<sup>NT</sup> operational-mode switch settings

Switch setting		Definition
<b>Switch 1</b>	<b>Switch 2</b>	<b>Serial-port configuration settings</b>
UP	UP	AppleTalk enabled (DB25 RS-232C port is disabled)
UP	DN	PostScript batch mode; serial RS-232C and RS-422 ports 9600 baud, with parity ignored
DN	UP	Emulation mode; RS-232 serial port set at 9600 baud
DN	DN	PostScript batch mode; serial RS-232C and RS-422 ports 1200 baud, with parity ignored

**Table 3-3**  
LaserWriter II<sup>NTX</sup> operational-mode switch settings

Switch setting		Definition
<b>Switch 1</b>	<b>Switch 2</b>	<b>Serial-port configuration settings</b>
UP	UP	AppleTalk enabled (RS-232C port is disabled)
UP	DN	Serial RS-232C and RS-422 ports 9600 baud
DN	UP	Serial RS-232C and RS-422 ports 1200 baud
DN	DN	Serial RS-232C 9600 baud, RS-422 0 baud
<b>Switch 3*</b>	<b>Switch 4*</b>	<b>Mode settings</b>
UP	UP	PostScript batch mode
UP	DN	PostScript interactive mode
DN	UP	Emulation mode (RS-232 serialport set at 9600 baud)
DN	DN	Hewlett-Packard LaserJet <sup>+</sup> emulation mode (RS-232 serial port set at 9600 baud)
<b>Switch 5*</b>	<b>Switch 6*</b>	<b>Handshake Protocol</b>
UP	UP	Xon/Xoff
UP	DN	Data Terminal Ready (DTR)
DN	UP	Etx/Ack
DN	DN	Xon/Xoff

\* If AppleTalk is selected, switches 3 through 6 have no effect.

The serial-port configuration information is stored in a 2K battery-backed-up Zero Power-RAM (ZPRAM). The ZPRAM maintains the switch-configuration settings between power-off and power-on cycles. Serial-port configuration settings can also be selected through software control with the PostScript control operators described in "Changing Communication Parameters" in Appendix A.

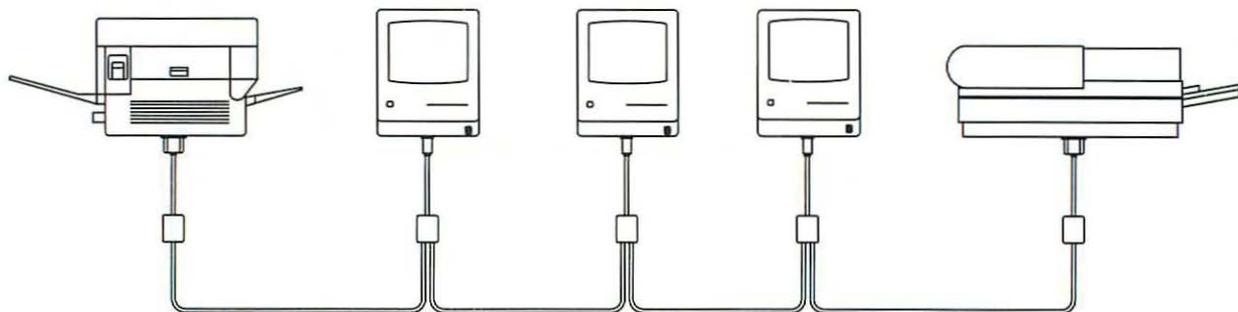
If you believe that the LaserWriter IINTX controller is incorrectly configured, toggle any switch, wait 30 seconds, and then toggle it back to its normal position. Doing this forces the ZPRAM configuration information to match the switches and overrides any conflicting software downloaded configuration.

---

---

## Using AppleTalk

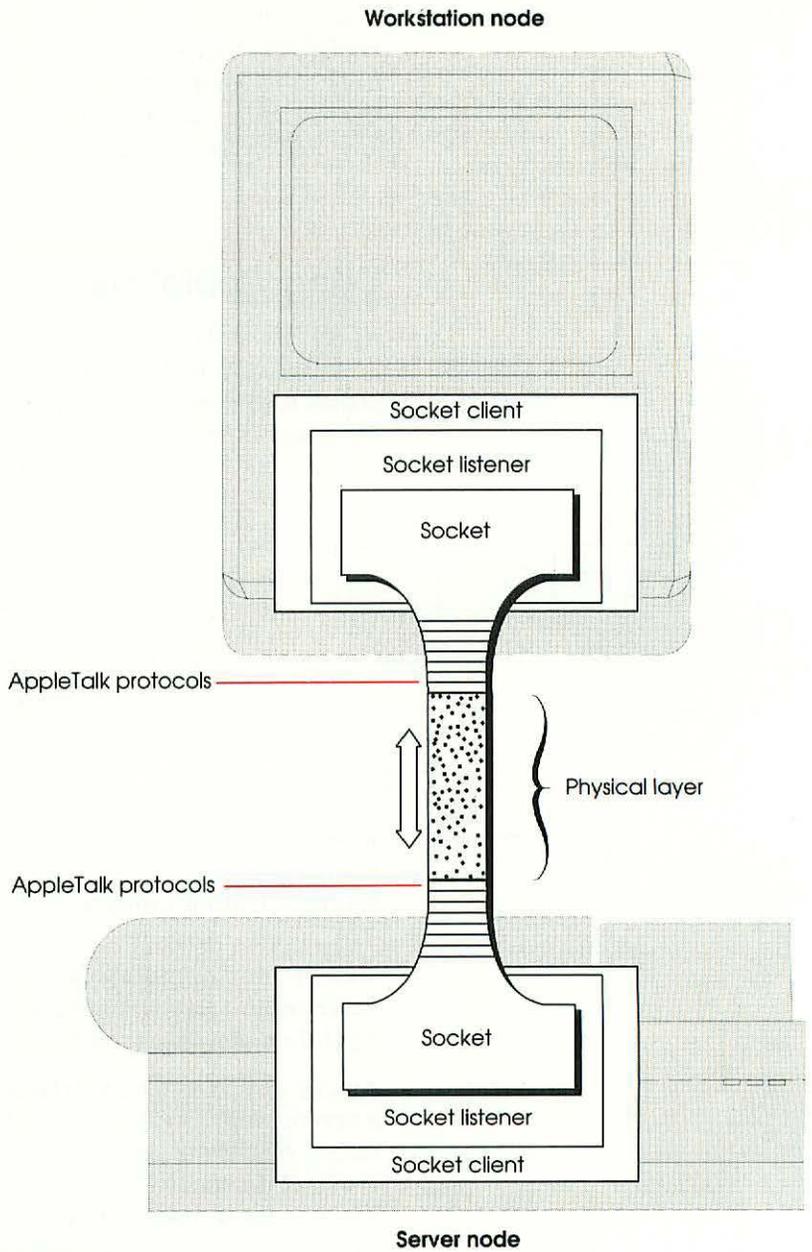
AppleTalk is the network architecture that governs communication between LaserWriter printers and Macintosh computers. Figure 3-1 shows such a network. This simple network consists of Macintosh computers and LaserWriter printers, each of which are joined to the network by a connection box.



**Figure 3-1**  
Simple AppleTalk network

Two or more AppleTalk networks can be connected to form an **internet**. A maximum of 32 devices can be connected to a single AppleTalk network. Each device is known as a **node**.

A node can be either a **workstation node** (such as a Macintosh) or a **server node** (such as a LaserWriter). Figure 3-2 shows these two nodes. A software process within a node can open **sockets**, which are logical connections to the network. The process that opens a socket is the socket's **client**; a client is said to *own* its sockets. The socket client must provide a **socket listener**, which is code to receive information from the socket.



**Figure 3-2**  
Network sockets

---

---

## Accessing the LaserWriter directly

You can connect a Macintosh that runs MacTerminal directly to the port on the back of the LaserWriter. You can also connect an ASCII terminal or another computer running terminal-emulation software. By connecting either of these, you can access the LaserWriter without going through several levels of network software. You can program in either interactive mode or batch mode. In interactive mode, you can hold a dialog with the LaserWriter, instructing it to execute PostScript code that you send to it. Using interactive mode can be a convenient way to test pieces of your PostScript programs. In batch mode, you send an entire file for the LaserWriter to execute; by doing this, you can test an entire PostScript program you have written or modified.

You connect a Macintosh (or an Apple II that runs Access II) to the LaserWriter's 9-pin connector, or the LaserWriter IINT or LaserWriter IINTX's 8-pin MINI DIN connector by using an Apple Personal Modem cable or an ImageWriter cable. These cables must have the 8-pin or 9-pin connector to match the LaserWriter, LaserWriter IINT, or LaserWriter IINTX 8-pin or 9-pin connector. You can also connect any terminal (or a computer that runs terminal-emulation software) with a standard RS-232C asynchronous interface directly to the LaserWriter, usually through the 25-pin connector on the back of the printer. When making this connection, you generally need to use a null-modem device or a modem-eliminator device that transposes the Transmit Data and Receive Data signals. See Appendix A for serial-interface signals.

---

### Warning

Macintosh applications using the Printing Manager communicate with the LaserWriter through AppleTalk cables and connectors. Do not use the AppleTalk connector for any other communication link to the printer. These connectors hook up to the DB-9 or MINI DIN-8 connector on the LaserWriter, LaserWriter IINT, or LaserWriter IINTX, respectively, and to the printer port on the Macintosh; you should use these connectors only for the AppleTalk network. Do not hook up AppleTalk and the RS-232C interface at the same time on the LaserWriter. If you use the RS-232C port and wish to leave AppleTalk connected on the LaserWriter IINT or the LaserWriter IINTX, you must set the SCC interactive baud parameter to 0 for the RS422 port. On the LaserWriter IINTX, you can also set the baud to 0 with mode switches one and two.

---

---

## Working interactively

There are two ways to put the LaserWriter into interactive mode. One way is first to select one of the batch-mode switch positions (1200 or 9600), then to make sure the terminal is set to the correct baud and parity, and to invoke the PostScript procedure `executive`, which you do by typing `executive` and then a Return character or a line feed. The other way is to redefine the meaning of the Special (printer-emulation) switch position by using the PostScript `seteescratch` operator, so that selecting the Special setting invokes interactive mode instead of emulation mode.

In interactive mode, the state of PostScript's VM persists until you explicitly end the job. While you type, the LaserWriter accepts the characters typed and echoes them back to your terminal so you can see them. You can use four special characters for editing while you type. Table 3-4 describes these characters.

**Table 3-4**  
Interactive mode line-editing characters

Characters	Effect
Command-H	Erases one character (backspace)
Command-U	Erases the current line
Command-R	Displays the current line again
Command-C	Cancels the entire statement and starts over

Interactive mode continues until you press Command-D (the serial end-of-file character), execute a PostScript `quit` command, or change the mode switch setting.

- ❖ *Note:* On a Macintosh, the **Command key** is the equivalent of the Control key on other computers. The Command key is the one immediately to the left of the space bar.

The following exercise gives step-by-step instructions for running an interactive session with any of the LaserWriter PostScript printers. The instructions assume that you are using a Macintosh computer that runs MacTerminal as an ASCII terminal interface, but the steps are similar for using any ASCII terminal or computer that runs terminal-emulation software. If you are using an MS-DOS computer and a LaserWriter IINT or LaserWriter IINTX, see the *LaserWriter IINT/IINTX Owner's Guide* for a tutorial example of an interactive session.

**1. Set the printer to a known state.**

Set the power switch to the Off position. Disconnect any communication cables attached to any of the connector ports. Set the operational-mode switch on the back of the LaserWriter to 1200 baud. (For operational-mode switch settings, see Tables 3-1, 3-2, and 3-3 in the first section of this chapter.)

**2. Connect the LaserWriter and the Macintosh.**

To connect a Macintosh 128K or a Macintosh 512K, use a cable with DB-9 plugs on both ends, or a cable with a DB-9 plug on one end and a DB-25 plug on the other (a standard ImageWriter cable will work). Connect the 9-pin end of the cable to the modem port on the back of the Macintosh. Connect the other end of the cable to the appropriate port (9-pin or 25-pin) on the LaserWriter.

To connect a Macintosh Plus, Macintosh SE, or Macintosh II, use an ImageWriter cable and a Macintosh Plus adapter cable. Connect the DB-25 connector to the 25-pin port on the LaserWriter. Connect the two DB-9 connectors together. Connect the MINI DIN-8 connector to the modem port on the Macintosh. You can also use a Macintosh modem cable to connect the Macintosh modem port to the MINI DIN-8 port on the LaserWriter IINT or the LaserWriter IINTX.

### 3. Initialize the printer.

Turn the power switch to the On position. Several things happen during initialization. First, the green light blinks. This light is the printer's normal warm-up indicator. Next, the yellow light blinks, signaling that the printer is processing data. In this case, the printer is processing the startup test page. (On the LaserWriter IINTX, the green light flashes either when the printer is warming up or during the data-processing cycle, and remains on steadily during idle periods). If the printer is operating properly, a test page prints within two minutes.

### 4. Run MacTerminal.

Start up MacTerminal by double-clicking the MacTerminal icon. A session window opens automatically.

### 5. Set the communication parameters.

Use the Settings menu to set the Terminal, Compatibility, and File Transfer parameters as follows:

Terminal	Terminal = VT100
	Mode = ANSI
	Cursor shape = underline
	Character set = U.S.
	Line width = 80 column
	On-Line, Auto Repeat, Auto
	Wraparound
Compatibility	Baud = 1200
	Bits per character = 8
	Parity = none
	Handshake = none
	Connection = another computer
	Connection port = modem
File Transfer	Transfer method = text

#### 6. Request the status of the LaserWriter.

Press Command-T, which causes the printer to display its status on the screen in the following format:

```
%% [status: waiting; source serial 9] %%
```

#### 7. Initialize the job cycle.

If the status is not idle, press Command-D to send an end-of-file indicator to the LaserWriter. The EOF prepares it to receive the new job. Then press Command-T again. This message appears:

```
%%[status: idle]%%
```

#### 8. Enter interactive PostScript mode.

Type the word `executive` and press Return. Because the LaserWriter is still in batch mode until it processes this command, the characters you type aren't echoed back to you. If you make a mistake, press Command-D and try again.

If you don't make any mistakes, information similar to the following appears on the screen:

```
PostScript(tm) version 47.0
```

```
Copyright (c) 1984 Adobe Systems Incorporated.
```

```
PS>
```

(The version number and copyright information varies according to the model LaserWriter in use.)

The PostScript prompt is `PS>`, which indicates that you are now communicating directly with the PostScript interpreter. Each time the LaserWriter displays the `PS>` prompt, it is waiting for you to type a PostScript statement followed by Return or a line feed. The LaserWriter then executes that statement and displays another `PS>` prompt.

#### 9. Test the connection.

Type `showpage` and press Return. If the connection is functioning correctly, the LaserWriter ejects a blank piece of paper.

## 10. Enter a PostScript program.

In response to the PS> prompt, carefully type the following commands and press Return after each one. Type your name in place of *your\_name\_here*.

```
/Times-BoldItalic findfont
72 scalefont setfont
100 100 moveto
30 rotate
(your_name_here) show
showpage
```

The PS> prompt does not appear immediately after execution of the `showpage` operator because the PostScript interpreter is busy creating a scaled and rotated font.

The LaserWriter then ejects a page with your name printed at a 30-degree angle.

## 11. End the job.

Press Command-D to end your print job. The end-of-file indicator causes the LaserWriter to terminate interactive mode.

---

## Working in batch mode

In batch mode, you can send many lines of PostScript to the printer at once as a text file. You might want to use this mode to test a PostScript program that you have created or modified. A PostScript program file is a text file that consists of valid PostScript commands. In batch mode, the PostScript program file is simply sent to the printer and executed. The program statements are not echoed on screen as they are in interactive mode. To experiment with the batch mode, follow these steps:

### 1. Set up the printing environment.

Prepare the LaserWriter as described in the first three steps of the interactive-mode exercise in “Working Interactively,” just given.

## 2. Create a PostScript program.

Using the MacWrite text-processing program or another text editor, create a text-only file by carefully entering the following PostScript commands:

```
2 2 scale
/Times-BoldItalic findfont 27 scalefont setfont
/rays
{0 1.5 179
{gsave
  rotate
0 0 moveto 108 0 lineto
stroke
grestore
} for
} def
125 200 translate
.25 setlinewidth
newpath
0 0 moveto
(StarLines) true
charpath clip
newpath
54 -15 translate
rays
showpage
```

## 3. Save your program file.

Use MacWrite's Save As (or the equivalent save command in your text editor) to write the program text just given to your disk as a text-only file by using the Text Only options with the name *PSTEST*.

## 4. Send your file to the LaserWriter.

Start up MacTerminal. Make sure your settings are correct, as given in the fifth step of the interactive-mode example. Choose Send File from the File menu, and then open the *PSTEST* file that you just created. This procedure should send the entire file to the printer. In about two minutes on the LaserWriter and LaserWriter IINT and about half a minute on LaserWriter IINTX, the page should print.

---

---

## Using the Diablo 630 emulator

In emulation mode, the LaserWriter interprets incoming data as text and Diablo 630 control codes, rather than as a PostScript program. With this capability, the LaserWriter can print simple text files generated by software packages that don't support PostScript, but that do support Diablo-630-compatible printers.

---

## Invoking the Diablo emulator

To invoke the Diablo emulator, set the mode switch on the back of the LaserWriter or LaserWriter Plus to the Special position, and connect one of the LaserWriter's serial ports to the workstation's RS-232C interface. Text to be printed can then be sent at 9600 baud with any parity. (For the emulation-mode switch settings for the LaserWriter IINT and LaserWriter IINTX, see Tables 3-2 and 3-3.)

## Invoking the Diablo emulator with software

As an alternative to setting the mode switches on the LaserWriter IINTX, you can also invoke the Diablo 630 emulator in software by setting the value of the PostScript operator, `softwareiomode`, to three. For more information on using `softwareiomode` and other LaserWriter IINTX PostScript operators, see Chapter 4.

Most of the information about serial communication in Appendix A applies to Diablo 630 emulation. However, control characters have different meanings in emulation mode: all characters are interpreted according to the Diablo 630 protocol, as shown in Table 3-5. The LaserWriter still sends XON and XOFF characters to control the flow of data from the host.

For a more detailed description of the AppleTalk network, see *Inside AppleTalk*.

- ❖ *Note:* Not all printer drivers in microcomputer operating systems support the XON/XOFF protocol. You may need a separate software package to support this protocol.

**Table 3-5**  
Control characters for Diablo 630 emulation

Operating code*	Meaning
SC <b>BS</b> <i>x</i>	Backspace
ESC <b>HT</b> <i>x</i>	Tab to column <i>x</i>
ESC <b>LF</b>	Negative line feed
ESC <b>VT</b> <i>x</i>	Vertical tab to line <i>x</i>
ESC <b>FF</b> <i>x</i>	Set lines per page
ESC <b>CR</b> P	Reset printer
ESC <b>DC1</b> <i>x</i>	Set offset for proportional spacing to <i>x</i>
ESC <b>RS</b> <i>x-1</i>	Set vertical-motion increment
ESC <b>US</b> <i>x-1</i>	Set horizontal-motion increment
ESC &	Set bold and shadow printing off
ESC ,	Set reverse printing mode
ESC -	Set vertical tab
ESC .	Set normal printing mode
ESC 0	Set right margin
ESC 1	Set horizontal tab
ESC 2	Clear all horizontal and vertical tabs
ESC 5	Set forward printing mode
ESC 6	Set backward printing mode
ESC 8	Clear current horizontal tab
ESC 9	Set left margin
ESC =	Center justification
ESC C	Clear top and bottom margins
ESC L	Set bottom margin
ESC S	Use the horizontal motion increment set via hardware switch
ESC T	Set top margin
ESC U	Half-line feed
ESC D	Negative half-line feed
ESC E	Set automatic underscore on
ESC R	Set automatic underscore off
ESC O	Set bold printing on
ESC W	Set shadow printing on (offset overstrike)
ESC X	Clear printing modes (bold, shadow, and so on)
ESC M	Full justification

\* Characters in **bold** indicate ASCII control characters. Characters in *italics* indicate an ASCII character whose decimal value is used to resolve the the command.

---

## Changing print parameters

All of the parameter settings that you can change with Diablo software commands are initialized in the emulator as they are in the Diablo 630 printer. Other parameters require setting hardware switches or changing print wheels in the Diablo printer; in the LaserWriter, these are persistent parameters that you can change via PostScript commands. The persistent parameters that pertain to Diablo 630 emulation are shown in Table 3-6.

**Table 3-6**  
Persistent parameters for the Diablo 630 emulator

Parameter	Initial setting
Pitch	10
Font	Courier
Font for bold	Courier-Bold
Auto line feed	Off

The Diablo 630 emulator supports all standard LaserWriter typefaces. The default font is Courier, which is the fixed-pitch font most commonly used on daisy-wheel printers and the font most likely to give correct results for typical application programs that provide Diablo 630 emulation. You specify plain and bold fonts separately. Thus, you can use Courier for regular printing and Courier-Oblique for bold; then text that you specify to print as bold on the Diablo 630 would print as italic instead.

Persistent parameters that control Diablo 630 emulation and mode selection (after you set the LaserWriter's mode switch) have been assigned by using EEROM locations that are accessed by the PostScript commands `eescratch` and `seteescratch`. For more details on these commands, see the *PostScript Language Reference Manual*.

- ❖ *Note:* In the LaserWriter IINTX, the EEROM has been replaced with ZPRAM (*zero power random-access-memory*). Because ZPRAM in the LaserWriter IINTX provides the same function as the EEROM in the LaserWriter, the PostScript operators `eescratch` and `seteescratch` also control the persistent parameters in the LaserWriter IINTX.

These `eescratch` locations have been assigned as shown in Table 3-7. For example, to change the meaning of the switch position from emulation mode to PostScript interactive mode, type the command

```
58 1 seteescratch
```

The default value of every `eescratch` cell is 0.

**Table 3-7**

The `eescratch` location assignments

Location	Assignment
58	Selects the function of the Special switch setting: 0 means Diablo 630 emulation mode; 1 means interactive mode; other values are reserved for future capabilities.
59	Enables the auto line-feed feature of the Diablo 630 when set to 1.
60	Selects the pitch, or number of characters per inch (dpi). Reasonable values for pitch are 10, 12, and 15; the default value of 0 selects 10 dpi.
61	Selects the font number for the bold font to be used for Diablo 630 emulation. If the font number is 0 (Courier), then a 1 selects Courier-Bold. To select Courier as the bold font (effectively disabling the bold text option), use some illegal font number such as 255.
62	Selects the plain font used for Diablo 630 emulation. The default value of 0 selects Courier.
63	Used internally.

You can write new values to the EEROM, which can store the persistent parameters only a limited number of times before wearing out. Each location in the EEROM is capable of approximately 10,000 writes. For this reason, you should use the EEROM only for parameters that you expect to change infrequently. The copy count is an exception; it is implemented in such a way that the wear is distributed over many locations.

- ❖ *Note:* The LaserWriter IINTX use ZPRAM in place of the EEROM. ZPRAM can be written to an infinite number of times. Therefore, the statements just given regarding storage of the persistent parameters in the EEROM does not apply to the LaserWriter IINTX.

When you switch the LaserWriter on, it checks the contents of the EEROM for consistency, and uses the entry `eerom` in `statusdict` to report the result. Normally, `eerom` contains the value `TRUE`. If an inconsistency is detected, the value of `eerom` is redefined to be a 512-character PostScript string into which the entire contents of the EEROM are read. Then the page count is set to zero, and all parameters are reset to default values. If the EEROM fails completely, `eerom` is set to `FALSE`, and the software shifts to a simulation of the EEROM parameters in RAM. All of the operations for setting and reading parameters continue to work, but the values are no longer saved when the LaserWriter is switched off.

---

## Deviations from Diablo 630 protocol

The LaserWriter emulates the Diablo 630 as closely as possible; however, there are some important differences.

### Detecting the end of a document

The LaserWriter can detect the end of a document only by noticing that data has stopped arriving. After the LaserWriter processes the last page, all Diablo 630 settings such as margins, tabs, and spacing stay in effect for about 30 seconds (unless you change the wait timeout). Then a reset operation automatically restores all settings to their standard values; that is, the operation clears the margins, sets spacing to normal, and clears tab settings and any special word-processing modes. The LaserWriter prints a page when it either reaches the bottom of the page or receives a form feed (Command-L). If the last page of a document isn't full and doesn't end with a form feed character, the page won't be printed immediately. Instead, it will be printed either when the LaserWriter resets approximately 30 seconds later, or as part of the next document (at the top of the first page). To allow documents to be printed in close succession, make sure that each one has a final form feed character so that they do not run together.

## **Double-striking for bold text**

Some text processors produce a bold style by double-striking a character. Characters printed in this way do not appear as bold in the LaserWriter. You must use the correct Diablo 630 command sequence (Escape-O) to print bold characters.

## **Proportional fonts**

Times Roman and Helvetica are narrow proportional fonts that may look squeezed if the word processor does not adjust the page width. Few non-Macintosh text-processing programs are capable of producing correctly formatted output by using proportionally spaced fonts like these.

## **Paper positioning**

The Diablo 630 emulator uses exact positioning on the paper. Output from a word processor that attempts to compensate for slippage during vertical paper movement may appear slightly uneven.

## **Unsupported features and commands**

The LaserWriter does not support the Diablo 630 commands for the following features:

- print suppression
- HY-Plot
- extended character set
- the ability to download information for print wheels, directly or in program mode
- the ability to override print-wheel spacing for proportional spacing (although the offset for proportional spacing can be changed)
- page lengths other than 11 inches
- paper feeder control
- hammer energy control
- remote diagnostic tests
- The LaserWriter and LaserWriter IINT do not support DTR flow control

---

---

## Using the Hewlett-Packard LaserJet+ Emulator

This section describes the LaserWriter IINTX implementation of the Hewlett-Packard LaserJet+ emulator.

❖ *Note:* The Hewlett-Packard LaserJet+ emulator is referred to as the *LaserJet+ emulator* throughout this section.

In LaserJet+ emulation mode, the LaserWriter IINTX interprets incoming data as text and LaserJet+ control codes, rather than as a PostScript program. With this capability, the LaserWriter IINTX can print simple text files sent from host software packages that don't support PostScript, but that do support printing on the LaserJet or LaserJet+ printer.

With a few exceptions, the LaserJet+ emulator supports the entire set of escape sequences for both the LaserJet and the LaserJet+. This set includes the positioning commands; font-selection commands; and bit-mapped graphics with no limitation on areas covered by bit maps; as well as the rules, macros, overlays, and downloadable fonts. (See the *Hewlett-Packard LaserJet+ Technical Reference Manual*, January 1986 edition, for a complete description of the LaserJet+ features).

LaserJet+ printer programs that use the emulator have access to the roman, bold, italic, and bold italic fonts of the following built-in typefaces:

- Courier
- Times
- Helvetica

Unlike the fonts in the actual LaserJet+, all fonts in the LaserJet+ emulator can be scaled to any size, and used for printing either in portrait or landscape mode.

---

## Deviations from LaserJet protocol

To invoke the LaserJet<sup>+</sup> emulator, you set the mode switches on the back of the LaserWriter IINTX to the position indicated in Table 3-3, and connect one of the LaserWriter IINTX's serial ports to the workstation's RS-232 interface. You can send text to be printed at 9600 baud with any parity.

The remainder of this chapter covers the ways in which the LaserWriter IINTX implementation of the LaserJet<sup>+</sup> emulator differs from the actual LaserJet<sup>+</sup>.

---

## Character font selection

The LaserJet<sup>+</sup> emulator follows the font-selection algorithm as faithfully as possible. (This is the same algorithm described in the *LaserJet<sup>+</sup> Technical Reference Manual*.) The basic scheme is an ordering by priority of criteria used to choose the closest-available font from those fonts resident in the LaserJet<sup>+</sup> printer. The ordering scheme is as follows.

### Printing orientation (portrait versus landscape)

PostScript can rotate any font in portrait orientation to be in landscape orientation, so explicit fonts in landscape orientation need not be present. However, if a font in landscape orientation is downloaded by using the LaserJet<sup>+</sup> font-downloading capability, the LaserJet<sup>+</sup> emulator can successfully use it.

### Symbol set (Roman-8, Line draw, Math, and so on)

The built-in PostScript fonts are re-encoded for Roman-8 (USASCII plus Roman extension). Fonts with other symbol sets may be downloaded into the emulator.

### Print pitch (nonproportional fonts only)

The LaserJet<sup>+</sup> emulator scales an existing font to whatever pitch is requested.

### Character height (point size)

The LaserJet<sup>+</sup> emulator scales the height of characters, but the specified pitch has priority for fixed-pitch fonts. Asking for 10-pitch spacing and 14-point height results in the selection of 12-point Courier (unless a 14-point, 10-pitch font was previously downloaded).

### Stroke weight (light, medium, and bold)

The LaserJet<sup>+</sup> specifies a line weight between -7 and +7, with a weight of 0 specifying medium. The PostScript font-weight descriptions are mapped into this range, with standard bold having a weight of 3 and standard medium having a weight of 0. Like the LaserJet<sup>+</sup>, the LaserJet<sup>+</sup> emulator chooses the font with the closest weight if the font of the requested weight is not available.

### Typeface (Courier, Pica, and so on)

The built-in fonts—Courier, Helvetica, and Times—are mapped to the following LaserJet<sup>+</sup> typeface values:

- 3 Courier (Courier)
- 4 Helvetica (Helv)
- 5 Times Roman (Tms Rmn)

---

## Clipping region

The left margin of the LaserJet<sup>+</sup> defaults to the leftmost printable position on the page. The same is done for the LaserJet<sup>+</sup> emulator, but the image areas of the page on the two printers may not be exactly the same. The LaserJet<sup>+</sup> printer avoids printing any characters that would lie partially outside the printable region. The LaserJet<sup>+</sup> emulator does print them and allows them to be clipped. Both the printer and the emulator may throw away information if instructed to print completely outside the printable area (as in the Perforation Skip mode). The *LaserJet<sup>+</sup> Technical Reference Manual* points out this danger.

---

## Paper-size interactions

The LaserJet<sup>+</sup> printer provides commands that allow you to change the paper interactively during the course of a printing job. These commands include the `ESC&1#H` for manual feed of paper or envelopes, and `ESC&1#P` to change between such paper sizes as Letter, A4, and Legal. The LaserJet<sup>+</sup> emulator treats these commands as form-feed requests, but does not halt printing for changing of the paper. In the case of `ESC&1#P`, if # is less than the maximum lines for the current paper size, the number of lines per page is set to # (as if `ESC&1#F` had been used). The paper size that the LaserJet<sup>+</sup> emulator uses is the size that the server loop passes to it when the emulator job is executed.

---

## Symbol set

The LaserJet<sup>+</sup> escape sequences define a number of symbol sets (the supported sets are listed later in this chapter.) The most widely used symbol set is Roman8, an 8-bit extension of the ASCII symbol set. The Roman8 symbol set is the symbol set for all of the LaserJet<sup>+</sup> printer's built-in fonts (although on the LaserJet<sup>+</sup> printer, either half of the font is considered a 7-bit font). The LaserJet<sup>+</sup> emulator uses the PostScript interpreter's capability of re-encoding a typeface to produce Roman8 versions of the built-in PostScript typefaces Courier, Times, and Helvetica.

There are 11 characters in the Roman8 symbol set that are not in the Adobe standard symbol set and that are not printed. These characters are as follows:

Decimal 127	Gray patch for rubout
Decimal 176	Overline
Decimal 179	Degree symbol
Decimal 227	Uppercase D with stroke (Eth)
Decimal 228	Lowercase d with stroke (eth)
Decimal 240	Uppercase Thorn
Decimal 241	Lowercase Thorn
Decimal 247	One-fourth symbol
Decimal 248	One-half symbol
Decimal 252	Solid black square
Decimal 254	Plus/minus sign

---

## Character widths

The Times and Helvetica PostScript typefaces (licensed from Linotype Company) have the character metrics for which they were originally designed. The LaserJet<sup>+</sup> approximations of these typefaces, called *Tms Rmn* and *Helv*, respectively, have copied the glyphs, but have other metrics. In fact, several versions of Tms Rmn and Helv have different metrics, even for the same point size, and these metrics do not scale with the dimensions of the font. There are separate width tables for each point size of the typeface. As a result, applications that produce justified text in Helvetica or Times Roman do not have properly justified output when printed on the LaserJet<sup>+</sup> emulator. In the somewhat-limited sample of ragged Times Roman and Helvetica output that Adobe Systems has evaluated, the PostScript typeface widths are close enough so that the appearance of the right margin is not objectionable.

If users want to print justified documents in proportional-spaced fonts using the LaserJet<sup>+</sup> emulator, they can purchase a downloadable bit-mapped font design for a LaserJet<sup>+</sup>, download it into the emulator, and use an application that knows the widths of the downloadable fonts. Unlike the actual fonts in the LaserJet<sup>+</sup>, downloaded fonts can print in either portrait or landscape orientation. The LaserJet<sup>+</sup> emulator prints justified documents in Courier properly, because Courier is a monospaced font that prints with the same width on the emulator as on the actual LaserJet<sup>+</sup>. Users will most likely purchase a LaserWriter IINTX for the primary reason that they plan to do their desktop publishing by using the capabilities of the PostScript interpreter. They will use the LaserJet<sup>+</sup> emulator for applications such as accounting packages that are likely to print in Courier anyway.

---

## Typeface size

PostScript typefaces can be scaled, whereas LaserJet<sup>+</sup> printer fonts cannot be. Thus, your results with the LaserJet<sup>+</sup> emulator probably will be more precise than those you create with the actual LaserJet<sup>+</sup>. There is, in fact, an approximation margin in the LaserJet<sup>+</sup> emulator for picking font height, pitch (currently 0.4 points), or both. If you ask for 16.67 pitch and then ask for 17 pitch, the emulator will reuse the 16.67 pitch. On the LaserJet<sup>+</sup>, if you ask for 12 pitch but 10 pitch is the closest available on the printer, it will produce 10 pitch. However, under the same conditions, the LaserJet<sup>+</sup> emulator would provide 12 pitch. The same holds true for selecting point sizes.

---

## Line printer font

The LaserJet<sup>+</sup> printer has a built-in line printer font, which is a 16.67-pitch, 8.5-point font. When the line printer font is called, Courier is scaled to 16.67 pitch but, because it then has a height of 7.2 points, it is shorter than the LaserJet<sup>+</sup> printer built-in font. Users might find the bold version of the font easier to read, as the stroke weights of 7-point Courier are rather light.

---

## Transparent communications

The bit-mapped graphics operators of the LaserJet<sup>+</sup> require that 8-bit data be transmitted to the printer. Hence, when the LaserWriter IINTX is in the LaserJet<sup>+</sup> emulator mode, it configures the communications parameters so that all two-hundred-fifty-six 8-bit characters are transmitted uninterpreted to the emulator. Doing this eliminates the ^T status request and the ^C job-interrupt commands. In addition, doing this eliminates any way of specifying the end-of-file (^D for PostScript jobs).

---

## Longevity of downloaded information

Storage management in a LaserWriter IINTX is implemented by means of the `save` and `restore` operators. In particular, there is an implicit `save` before each PostScript job and an implicit `restore` after the job's completion. Thus, all virtual memory used by that job is recovered at the end of the job. The LaserJet<sup>+</sup> emulator also runs as a series of jobs with the same type of storage reclamation. An EOF on the input file signals the end of a PostScript job. Because transparent communication makes sending an EOF impossible in the LaserJet<sup>+</sup> emulator, the only end-of-job in LaserJet<sup>+</sup> emulation mode is either a manual reset or an I/O wait timeout. The user may set the length of the timeout period, with infinite being one of the options.

The LaserJet<sup>+</sup> allows fonts and macros to be downloaded into the printer for use in printing subsequent pages. On the LaserJet<sup>+</sup>, there is a hierarchy of temporary and permanent for both fonts and macros. Using either a printer reset (ESC E) or explicit escape sequences deletes temporary ones. Permanent ones are deleted when the machine is turned off (or by using other explicit escape sequences). When fonts or macros are deleted, their space becomes available for reuse by subsequently downloaded fonts or macros.

---

## LaserJet<sup>+</sup> control codes

Most of the information about serial communication in Appendix A applies to LaserJet<sup>+</sup> emulation. However, control characters have different meanings in LaserJet<sup>+</sup> emulation mode; all characters are interpreted according to the LaserJet<sup>+</sup> protocol, as shown in Table 3-8.

**Table 3-8**  
LaserJet+ control codes

Control	Escape sequence	Meaning
<b>Page orientation</b>	ESC&l00	Portrait
	ESC&l10	Landscape
<b>Font selection</b>	ESC(8U	Roman8 symbol set
	ESC(OU	USASCII symbol set
	ESC(s1P	Proportional character spacing
	ESC(s0P	Fixed character spacing
	ESC(s10H	10 characters per inch pitch
	ESC(s12H	12 characters per inch pitch
	ESC(s16.6H	16.66 characters per inch pitch
	ESC(s7C	7-point character size
	ESC(s8C	8-point character size
	ESC(s8.5C	8.5-point character size
	ESC(s10C	10-point character size
	ESC(s12C	12-point character size
	ESC(s14.4C	14.4-point character size
	ESC(s0S	Upright character style
	ESC(s1S	Italic character style
	ESC(s-3B	Line (stroke) light weight
	ESC(s0B	Line (stroke) medium weight
	ESC(s3B	Line (stroke) bold weight
	ESC(s3T	Courier
	ESC(s0T	Line Printer
ESC(s4T	Helvetica	
ESC(s5T	Times Roman	
ESC(s6T	Gothic	
ESC(s8T	Prestige Elite	
<b>Page length</b>	ESC&1#P	Number (#) of lines per page
	ESC&1#E	Number (#) of lines at top margin
	ESC&1#F	Number (#) of lines at bottom margin
<b>Margins</b>	ESC9	Clear margin
	ESC&a#L	Left-margin column number (#)
	ESC&a#M	Right-margin column number (#)

**Table 3-8** (continued)  
LaserJet+ control codes

Control	Escape sequence	Meaning
<b>Vertical line spacing</b>	ESC&1#C	Number # of $\frac{1}{8}$ inch increments
	ESC&l1D	1 line per inch
	ESC&l2D	2 lines per inch
	ESC&l3D	3 lines per inch
	ESC&l4D	4 lines per inch
	ESC&l6D	6 lines per inch
	ESC&l8D	8 lines per inch
	ESC&l12D	12 lines per inch
	ESC&l14D	14 lines per inch
	ESC&l16D	16 lines per inch
	ESC&l24D	24 lines per inch
	ESC=	Half-line feed
<b>Raster graphics</b>	ESC*t75R	75 dots per inch
	ESC*t100R	100 dots per inch
	ESC*t150R	150 dots per inch
	ESC*t300R	300 dots per inch
	ESC*r0A	Start raster graphics at leftmost position
	ESC*r1A	Start raster graphics at current position
	ESC*b#W [ <i>data</i> ]	Transfer raster graphics by number (#) of rows
ESC*rB	End raster graphics	
<b>Cursor positioning</b>	ESC&a#R	Row number (#)
	ESC&a#C	Column number (#)
	ESC&a#H	Horizontal number (#) of decipoints
	ESC&a#V	Vertical number (#) of decipoints
<b>Underline</b>	ESC&dD	Underline on
	ESC&d@	Underline off
<b>Display functions</b>	ESCY	Display functions on
	ESCZ	Display functions off
	ESC&p#X [ <i>data</i> ]	Transparent print data number (#) of bytes
<b>Perforation skip</b>	ESC&l0L	Disable perforation skip
	ESC&l1L	Enable perforation skip
<b>Miscellaneous features</b>	ESC&k#H	Horizontal motion index number (#) of $\frac{1}{20}$ inch
	ESC&k0S	Standard font pitch
	ESC&k2S	Compressed font pitch

**Table 3-8** (continued)  
LaserJet+ control codes

Control	Escape sequence	Meaning
<b>Line termination</b>	ESC&k0G	CR=CR, LF=LF, FF=FF
	ESC&k1G	CR=CR+LF, LF=LF, FF=FF
	ESC&k2G	CR=CR, LF=CR+LF, LF=LF, FF=CR+FF
	ESC&k3G	CR=CR+LF, LF=CR+LF, LF=LF, FF=CR+FF
	ESC&s0C	Enable end-of-line wrap
	ESC&s1C	Disable end-of-line wrap
	ESC&l#X	Select number (#) of copies
<b>Paper input control</b>	ESC&l0H	Eject page
	ESC&l1H	Feed from tray
	ESC&l2H	Manual feed
	ESC&l3H	Envelope feed
<b>Cursor position</b>	ESC*p#X	Horizontal cursor position in number (#) of dots
	ESC*p#Y	Vertical cursor position in number (#) of dots
<b>Font management</b>	ESC*c#D	Font ID number (#)
	ESC*c#E	Character code ASCII code number (#); decimal
	ESC*c0F	Font and Character control; delete all fonts
	ESC*c1F	Font and Character control; delete all Temp. fonts
	ESC*c2F	Delete last Font ID specified
	ESC*c3F	Delete last Font ID and Character code
	ESC*c4F	Make temporary font
	ESC*c5F	Make permanent font
	ESC*c6F	Copy or assign font
	ESC)s#W [data]	Create font header number (#) of bytes
	ESC(s#W [data]	Download Character number (#) of bytes
	ESC(#X	Designate Primary Download Character Font ID number (#)
	ESC)#X	Designate Secondary Download Character Font ID number (#)
	ESC(0@	Font defaults; primary font value 0
	ESC(1@	Font defaults; primary font value 1
	ESC(2@	Font defaults; primary font value 2
	ESC(3@	Font defaults; primary font value 3
	ESC)0@	Font defaults; secondary font value 0
	ESC)1@	Font defaults; secondary font value 1
ESC)2@	Font defaults; secondary font value 2	
ESC)3@	Font defaults; secondary font value 3	

**Table 3-8** (continued)  
LaserJet+ control codes

Control	Escape sequence	Meaning
<b>Macros</b>	ESC&f#Y	Macro ID number (#)
	ESC&f0X	Start macro
	ESC&f1X	Stop macro
	ESC&f2X	Execute macro
	ESC&f3X	Call macro
	ESC&f4X	Enable overlay macro
	ESC&f5X	Disable overlay macro
	ESC&f6X	Delete macro
	ESC&f7X	Delete all temporary macro
	ESC&f8X	Delete macro ID
	ESC&f9X	Make temporary macro
	ESC&f10X	Make permanent macro
	ESC&f0S	Push position
	ESC&f1S	Pop position
<b>Advanced graphics</b>	ESC*c#A	Horizontal rule/pattern size number (#) of dots
	ESC*c#H	Horizontal rule/pattern size number (#) of decipoints
	ESC*c#B	Vertical rule/pattern size number (#) of dots
	ESC*c#V	Vertical rule/pattern size number (#) of decipoints
	ESC*c0P	Print rule
	ESC*c2P	Print Grayscale
	EsC*c3P	HP pattern
	ESC*c2G	Grayscale pattern ID; 2% gray
	ESC*c10G	Grayscale pattern ID; 10% gray
	ESC*c15G	Grayscale pattern ID; 15% gray
	ESC*c30G	Grayscale pattern ID; 30% gray
	ESC*c45G	Grayscale pattern ID; 45% gray
	ESC*c70G	Grayscale pattern ID; 70% gray
	ESC*c90G	Grayscale pattern ID; 90% gray
	ESC*c100G	Grayscale pattern ID; 100% gray
	ESC*c1G	HP pattern ID; 1 vertical lines
	ESC*c2G	HP pattern ID; 2 horizontal lines
	ESC*c3G	HP pattern ID; 3 diagonal lines
	ESC*c4G	HP pattern ID; 4 diagonal lines
	ESC*c5G	HP pattern ID; 5 grid
	ESC*c6G	HP pattern ID; 6 diagonal grid



## Chapter 4



# PostScript Language for the LaserWriter II NT and LaserWriter II NTX

This chapter refers to features available in the PostScript interpreter, version 47.0 or higher, and contains details about the programming and operation of the LaserWriter IINT and LaserWriter IINTX. The contents of this chapter parallel the contents of Appendix D in the *PostScript Language Reference Manual*. In order to present a comprehensive reference, this chapter incorporates most of the System Parameter information from the *Adobe PostScript Language Supplement for the LaserWriter IINT and LaserWriter IINTX*.

This chapter is useful for programmers of host software that accesses the LaserWriter IINT or LaserWriter IINTX, and for programmers who are interested in configuring the LaserWriter IINT or LaserWriter IINTX in nonstandard ways.

---

---

## Basic operation of the LaserWriter IINT and LaserWriter IINTX

The LaserWriter IINT and the LaserWriter IINTX operate similarly to the original LaserWriter and LaserWriter Plus. Software that was written for the original LaserWriter or LaserWriter Plus will perform equally well on both the LaserWriter IINT and the LaserWriter IINTX. However, the PostScript operators covered in this chapter can help you to take advantage of the added features of these new printers.

The user can change many aspects of the LaserWriter IINT or LaserWriter IINTX's operation. There is a collection of operators and other parameters in the special PostScript dictionary `statusdict`. While this section contains many of them, see "System Parameters" later in this chapter for complete documentation.

---

### Page types

The region of the page that can be produced as an image is subject to both hardware limits (the physical page size and margins required by the printing engine) and software constraints (the amount of memory available for the full-page frame buffer). The LaserWriter IINT and LaserWriter IINTX include built-in device-setup procedures for establishing any of seven standard page types, described here.

letter	A region of 8.00 by 10.78 inches that can be produced as an image, centered on an 8.50-by-11.00-inch page (that is, with 0.25-inch margins on left and right and 0.11-inch margins on top and bottom). This is the standard page type for letter-size paper.
legal	A region of 8.00 by 13.78 inches that can be produced as an image, centered on an 8.50-by-14.00-inch page. This is the standard page type for legal-size paper.
a4	A region of 7.79 by 11.08 inches that can be produced as an image, centered on an 8.27-by-11.69-inch page. This page type is the standard one for the European A4-size paper.
b5	A region of 6.45 by 9.76 inches that can be produced as an image, centered on a 6.93-by-9.84-inch page. This is the standard page type for the European B5-size paper.
lettersmall	A region of 7.68 by 10.16 inches that can be produced as an image, centered on an 8.50-by-11.00-inch page.
a4small	A region of 7.47 by 10.85 inches that can be produced as an image, centered on an 8.27-by-11.69-inch page.
note	This page type selects one of the page types just described according to the paper tray that is installed. If the Letter or A4 tray is installed, note selects lettersmall or a4small, respectively. If the Legal or B5 tray is installed, note selects legal or b5. For the Letter and A4 paper sizes, this page type has the effect of increasing all four margins to approximately 0.42 inch. This reduction in the area that can be produced as an image frees up as much as 100,000 bytes of memory, which is added to the VM and made available for use by PostScript programs.

For compatibility with other PostScript printers, `statusdict` includes the following operators:

- `lettertray` Sets a `letter` page type. This operator raises the PostScript error `rangecheck` if the paper tray contains paper of a size other than letter.
- `legaltray` Sets a `legal` page type. This operator raises the PostScript error `rangecheck` if the paper tray contains paper of a size other than legal.
- `a4tray` Sets an `a4` page type. This operator raises the PostScript error `rangecheck` if the paper tray contains paper of a size other than A4.
- `b5tray` Sets a `b5` page type. This operator raises the PostScript error `rangecheck` if the paper tray contains paper of a size other than B5.

For all page types, the point 0,0 in default user-coordinate space is the lower-left corner of the entire page when viewed in a portrait orientation (in which the long edge is vertical). This 0,0 point is not at the lower-left corner of the region that can be produced as an image; that is, the origin lies some distance outside the lower-left corner of the that region. The coordinate system is arranged in this way so that switching page types does not affect the position of graphical objects on the paper, but only changes the sizes of the margins.

At the beginning of each job, the software detects whether a letter-, legal-, B5-, or A4-size paper tray is installed and sets the default page type automatically. A PostScript program can override the default page type by executing one of the procedures just described.

---

## Manual feed

The user can feed individual sheets of paper or other material (such as envelopes, transparency stock, and so on) into the LaserWriter IINTX manually. If a program defines `manualfeed` to be true in `statusdict`, the printer does not take paper from

the paper tray during subsequent showpage operations. Instead, for each page printed, the red light begins to blink and the printer waits for a sheet of paper to be inserted into the slot on top of the paper tray. If no paper is inserted within manualfeedtimeout seconds, a timeout error occurs and the job is aborted.

---

## Timeouts

There is a timeout facility for limiting the amount of time the server will remain in various states. There are three timeouts of interest: the job timeout, the manual feed timeout, and the wait timeout. At the beginning of a job, these timeouts are set to default values (initially 0, 60, and 30 seconds, respectively). A PostScript program can change the timeouts for the current job if you want it to. The operators for controlling timeouts are located in `statusdict` and are described later in this chapter.

The manual feed timeout was described earlier. If nonzero, the job timeout limits the total amount of time the job will execute; this limitation protects the LaserWriter IINTX from being tied up by a PostScript program that runs for an unexpectedly long time. Such a program itself can restart the timer at any time during the job if that is desirable.

If nonzero, the wait timeout limits the time the server will wait to receive additional input for a job that is in progress; this limitation protects the LaserWriter IINTX from being tied up indefinitely by a host that crashes or that is disconnected in the midst of sending a file to the server.

If a nonzero job or wait timeout has been set and if it expires, the PostScript interpreter executes the timeout error from `errordict`. With the standard definition of timeout, this causes a job running in batch mode or emulation mode to terminate. The timeout facility is not ordinarily enabled when the LaserWriter IINTX is in interactive mode.

---

## Idle-time font scan-conversion

While waiting for a job to begin (that is, before the first character of a new job has arrived from a current communication channel), the server uses the available time to scan-convert a standard selection of characters in commonly used fonts and point sizes, and to load the results into the font cache. If a subsequent page description happens to use those characters, it will execute faster than it otherwise would. If disk storage is available, idle-time caching will cache onto the disk-file system, as well as into the RAM font cache. Because this feature generates unnecessary disk activity, it is not turned on in LaserWriter IINTX. However, it can be turned on explicitly with PostScript operators.

In the LaserWriter IINTX, the characters normally scan-converted during idle time are as follows (the sizes marked with an asterisk are prescanned and permanently resident in ROM):

- Courier 10 point\*, full ASCII set (intended for program listings and other 'line printer' applications)
- Times-Roman and Helvetica both in 10 point\* and 12 point\*, alphanumerics and common punctuation
- Times-Roman 14 point\*, alphanumerics
- Helvetica 14 point, alphanumerics
- Times-Bold and Helvetica-Bold both in 10 point\*, 12 point\*, and 14 point, lowercase letters only
- Courier-Bold 10 point, lowercase letters only

You may override the standard selection of fonts to be scan-converted during idle time (except for the ones stored in ROM) by using the `setidlefonts` operator in `statusdict`. Each font to be scan-converted is specified by this sequence of five integers:

*font s<sub>x</sub>10 s<sub>y</sub>10 rot/5 nchars*

where *font* is a font number taken from the table given next, *s<sub>x</sub>* and *s<sub>y</sub>* are the scale factors for *x* and *y*, *rot* is the rotation in degrees, and *nchars* is the number of characters to be converted. The font numbers are given in Table 4-1.

**Table 4-1**  
Font numbers

Number	Font name	Number	Font name
0	Courier	18	Bookman-DemiItalic
1	Courier-Bold	19	Bookman-Light
2	Courier-Oblique	20	Bookman-LightItalic
3	Courier-BoldOblique	21	Helvetica-Narrow
4	Times-Roman	22	Helvetica-Narrow-Bold
5	Times-Bold	23	Helvetica-Narrow-BoldOblique
6	Times-Italic	24	Helvetica-Narrow-Oblique
7	Times-BoldItalic	25	NewCenturySchlbk-Roman
8	Helvetica	26	NewCenturySchlbk-Bold
9	Helvetica-Bold	27	NewCenturySchlbk-Italic
10	Helvetica-Oblique	28	NewCenturySchlbk-BoldItalic
11	Helvetica-BoldOblique	29	Palatino-Roman
12	Symbol	30	Palatino-Bold
13	AvantGarde-Book	31	Palatino-Italic
14	AvantGarde-BookOblique	32	Palatino-BoldItalic
15	AvantGarde-Demi	33	ZapfChancery-MediumItalic
16	AvantGarde-DemiOblique	34	ZapfDingbats
17	Bookman-Demi		

The characters converted are the first *nchars* characters of the following set, which contains 94 characters in all:

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789.,;?:-O'!+[]$%&*/_=#'{}<>^~|\
```

For example, the following sequence of numbers (used as operands to `setidlefonts`) would specify conversion of all lowercase and uppercase alphabetic characters of Helvetica-Bold in a 12-point size, narrowed by the ratio 10/12, and rotated by 90 degrees:

```
9 100 120 18 52
```

The complete set of fonts to be scan-converted is specified as a sequence of integers, interpreted in groups of five as just described. If the sequence is empty, the standard fonts are converted. See the description of `setidlefonts` later in this chapter for more information.

---

---

## System parameters

The LaserWriter IINTX has a fairly extensive collection of parameters that control its operation. Some of these parameters are stored in nonvolatile memory (ZPRAM), so they persist even when the machine is turned off. Other parameters are volatile and generally remain in effect only through the execution of a single job. This section documents both types of parameters.

All the system parameters are accessed through the special dictionary `statusdict`, which is separate from `systemdict` and `userdict`, where all standard PostScript operators and procedures are defined. Changing system parameters requires that you send the LaserWriter IINTX a PostScript program that accesses `statusdict` explicitly.

The easiest way to gain access to `statusdict` is to execute `statusdict begin`, which pushes `statusdict` onto the dictionary stack. Until the matching `end` occurs, the operators defined in `statusdict` may be invoked directly simply by executing their names. Some system parameters are stored in `statusdict` as ordinary data values (such as integers, booleans, and strings) that may be read by executing their names, and changed by using the `def` operator.

---

## Changing persistent parameters

Ordinarily, the server opens and closes each job with `save` and `restore` so that changes made to the VM by the job do not persist into the next job. To make permanent changes—for example, to install additional font definitions—you must escape from this context and execute a job that is not opened and closed by `save` and `restore`. You must also do this to execute any of the `statusdict` operators that change the persistent (nonvolatile) parameters.

A password controls the ability to make permanent changes. Some LaserWriter IINTXs are used in a shared environment in which it is undesirable for individual users to change a server's persistent state. In such cases, only a system administrator should be permitted to make such changes. But in the case of a LaserWriter IINTX that is dedicated to a single user or a small group of cooperative users, you should permit the users to make changes freely.

The system-administrator password is a PostScript integer. The default value is 0, but you may change it to any other value by executing the `setpassword` operator.

To escape from the normal server `save/restore` context, issue the PostScript statement

```
serverdict begin password exitserver
```

where *password* is the system-administrator password. If the password is incorrect, `exitserver` executes the error `PasswordIncorrect` (which immediately invokes `stop`, bypassing `errordict`). If the password is correct, however, `exitserver` responds with the message “%%[ `exitserver`: permanent state may be changed ]%%” as an acknowledgment of the `exitserver` request. The `exitserver` request then performs an implicit `restore`, and clears the operand and dictionary stacks as if it were preparing to execute the next job; however, it does not perform another `save`.

A PostScript program executed between a successful `exitserver` and the next EOF is permitted to invoke the `statusdict` operators that change persistent parameters. Additionally, all changes made by that program to the state of the PostScript VM—such as creating new objects, storing values into dictionaries, and so on—persist until the printer is turned off; the modified VM appears as the initial state of all subsequent jobs.

During execution of this type of PostScript program, the VM is not protected from harmful changes that could cause the server to malfunction. (This lack of protection permits the server itself to be patched, should that become necessary.) Also, VM resources consumed by this type of program remain in use indefinitely; there is no way to reclaim them other than by turning the LaserWriter IINTX off and then on.

---

## Persistent parameters

The `statusdict` operators for accessing persistent parameters are described next. The volatile parameters are dealt with later in this chapter.

To invoke any of the operators that change persistent parameters, you first must escape from the normal server `save/restore` context as just described; if you do not, an `invalidaccess` error will occur. The operators for which this procedure is required are marked with an asterisk (\*).

## **pagecount**

**Syntax** `pagecount int`

**Definition** The operator `pagecount` returns the number of pages (*int*) that have been printed by the LaserWriter IINT and the LaserWriter IINTX. (There is no way to reset this value.)

**Errors** `stackoverflow`

## **pagestackorder**

**Syntax** `pagestackorder bool`

**Definition** The operator `pagestackorder` returns the last boolean value (*bool*) set by `setpagestackorder`, described next: the value is TRUE if the second page printed faces the back of the first page when it is stacked in the output tray; or is FALSE if the second page faces away from the first.

**Errors** `stackoverflow`

## **setpagestackorder\***

**Syntax** `bool setpagestackorder`

**Definition** The operator `setpagestackorder` sets the value (*bool*) returned by `pagestackorder`, just described. A value of TRUE indicates that output is going to the face-up paper output tray; a value of FALSE indicates that output is directed to the face-down paper output tray. Since the LaserWriter IINTX cannot detect which tray the output is directed to, it is up to the user to ensure that the value `pagestackorder` is correctly set.

**Standard value** FALSE

**Errors** `stackunderflow`, `typecheck`

## **ramsize**

<b>Syntax</b>	<code>ramsize <i>int</i></code>
<b>Definition</b>	The operator <code>ramsize</code> returns the number of bytes ( <i>int</i> ) of RAM in the LaserWriter IINTX.
<b>Errors</b>	<code>stackoverflow</code>

## **setprintrname\***

<b>Syntax</b>	<code>string setprintrname</code>
<b>Definition</b>	The operator <code>setprintrname</code> establishes <i>string</i> to be the printer's name. This variable <i>string</i> is printed on the test page at power-on time; it also defines the name used to identify this LaserWriter IINTX on AppleTalk. It should be 31 characters or fewer, should consist entirely of printable characters, and should not contain the characters <code>:</code> or <code>@</code> .
<b>Errors</b>	<code>invalidaccess</code> , <code>rangecheck</code> , <code>stackunderflow</code> , <code>typecheck</code>

## **printrname**

<b>Syntax</b>	<code>string printrname <i>substring</i></code>
<b>Definition</b>	The operator <code>printrname</code> stores the printer's name into the variable <i>string</i> (overwriting some initial portion of its value) and returns a string object that designates the part of the string ( <i>substring</i> ) actually used.
<b>Standard value</b>	The standard value depends on the model and may be defined by the user.
<b>Errors</b>	<code>invalidaccess</code> , <code>rangecheck</code> , <code>stackunderflow</code> , <code>typecheck</code>

## **setdosysstart\***

<b>Syntax</b>	<code>bool setdosysstart</code>
<b>Definition</b>	The operator <code>setdosysstart</code> sets whether or not the LaserWriter IINTX will execute a Sys/Start file at power-on time.
<b>Errors</b>	<code>stackoverflow</code> , <code>typecheck</code>

## **dosysstart**

<b>Syntax</b>	<code>dosysstart <i>bool</i></code>
<b>Definition</b>	The operator <code>dosysstart</code> returns the current setting ( <i>bool</i> ) of whether or not the LaserWriter IINTX will execute a Sys/Start file at power-on time.
<b>Standard value</b>	FALSE
<b>Errors</b>	<code>stackoverflow</code>

## **setsoftwareiomode\***

<b>Syntax</b>	<code><i>setting</i> setsoftwareiomode</code>
<b>Definition</b>	<p>The operator <code>setsoftwareiomode</code> sets the software interface for communication between the LaserWriter IINTX and the controlling host to the value designated by the argument <i>setting</i>.</p> <p>The possible <i>setting</i> values are as follows:</p> <ul style="list-style-type: none"><li>0 PostScript batch</li><li>1 PostScript interactive</li><li>2 Diablo 630 emulation</li><li>3,4 not used</li><li>5 HP LaserJet Plus emulation</li></ul> <p>The new value of <i>setting</i> does not take effect until the end of the current job.</p>
<b>Errors</b>	<code>rangecheck</code> , <code>stackunderflow</code> , <code>typecheck</code>

## **softwareiomode**

<b>Syntax</b>	<code>softwareiomode <i>integer</i></code>
<b>Definition</b>	The operator <code>softwareiomode</code> returns the current setting ( <i>integer</i> ) of the software I/O mode. (See <code>setsoftwareiomode</code> , just given, for a description of the integer values.)
<b>Standard value</b>	0
<b>Errors</b>	<code>rangecheck</code> , <code>stackoverflow</code> , <code>stackunderflow</code> , <code>typecheck</code>

## sethardwareiomode

**Syntax** `setting sethardwareiomode`

**Definition** The operator `sethardwareiomode` sets the hardware method of communication between the LaserWriter IINTX and the controlling host to the value designated by the argument *setting*. The possible values of *setting* are as follows:

- 0 serial
- 1 not used
- 2 AppleTalk

The new value of *setting* does not take effect until the end of the current job.

**Errors** `rangecheck, stackunderflow, typecheck`

## hardwareiomode

**Syntax** `hardwareiomode integer`

**Definition** The operator `hardwareiomode` returns the current setting (*integer*) of the hardware I/O mode. (See `sethardwareiomode`, just given, for a description of the integer values.)

**Standard value** 0

**Errors** `rangecheck, stackoverflow, stackunderflow, typecheck`

## setsccbatch\*

**Syntax** `channel baud options setsccbatch`

**Definition** The operator `setsccbatch` sets communication parameters as specified by three integers—*channel*, *baud*, and *options*—that designate serial channel (9 or 25), the baud, and the options, respectively. These integers determine how serial communication is to be performed on that channel when serial communication is selected as the `hardwareiomode`. These parameters may be set independently for each of the two channels.

- ❖ *Note:* The LaserWriter IINT and the LaserWriter IINTX both use the MINI DIN-8 connector, rather than the 9-pin connector used on the original LaserWriter. However, the serial channel variable for the LaserWriter IINT and LaserWriter IINTX is still specified by the integer 9.

The integer *options* encodes parity, flow control, number of data bits, and number of stop bits to be used in serial-communication mode. If you view *options* as a binary integer, you encode the different fields as follows: Parity is the least-significant 2 bits (bits 0 and 1). Flow control is the next most-significant 3 bits (bits 3 through 5). Number of data bits is the next most-significant 2 bits (bits 5 and 6), number of stop bits is the most-significant bit (bit 7). The proper field value to use for the different settings is given next.

Another way to determine the value of the integer *options* is to add decimal integers listed in Table 4-2, selecting one value for each field.

- ❖ *Note:* The LaserWriter IINT does not work in 7-bit mode with mark parity selected and only works in 8-bit mode with space parity selected.

**Table 4-2**  
setscbcbatch options

Field	Setting	Decimal value	Field value
Parity	space	0	0
	odd	1	1
	even	2	2
	mark	3	3
Flow control	xon/xoff	0	0
	dtr	4	1
	etx/ack	8	2
Data bits	std	0	0
	7	32	1
	8	64	2
Stop bits	1	0	0
	2	128	1

The new values of *baud* and *options* do not take effect until the end of the current job. Setting a channel's baud to 0 disables the channel; but disabling both channels is not permitted. Giving the following values to the operator setscbcbatch

```
25 19200 2 setscbcbatch
```

sets the 25-pin channel to 19,200 baud with even parity.

## Errors

```
invalidaccess, rangecheck, stackunderflow, typecheck
```

## sccbatch

<b>Syntax</b>	<i>channel sccbatch baud options</i>
<b>Definition</b>	The operator <code>sccbatch</code> returns the rate ( <i>baud</i> ) and options ( <i>options</i> ) for the specified serial channel, <i>channel</i> (9 or 25). These are the parameters used when <code>hardwareiomode</code> is 0, which selects serial communication.
<b>Standard value</b>	9600 0
<b>Errors</b>	<code>rangecheck</code> , <code>stackoverflow</code> , <code>stackunderflow</code> , <code>typecheck</code>

## setdostartpage\*

<b>Syntax</b>	<i>bool setdostartpage</i>
<b>Definition</b>	The operator <code>setdostartpage</code> specifies whether or not the LaserWriter IINT or the LaserWriter IINTX is to print a test page upon power-on.
<b>Errors</b>	<code>invalidaccess</code> , <code>stackunderflow</code> , <code>typecheck</code>

## dostartpage

<b>Syntax</b>	<code>dostartpage</code> <i>bool</i>
	The operator <code>dostartpage</code> returns the boolean ( <i>bool</i> ) that specifies whether or not the LaserWriter IINT or the LaserWriter IINTX is to print a test page at power-on.
<b>Standard value</b>	TRUE
<b>Errors</b>	<code>stackoverflow</code>

## setmargins\*

<b>Syntax</b>	<i>top left setmargins</i>
<b>Definition</b>	The operator <code>setmargins</code> adjusts the printer's margins, thereby changing the alignment of the area that can be produced as an image on the physical page. The operands <i>top</i> and <i>left</i> are integers that specify distances in device space (the unit size is one device pixel, or 1/300 inch). A positive value of <i>top</i> widens the top margin and a negative value of <i>top</i> narrows it relative to the standard margin width. (The top of the page is the edge that emerges first from the printer.) Similarly, a positive value of <i>left</i> widens the left margin and a negative value of <i>left</i> narrows it.

The operator `setmargins` is intended only for use at installation time to correct any physical alignment errors that may exist; it has nothing to do with setting the dimensions of the area that can be produced as an image (see `letter` and `legal` described earlier). There are limits to the range of adjustment that is possible. The printer hardware imposes margins that cause the image to be clipped if it is moved too close to the edge of the paper; unfortunately, the center determined by hardware-imposed margins is not the same as the center of the paper.

**Errors** `invalidaccess, rangecheck, stackunderflow, typecheck`

### **margins**

**Syntax** `margins top left`

**Definition** The operator `margins` returns the two margin adjustment parameters, `top` and `left`, that are set by `setmargins`.

**Standard value** `0 0`

**Errors** `stackoverflow`

### **setpagetype\***

**Syntax** `int setpagetype`

**Definition** The operator `setpagetype` specifies the default page type (`int`) to be used subsequently when any paper tray except the legal-size paper tray is installed. The value 0 selects page type `letter` and the value 1 selects `note`.

**Errors** `invalidaccess, rangecheck, stackunderflow, typecheck`

### **pagetype**

**Syntax** `pagetype int`

**Definition** The operator `pagetype` returns the default page type parameter, `int`, set by the operator `setpagetype`.

**Standard value** `0`

**Errors** `stackoverflow`

## setdefaulttimeouts\*

<b>Syntax</b>	<i>job manualfeed wait</i> setdefaulttimeouts
<b>Definition</b>	The operator <code>setdefaulttimeouts</code> establishes the default values— <i>job</i> , <i>manualfeed</i> , and <i>wait</i> —for the three timeouts. At the beginning of each job, these values are used to initialize the job, manual feed, and wait timeouts. (A PostScript program may change a timeout for the remainder of the current job by executing the <code>setjobtimeout</code> operator or changing the value <code>manualfeedtimeout</code> or the value <code>waittimeout</code> in <code>statusdict</code> .) Each parameter must be a nonnegative integer denoting the length of a timeout in seconds. The value 0 indicates that the corresponding timeout should never occur.
<b>Errors</b>	invalidaccess, rangecheck, stackunderflow, typecheck

## defaulttimeouts

<b>Syntax</b>	defaulttimeouts <i>job manualfeed wait</i>
<b>Definition</b>	The operator <code>defaulttimeouts</code> returns the default values <i>job</i> , <i>manualfeed</i> , and <i>wait</i> of the job, manual feed, and wait timeouts, respectively.
<b>Standard value</b>	0 60 30
<b>Errors</b>	stackoverflow

## setpassword

<b>Syntax</b>	<i>old new</i> setpassword <i>bool</i>
<b>Definition</b>	The operator <code>setpassword</code> sets the system-administrator password, which controls the ability to escape from the server <code>save/restore</code> context, and the ability to make persistent changes to system parameters or to the VM. The operator <code>setpassword</code> requires two integer operands: the password <i>old</i> and <i>new</i> . If <i>old</i> is the correct old password, <code>setpassword</code> changes the password to <i>new</i> and returns TRUE as the value of <i>bool</i> ; if <i>old</i> is not correct, <code>setpassword</code> returns FALSE as the value of <i>bool</i> .
<b>Standard value</b>	0
<b>Errors</b>	stackunderflow, typecheck

## checkpassword

<b>Syntax</b>	<i>int</i> checkpassword <i>bool</i>
---------------	--------------------------------------

**Definition** The operator `checkpassword` returns TRUE as the value of *bool* if *int* is equal to the current system-administrator password; `checkpassword` returns FALSE (after delaying for 1 second) in all other cases.

**Standard value** 0

**Errors** `stackunderflow`, `typecheck`

### **setidlefonts\***

**Syntax** `mark font  $s_x s_y$  rot  $nchars$  ... setidlefonts`

**Definition** The operator `setidlefonts` expects the operand stack to contain up to 150 integers in the range 0 to 255, delimited by *mark* immediately below them. The operator `setidlefonts` removes *mark* and the integers, and stores them permanently. The integers are interpreted in groups of five to specify characters to be scan-converted while the LaserWriter IINTX is idle. An empty list of integers (denoted by *mark* on the top of the operand stack) specifies that the standard set of characters is to be scan-converted.

**Errors** `invalidaccess`, `rangecheck`, `typecheck`, `unmatchedmark`

### **idlefonts**

**Syntax** `idlefonts mark font  $s_x s_y$  rot  $nchars$  ...`

**Definition** The operator `idlefonts` pushes *mark* followed by the integers controlling idle time scan-conversion (see `setidlefonts`, just described).

**Standard value** `mark`

**Errors** `stackoverflow`

### **seteescratch\***

**Syntax** `index value seteescratch`

**Definition** The operator `seteescratch` writes *value* at position *index* in an array in the LaserWriter EEROM (ZPRAM in the LaserWriter IINT and the LaserWriter IINTX) reserved for scratch use. Here, *index* must be an integer in the range 0 to 63; *value* must be an integer in the range 0 to 255. The EEROM scratch array is intended for storing persistent parameters that were not taken into account in the original design of the LaserWriter IINT or the LaserWriter IINTX. Several entries in the array have already been assigned; they are described next.

**Errors** `invalidaccess`, `rangecheck`, `stackunderflow`, `typecheck`

## eescratch

<b>Syntax</b>	<i>index</i> eescratch <i>value</i>
<b>Definition</b>	The operator <code>eescratch</code> returns the value at position <i>index</i> in the EEROM scratch array (see <code>seteescratch</code> , above, for a definition of the return values).
<b>Standard value</b>	0
<b>Errors</b>	rangecheck, stackunderflow, typecheck

---

## Additional persistent parameters

Several capabilities have been added to the LaserWriter IINT and LaserWriter IINTX since the standard set of persistent parameters, just described, was established. Notable among these new capabilities is the Diablo 630 emulator, described earlier. Entries in the LaserWriter IINT EEROM and LaserWriter IINTX ZPRAM scratch array (accessed by `seteescratch` and `eescratch`) have been assigned to control these capabilities. Table 4-3 summarizes the Diablo 630 persistent parameters.

For example, to enable the Diablo 630 auto line-feed feature, execute this PostScript program:

```
serverdict begin 0 exitserver
statusdict begin
59 1 seteescratch
```

**Table 4-3**  
Persistent parameters for the Diablo 630 emulator

Parameter	Index	Definition
Auto line-feed	59	The value 1 enables the Diablo 630 auto line-feed feature; any other value disables it.
Pitch	60	This parameter selects the Diablo 630 pitch (characters per inch). Reasonable values are 10, 12, and 15; the default value 0 selects 10.
Bold font	61	This parameter selects the bold font used for Diablo 630 emulation. This is a font number taken from Table 4-1, except that if the number is 0 (selecting Courier), then 1 (selecting Courier-Bold) is used instead. (To select Courier as the bold font, use some unreasonable font number, such as 255.)
Normal font	62	This parameter selects the normal font used for Diablo 630 emulation. This is a font number taken from Table 4-1. The default value 0 selects Courier.
	63	This index has an internal use that is not documented.

The EEROM in which the persistent parameters are stored can be written to only a limited number of times before it wears out. Each location in the EEROM is capable of being written approximately 10,000 times. For this reason, the EEROM is used only for parameters that are expected to change infrequently.

❖ *Note:* The description just given applies to the original LaserWriter, LaserWriter Plus, and LaserWriter IINT. The LaserWriter IINTX uses a ZPRAM, which can be written to indefinitely.

At power-on time, the LaserWriter IINTX's PostScript interpreter checks the contents of the ZPRAM for consistency; the interpreter reports the result by defining an entry named `eerom` in `statusdict`. Normally, the interpreter defines `eerom` to be TRUE. If it detects an inconsistency, it defines `eerom` to be a 512-character PostScript string (LaserWriter and LaserWriter IINT) 2048-character PostScript string (LaserWriter IINTX) into which the interpreter reads the entire contents of the ZPRAM; then it sets the page count to 0 and resets all system parameters to their default values. If the interpreter finds that the ZPRAM has failed altogether (perhaps because it has worn out), the interpreter shifts to a simulation of the ZPRAM parameters in RAM. All operations for setting and reading parameters continue to work, but newly set values no longer persist beyond power-off.

---

## Volatile parameters

The PostScript dictionary `statusdict` contains two operators—`setjobtimeout` and `jobtimeout`—with immediate effects that do not persist from one job to the next. The remaining `statusdict` entries are not operators but are ordinary data values such as booleans, integers, and strings. PostScript dictionary operators such as `get` and `put` may read and write these values in the usual way. There are no restrictions on changing these parameters; the effects of changes persist only until the end of the current job.

There are several additional `statusdict` entries that are not documented. These entries have to do with the operation of the server and are not intended for execution by user programs.

## setjobtimeout

**Syntax** `int setjobtimeout`

**Definition** The operator `settimeout` sets the timeout for the current job to the value *int*, a nonnegative integer that specifies a time interval in seconds. If the current job continues for *int* seconds without either completing or executing `setjobtimeout` again, the PostScript interpreter executes a `timeout` error. The value 0 disables the job timeout altogether.

At the beginning of a job, the server initially sets the job timeout to the default job timeout returned by `defaulttimeouts`. (However, in interactive mode, the initial job timeout is always 0.)

**Errors** `rangecheck`, `stackunderflow`, `typecheck`

## jobtimeout

**Syntax** `jobtimeout int`

**Definition** The parameter `jobtimeout` returns the number of seconds (*int*) remaining before the job timeout will occur. A value of 0 indicates that the job will never occur.

**Standard value** 0

**Errors** `stackoverflow`

## manualfeedtimeout

**Syntax** `manualfeedtimeout int`

**Definition** The operator `manualfeedtimeout` is the manual feed timeout currently in effect; that is, it is the number of seconds (*int*) that the LaserWriter IINTX will wait for a page to be inserted into the manual feed slot. This timeout applies only when the LaserWriter IINTX is in manual feed mode, when `manualfeed` is TRUE (see the description of `manualfeed` given later in this chapter).

At the beginning of a job, the server initializes `manualfeedtimeout` to the default manual feed timeout returned by `defaulttimeouts`; however, a PostScript program may change `manualfeedtimeout` to any nonnegative integer value (by using `def`, `put`, or `store`).

**Standard value** 60

### **waittimeout**

**Syntax**           waittimeout *int*

**Definition**       The parameter `waittimeout` is the wait timeout currently in effect; that is, it is the number of seconds (*int*) the LaserWriter IINTX will wait to receive additional characters from the host before giving up and aborting the current job by executing a `timeout`. At the beginning of a job, the server initializes `waittimeout` to the default wait timeout returned by `defaulttimeouts`. However, a PostScript program may change `waittimeout` to any nonnegative integer value. (However, in interactive mode, the value of `waittimeout` is always 0.)

**Standard value**   30

### **manualfeed**

**Syntax**           manualfeed *bool*

**Definition**       The parameter `manualfeed` is a boolean (*bool*) that controls whether paper is to be fed manually TRUE or from the paper tray FALSE.

**Standard value**   FALSE

### **jobname**

**Syntax**           jobname *string*

**Definition**       The parameter `jobname` is a string that specifies the name (*string*) of the current job. If a PostScript program defines `jobname`, status messages generated during the remainder of the current job will include a job field that reports the text of *string*. The string should not contain the characters `;` or `]`, since they would disrupt the syntax of status messages.

**Standard value**   null

### **printererror**

**Syntax**           *status tries* printererror

**Definition** The procedure `printererror` is called during execution of `showpage` or `copypage` if the printer mechanism reports an error, such as no paper, a paper jam, or the cover open. The variable `status` is an integer that encodes details of the error condition; it is device-dependent and is not documented here. The variable `tries` is the number of times `printererror` has previously been called during the same `showpage` or `copypage`. If `printererror` returns, the printing operation is retried; if `printererror` aborts (by executing `stop`), the printing operation is abandoned.

The standard `printererror` procedure interprets `status` and generates a `PrinterError` status message. The procedure then returns, thus allowing printer errors to be retried indefinitely.

### **product**

**Syntax** `product string`

**Definition** The parameter `product` is a string object (*string*), which is the name of the laser printer product, *LaserWriter IINTX*. The rare program that needs to know what type of printer it is running on should check this string.

**Standard value** `LaserWriter IINTX`

### **revision**

**Syntax** `revision int`

**Definition** The parameter `revision` is an integer (*int*) that designates the current revision level of the machine-dependent portion of the PostScript interpreter.

**Standard value** `3`

### **appletalktype**

**Syntax** `appletalktype string`

**Definition** The parameter `appletalktype` is a string object (*string*) that defines the type portion of the printer's AppleTalk name according to the Name Binding Protocol. The default value is `LaserWriter` because the Macintosh Print Manager assumes that all PostScript printers are of type `LaserWriter`.

**Standard value** `LaserWriter`

**Errors** `stackoverflow`

---

---

## PostScript language changes

Several additions have been made to the standard PostScript language. These additions are compatible with later versions and do not affect the function of any existing PostScript programs.

The changes are included in the LaserWriter IINTX and in other PostScript printers; they will be documented in a future edition of the *PostScript Language Reference Manual*.

In general, PostScript programs that are intended to be compatible with all PostScript printers should not make use of the new features. However, a PostScript program can determine whether or not the new features are present, and invoke them conditionally. The descriptions presented next suggest how to determine whether a particular PostScript feature is present or absent.

---

### Packed arrays

PostScript procedures are represented as executable arrays that, until now, have been stored in the same fashion as literal data arrays. While offering maximum flexibility, this representation wastes space (8 bytes per element). Large PostScript programs, such as the built-in server program and downloaded preambles, consume considerable amounts of VM.

Since most programs do not require the ability to be treated as data but only require the ability to be executed, a more compact representation has been introduced: the packed array. Programs represented as packed arrays are typically 50% to 75% smaller than the same programs represented as ordinary arrays.

A packed-array object has a type different from an ordinary array object ('packedarraytype' versus 'arraytype'); but in most respects, a packed-array object behaves the same as an ordinary array. Therefore, you can execute a packed array; you can extract elements (using `get`) or subarrays (using `getinterval`); you can enumerate a packed array (using `forall`); and so forth. Individual elements extracted from a packed array are ordinary PostScript objects; a subarray of a packed array is also a packed array.

The differences between packed arrays and ordinary arrays are as follows:

- Packed arrays are always read-only: you can't use `put`, `putinterval`, or other operators to store into one.
- Packed arrays are created differently from ordinary arrays, as described later.
- Accessing arbitrary elements of a packed array can be quite slow; however, accessing the elements sequentially (as is done by the PostScript interpreter and by the `forall` operator) is approximately as efficient as accessing an ordinary array.
- The `copy` operator cannot copy into a packed array since it is read-only; however, `copy` can copy the value of a packed array to an ordinary array of at least the packed array's length.

There are two ways in which packed arrays come into existence. The first and more common way is for the PostScript input scanner to create packed arrays automatically for all executable arrays that it reads. Whenever the scanner encounters a `{ while` reading a file or string, it accumulates all tokens up to the matching `}` and turns them into a packed array instead of an ordinary array.

The choice of array type is controlled by a mode setting, manipulated by the new operators `setpacking` and `currentpacking`, described later. If the array packing mode is `TRUE`, PostScript procedures encountered subsequently by the scanner are created as packed arrays; if the mode is `FALSE`, procedures are created as ordinary arrays. The default value is `FALSE` for compatibility with existing programs.

The other way to create a packed array is to build it explicitly by invoking the `packedarray` operator with a list of operands to be incorporated into a new packed array.

---

## Immediately evaluated names

The language syntax has been extended to include a new kind of name token called the *immediately evaluated name*. After encountering the token `//name` (a name preceded by two slashes with no intervening spaces), the scanner immediately looks up the name in the context of the current dictionary stack and substitutes the corresponding value for the name. If the name is not found, an undefined error occurs.

The substitution occurs *immediately*, regardless of whether or not the token appears inside an executable array delimited by braces, { and }. This process is a substitution and not an execution; that is, the name's value is not executed, but rather is substituted for the name itself, just as if the `load` operator had been applied to the name. This process is related to the action performed by the `bind` operator (see the *PostScript Language Reference Manual*); but whereas `bind` performs substitution only for names whose values are operators, the substitution process replaces each occurrence of the `//name` syntax by the value associated with *name*, regardless of the value's type. The following examples illustrate this:

```
/a 3 def
/b {(test) print} def
//a→3
//b →{(test) print}
{/a //b a /b} →{3 {(test) print} a /b}
```

The purpose of using immediately evaluated names is similar to the purpose of using the `bind` operator: using either causes names in procedures to become tightly bound to their values. However, be careful: indiscriminate use of immediately evaluated names may change the semantics of a program. In particular, after encountering a procedure object *directly*, the interpreter simply pushes the object on the operand stack; but when the interpreter encounters a procedure object *indirectly* (by looking up an executable name), the interpreter executes the procedure. (For more information, see the *PostScript Language Reference Manual*.) Therefore, execution of the program fragments

```
{... b ...}
{... //b ...}
```

may have different effects if the value of the name *b* is a procedure.

The immediately evaluated name facility is present in all versions of the PostScript interpreter since version 25.0 (as reported by the `version` operator). Earlier versions of the interpreter will scan `//name` as two distinct tokens: `/`, as a literal name with no text at all, and `/name`, as a literal name whose text is *name*.

---

## Font-cache operation

The operation of the font cache has changed somewhat. Formerly, there was a single limit on the number of bytes occupied by a character in the cache; a character larger than that would not be cached. Now there are two cache thresholds, a lower one and an upper one. If a character is larger than the upper threshold (as determined by the bounding box specified to `setcachedevice`), the character will not be cached; if it is smaller, it will be. If it is cached and is larger than the lower threshold, it will be *compressed*; if not, it will be stored as a full-pixel array.

As described later, the new operators `setcacheparams` and `currentcacheparams` manipulate the two thresholds. The old operators, principally `cachestatus` and `setcachelimit`, remain valid. It is a rare PostScript program that needs to deal with either set of these operators.

Compressed characters consume much less space in the font cache than full-pixel arrays (by factors of up to 40), but require more computation to reconstitute when you need them. Reconstituting a compressed character is still substantially faster than re-executing the original character description. In systems that print at 300 pixels per inch or less (including the LaserWriter IINTX), the default lower threshold is set so that characters up to approximately 20 points are stored as full-pixel arrays, while larger ones are stored in compressed form. Doing this causes ordinary body text to be cached using the time-efficient full-pixel array representation, but also causes large characters to be cached using the space-efficient compressed representation.

---

## Device-resolution images

A larger class of sampled images is now transferred directly from a binary source image to the raster output device, rather than by using the more general technique of sampling and halftoning. The conditions for the `image` operator's fast case are now as follows.

- The image is 1 bit per sample.
- The combination of the image matrix and the current transformation matrix is such that one unit in image space corresponds to one unit in device space. (Therefore, the image and device resolutions are the same.)
- The image-coordinate system's  $x$ -axis and  $y$ -axis are either parallel or perpendicular to the corresponding axis of device space. (This condition expands the fast case to include rotations of 0, 90, 180, and 270 degrees, and their  $x$ -axis and  $y$ -axis reflections for a total of 8 different image orientations, instead of the 2 orientations that were previously allowed.)

If an image that meets these conditions on the LaserWriter IINTX is printed on an earlier PostScript printer, the printer may treat it as a general image. In this case, the earlier printer may process the image more slowly than the LaserWriter IINTX, but the results will still be correct, preserving the device independence of PostScript page descriptions.

---

## New Operators

Here are the descriptions of the new operators.

### **setpacking**

#### **Syntax**

*bool* setpacking

#### **Definition**

The operator `setpacking` sets the array-packing mode to the specified boolean value (*bool*). Doing this determines the type of executable arrays subsequently created by the PostScript scanner. The value `TRUE` selects packed arrays; `FALSE` selects ordinary arrays.

The packing mode affects only the creation of procedures by the scanner when it encounters program text bracketed by `{` and `}` either during interpretation of an executable file or string object, or during execution of the `token` operator. The packing mode does not affect the creation of literal arrays by the `[` and `]` operators or by the `array` operator.

The array-packing mode setting persists until it is overridden by another execution of `setpacking` or until it is undone by a `restore`.

Here is an example:

```
systemdict /setpacking known
  {/savepacking currentpacking def
   true setpacking
  } if
```

... arbitrary procedure definitions ...

```
systemdict /setpacking known {savepacking setpacking} if
```

If the packed-array facility is available, the procedures represented by *arbitrary procedure definitions* are defined as packed arrays; if the facility is not available, they are defined as ordinary arrays. The example just given carefully preserves the array-packing mode in effect before its execution.

**Errors** stackunderflow, typecheck

### **currentpacking**

**Syntax** currentpacking *bool*

**Definition** The operator `currentpacking` returns the array-packing mode (*bool*) currently in effect. See `setpacking`, described earlier, for a definition of the array-packing boolean values.

**Standard value** FALSE

**Errors** stackoverflow

### **packedarray**

**Syntax** *any<sub>0</sub> ... any<sub>n-1</sub> n* packedarray *packedarray*

**Definition** The operator `packedarray` creates a packed array object, *packedarray*, of length *n* that contains the objects *any<sub>0</sub>* through *any<sub>n-1</sub>* as elements. The operator `packedarray` first removes the nonnegative integer *n* from the operand stack. It then removes *n* objects from the operand stack, creates a packed array that contains those objects as elements, and finally pushes the resulting packed array object on the operand stack.

The resulting object has a type of 'packedarraytype', a literal attribute, and read-only access. In all other respects, the behavior of the object is identical to that of an ordinary array object.

**Standard value** FALSE

**Errors** rangecheck, stackunderflow, typecheck, VMerror

## setcacheparams

**Syntax** *mark lower upper setcacheparams*

**Definition** The operator `setcacheparams` sets cache parameters as specified by the integer objects above the topmost mark (*mark*) on the stack, and then removes all operands and the mark object as if `cleartomark` had been used.

The number of cache parameters varies. If more operands are supplied to `setcacheparams` than are needed, it uses the topmost ones and ignores the remainder; if fewer are supplied than are needed, `setcacheparams` implicitly inserts default values between *mark* and the first supplied operand.

The operand *upper* specifies the maximum number of bytes that may be occupied by the pixel array of a single cached character, as determined from the information presented by the `setcachedevice` operator. This operand is the same parameter as is set by `setcachelimit`; see the description of that operator in the *PostScript Language Reference Manual*.

The operand *lower* specifies the threshold at which characters are stored in compressed form, rather than as full-pixel arrays. If a character's pixel array requires more than *lower* bytes to represent, the array will be compressed in the cache and reconstituted from the compressed representation each time it is needed.

Setting *lower* to a value of 0 forces all characters to be compressed, which permits more characters to be stored in the cache but increases the work required to print them. Setting *lower* to a value greater than or equal to *upper* disables compression altogether.

**Errors** `rangecheck`, `unmatchedmark`

## currentcacheparams

**Syntax** `currentcacheparams mark lower upper`

**Definition** The operator `currentcacheparams` pushes a mark object (*mark*) followed by the current cache parameters *lower* and *upper* on the operand stack. The number of cache parameters returned varies. (See `setcacheparams` for a description.)

**Standard value** `mark 1250 12500`

**Errors** `stackoverflow`

---

## Showpage and copypage

The correct use of `showpage` and `copypage` is a matter that requires some clarification. Inappropriate use of `copypage` can result in significant degradation of performance in new PostScript printers.

The operator `showpage` is the normal operator for causing pages to be output. It has three effects: it prints the current page, erases the current page, and reinitializes the graphics state.

The operator `copypage` is a somewhat more specialized operator that simply prints the current page but does not erase it or reset the graphics state. The main intended use of `copypage` is to permit the adding of new marks to an existing page, such as when building up a page incrementally.

The operator `showpage` is logically equivalent to the sequence `copypage erasepage initgraphics`

However, use of `copypage` for printing pages can degrade page throughput significantly. One reason for this is that `showpage` performs the printing and the erasing in parallel, whereas the `copypage erasepage` method performs them serially.

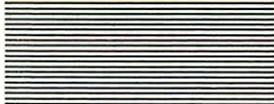
You should not use `copypage` to defeat the automatic `initgraphics` of `showpage`; that is, to print and erase the current page but leave the graphics state unchanged, you should *not* use

```
copypage erasepage
```

Instead, you should use

```
gsave showpage grestore
```

- ❖ *Note:* The correct way to print multiple copies of a page is to associate the desired number of copies with the name `#copies` prior to invoking `showpage`, as discussed in the section on `showpage` in the *PostScript Language Reference Manual*. The `#copies` convention now applies uniformly to both `showpage` and `copypage`, whereas formerly it applied only to `showpage`.



## Chapter 5



# LaserWriter Functional Overview

The LaserWriter family of printers print by using a combination of laser light, electrical charges, and a black plastic powder called *toner*.

Before the printing process, QuickDraw creates a bit-mapped image of the page in the Macintosh. The Printing Manager uses the LaserWriter driver to convert that bit-mapped image into a PostScript file that the LaserWriter controller can process. The PostScript file is sent from the Macintosh to the LaserWriter over the AppleTalk network channel. The PostScript program data is sent until an entire page image is reconstructed in the LaserWriter controller's RAM by the LaserWriter PostScript interpreter in the controller's ROM. The LaserWriter then activates a laser, and transfers the bit map of the page to a negatively charged photosensitive drum. Laser beam pulses that match each bit of the bit-mapped page image are reflected through a series of mirrors onto the rotating drum. The laser light reduces the negative charge in the areas that form the printed image (black), while leaving the surrounding areas with a greater negative charge (white). The negatively charged toner is repelled from the areas on the drum that have the greater negative charge, and is attracted to the less negatively charged dots on the drum that make up the page image.

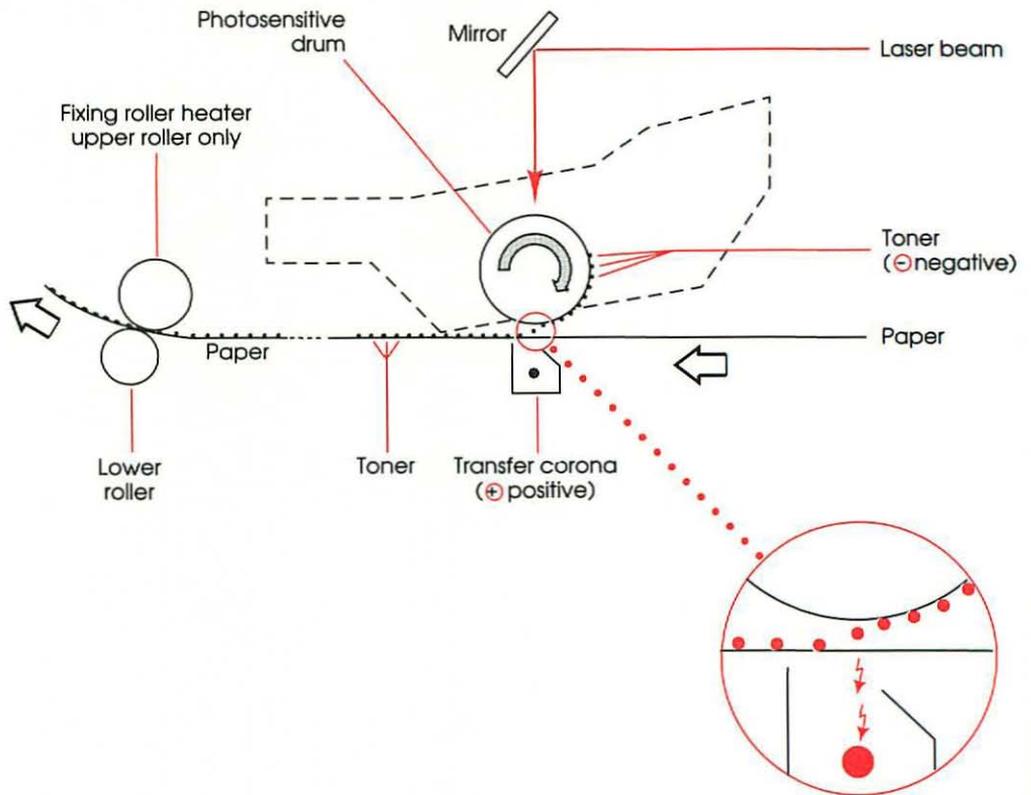
The page image is transferred from the drum to the paper according to the same principles of attraction that cause the toner to adhere to the dots on the drum. The paper passes between the drum and a positively charged wire called the *transfer corona wire*. The toner is attracted to the positively charged transfer corona wire and adheres to the paper passing under the drum. The paper then passes between two fuser rollers, which fuse the dots of toner onto the paper with a combination of heat and pressure. A simplified diagram of the image-transfer process is shown in Figure 5-1.

---

---

## The LaserWriter IINT/IINTX controllers

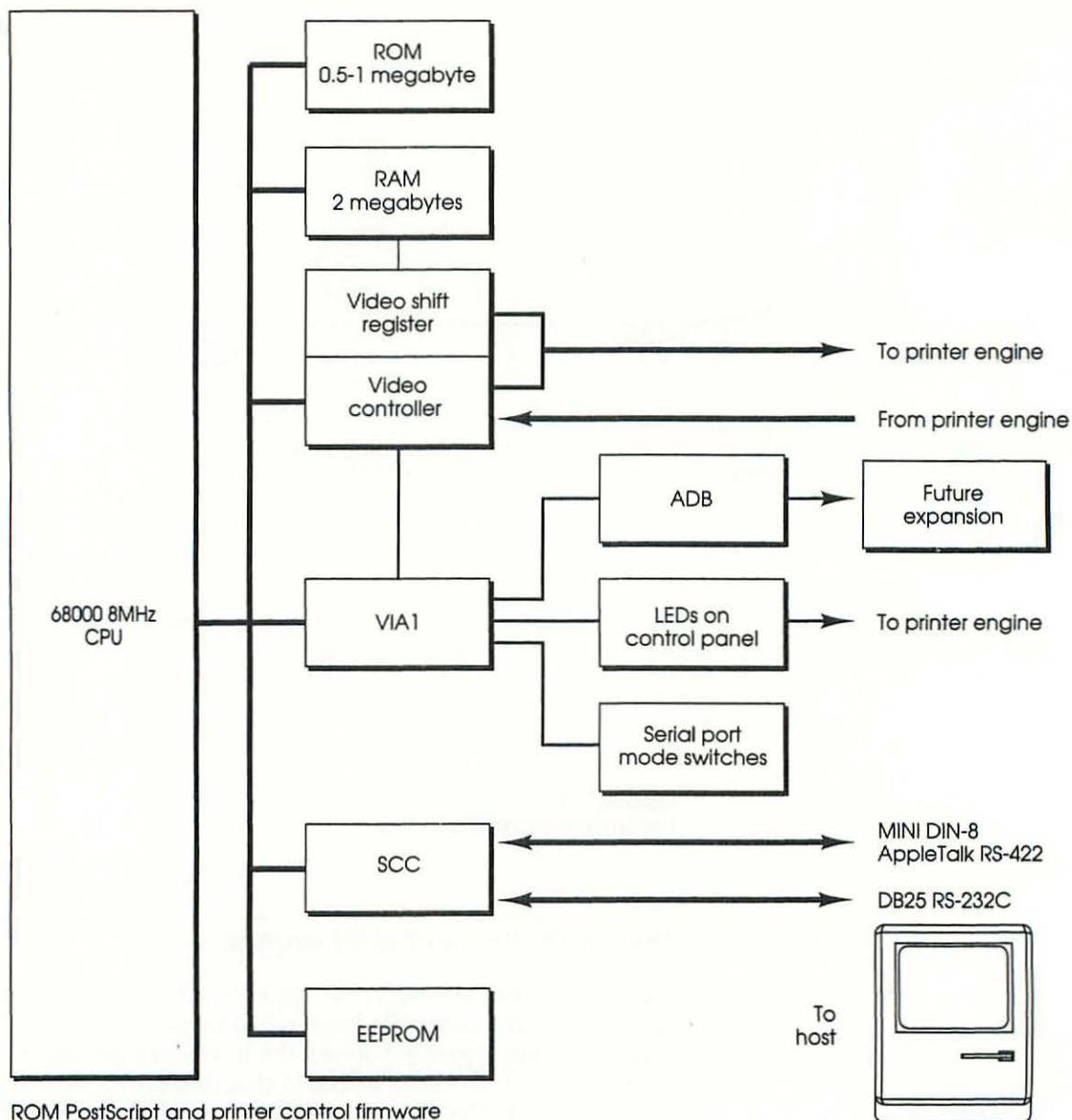
The LaserWriter IINT and LaserWriter IINTX controllers are electronic circuit boards designed to interface with the printer engine and Apple computers over the AppleTalk or RS-232 interfaces. The circuit boards and interface connectors are configured as modules that slide into a card slot in the base of the print engine. The modules automatically connect to the print engine's control electronics through a 32-pin, right-angle connector. Standard AppleTalk or serial interface cables attach the controller module to the host computer.



**Figure 5-1**  
The Image transfer process

## The controller and print engine

Figure 5-2 shows the functional modules on the LaserWriter and LaserWriter IINT controller boards that interact to cause the print engine to print. Figure 5-3 shows the functional modules on the LaserWriter IINTX controller board that interact to cause the print engine to print. The functional modules of each controller are described following a general explanation of the video interface.



ROM PostScript and printer control firmware

RAM virtual Memory for page imaging and font cache

ADB (Apple Desktop Bus) Serial, low-speed bus for future expansion

VIA (Versatile Interface Adapter) controls video and powers LEDs

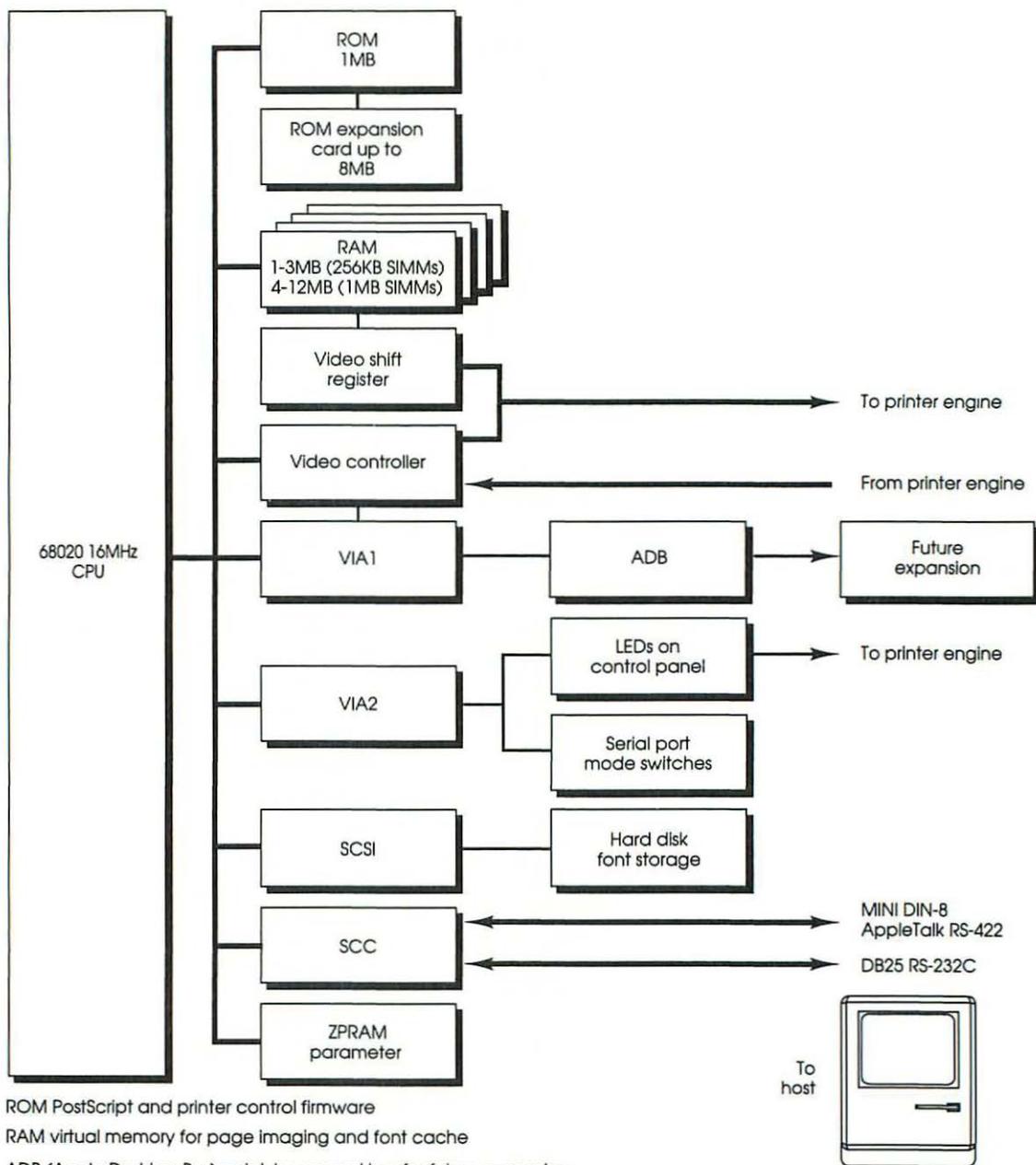
SCC (Serial Communication Controller) chan-A: AppleTalk, Async RS-422 chan-B: Async RS-232C

EEPROM stores serial port configuration settings

Video control provides bit image data to the printer; returns printer status to the controller

**Figure 5-2**

LaserWriter and LaserWriter II NT controller functional block diagram



ROM PostScript and printer control firmware

RAM virtual memory for page imaging and font cache

ADB (Apple Desktop Bus) serial, low-speed bus for future expansion

VIA (Versatile Interface Adapter) controls video and powers LEDs

SCC (Serial Communication Controller) chan-A: AppleTalk, Async RS-422 chan-B: Async RS-232C

ZPRAM stores serial port configuration settings

Video control provides bit image data to the printer; returns printer status to the controller

**Figure 5-3**  
LaserWriter II/IIXTX controller functional block diagram

All LaserWriter controllers that utilize a PostScript interpreter in firmware on the controller board interact with the print engine similarly. The controller receives PostScript programs or ASCII control data (when in emulation mode) from the host over an AppleTalk or RS-232 serial connection. All of the serial data that the controller receives through the serial port is under the control of the Serial Communications Controller (SCC). When the internal buffer in the SCC is filled, the SCC generates an interrupt to the controller CPU. This interrupt causes the CPU to take the data out of the SCC internal buffer, and to place it in a larger AppleTalk buffer in controller RAM. After the program data is in the AppleTalk buffer, the CPU can begin to execute the PostScript program. The CPU then creates the page image in the controller RAM according to the commands in the PostScript program. This process continues until the entire bit-mapped image of the page is created in the frame buffer. When the page image is constructed, the CPU takes the image data out of RAM, and puts it into a first-in-first-out (FIFO) buffer. When the controller Versatile Interface Adapter (VIA) is notified that the printer is ready, the Video Control circuit loads the scan line data, 1 byte at a time, into the Shift Register and sends it to the print engine.

The print engine's video interface receives the data as digital bits in the form of serial signals that correspond to either black or white dots of the image. The print engine modulates a laser beam to create on a rotating drum an image that is based on the serial signals received over the video interface from the controller.

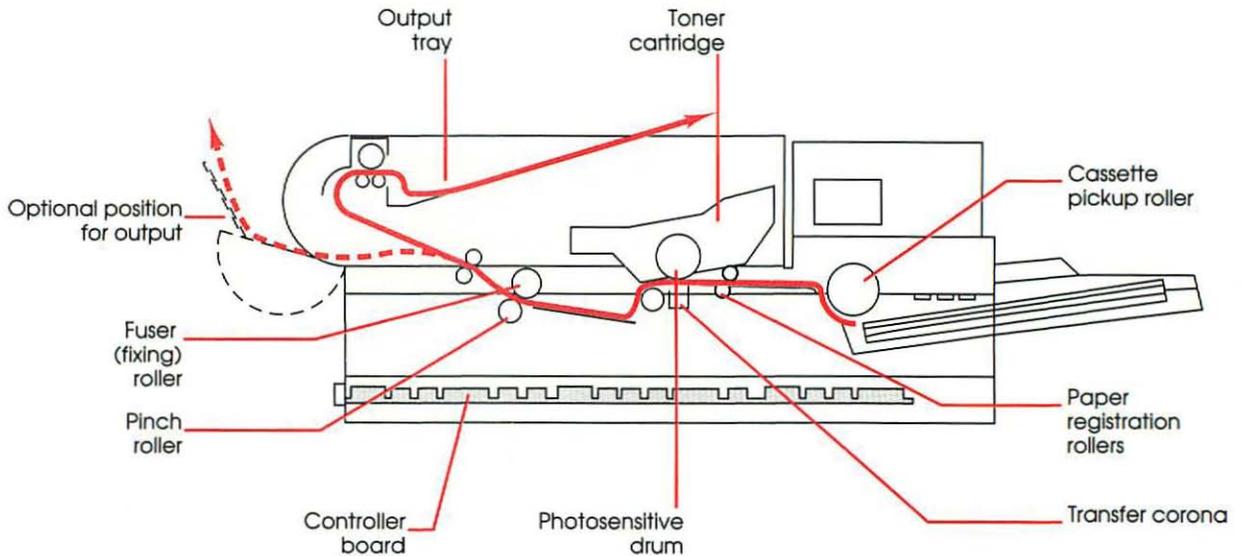
The drum rotates at a constant speed. In the time the laser beam uses to cross the drum (in order to write one scan line), the drum's surface shifts downward  $\frac{1}{300}$  of an inch. A beam-position sensor in the print engine notifies the controller that the laser beam has completed a scan line and that it is at home position, ready for the next scan line.

### **Video interface**

The handshaking sequence that takes place over the video interface between the controller VIA and the laser print engine for the page-printing process is as follows:

1. The print engine asserts a Ready signal. This signal indicates that the print engine is ready to print or to continue printing.

2. The controller asserts a Print signal. This signal tells the print engine to take a sheet of paper out of the tray (or other selected source), and to position it for printing. Figure 5-4 shows the paper path for face-down paper delivery during the printing process, and the location of the mechanical components discussed in this section.



**Figure 5-4**  
The paper path and the mechanical-component location (In the print engine)

3. The print engine asserts a Vertical Sync Request signal. This signal indicates that the paper-feed rollers have positioned the leading edge of the sheet of paper against the paper-detect sensor, which indicates that the paper is positioned at the top of page, and that the controller can send video data by starting from the top of the page in memory.
4. The controller asserts a Vertical Sync signal. This signal tells the print engine that the controller is sending video data. The signal also starts paper motion.
5. The print engine asserts a Beam Detect signal. This signal indicates that the laser beam is at the starting point of the horizontal axis on the drum and is ready for scan-line data.

6. The controller sends a video signal. This signal is the data image signal for printing. The print engine prints black for TRUE video signal and white for a FALSE video signal.

The Beam Detect and video signals are transmitted between the print engine and controller until the page image is complete or an error condition occurs. When the print engine indicates an error (possibly a paper jam) to the controller, the controller sets the LED indicators (in this case, the paper jam LED) and waits for the operator to correct the condition. After the error is corrected and the print engine indicates that it is ready, the print job restarts from the page that was in progress at the time of the paper jam.

---

## LaserWriter and LaserWriter IINT functional modules

The information that follows describes the LaserWriter functional block diagram and the additional logic that supports the functional modules.

- **CPU:** The CPU on the LaserWriter and LaserWriter IINT is a MC68000 32-bit microprocessor that runs at 7.455 MHz. The CPU provides all the necessary central processing functions for the LaserWriter or LaserWriter IINT controller board.
- **ROM:** On the controller board, the LaserWriter has 0.5 MB of ROM, and the LaserWriter Plus and LaserWriter IINT have 1.0 MB of ROM. The ROM firmware contains printer-control routines, Adobe Systems PostScript interpreter, printer-diagnostic routines, printer-emulator routines, AppleTalk routines, and error-reporting routines.
- **DRAM:** The LaserWriter contains 1.5 MB and the LaserWriter Plus and LaserWriter IINT contain 2 MB of dynamic random access memory (DRAM). The DRAM is used as an AppleTalk data buffer, a font-cache buffer for caching character bit maps, a frame buffer for constructing the bit-mapped page image, and a display-list buffer for storing compiled PostScript instructions.
- **VIA:** The Versatile Interface Adapter (VIA) provides parallel I/O for the handshaking signals between the controller and print engine.

- **FIFO/video control logic:** In LaserWriter IINT, the FIFO memory is used as a scan-line buffer for printing. Video data in RAM is loaded into the FIFO by the CPU on a scan-line-by-scan-line basis, and unloaded into the Shift Register by the video control logic.
- **Video shift register:** The video shift register converts the FIFO output into a serial bit stream that is under the control of the video control logic.
- **Video control logic:** The video control logic reads the FIFO and loads the video shift register. It also provides the shift clock to the video shift register.
- **SCC interface:** The Serial Communications Controller (SCC) controls all communication with the host over the serial bus. The SCC has two independently programmable channels. Each channel can be programmed for asynchronous, synchronous, or AppleTalk protocols.

The serial interface is connected to host computers through subminiature DB25 RS-232 connectors or 8-pin miniature DIN connectors (MINI DIN-8). See Appendix A for the serial connector pinouts.

---

## LaserWriter IINTX functional block diagram

The information given next describes the LaserWriter IINTX functional block diagram.

- **LaserWriter IINTX CPU:** The 16MHz MC68020 microprocessor provides all central-processing functions for the LaserWriter IINTX controller. The 68020 instruction set is a superset of the 68000 instruction set, and provides compatibility with programs written for the original LaserWriter, LaserWriter Plus, and LaserWriter IINT. Because the 68020 is a true 32-bit microprocessor and because it runs at 16MHz, processing instructions execute much faster on the LaserWriter IINTX than they do on the other Apple laser printer models. For more information about the capabilities of the MC68020 microprocessor, see the *MC68020 32-Bit Microprocessor User's Manual* second edition, (Prentice-Hall, 1985).
- **DRAM:** The dynamic random access memory (DRAM) on the LaserWriter IINTX controller board is designed to contain from 2 MB to 12 MB of RAM. RAM sizes of 2, 3, 4, 5, 8, 9 and 12 MB are supported.

The RAM is used as an AppleTalk data buffer, a font-cache buffer for caching character bit maps, a frame buffer for constructing the bit-mapped page image, and a display-list buffer for storing compiled PostScript instructions.

- **ROM:** LaserWriter IINTX's 1 MB of read-only memory (ROM) firmware contains printer-control routines, Adobe Systems PostScript interpreter, printer-diagnostic routines, printer-emulator routines, AppleTalk routines, LaserWriter Plus fonts, and fixed-disk support as described in "LaserWriter IINTX Font Cache and Disk System" in Chapter 2 and the Adobe Systems *PostScript Language Reference Manual*.

A connector on the controller board allows for expansion of the ROM bus. You can use this connector to install a ROM expansion board that allows you to add more ROM banks for storage of optional fonts.

- **Serial I/O (SCC):** The Serial Communications Controller (SCC) has two independent ports for serial communication. Each port can be independently programmed for asynchronous, synchronous, or AppleTalk protocols.

The serial interface is connected to a host computer through subminiature DB25 RS-232 connectors or 8-pin DIN connectors (MINI DIN-8). For the serial connector pinouts, see Appendix A.

- **SCSI:** The Small Computer System Interface (SCSI), together with extensions in LaserWriter IINTX ROM for a disk-based file system, provides for connection of a SCSI hard disk. You can use this disk as additional virtual memory for expanding display-list buffers for printing complex pages, as a user-accessible file storage facility, and as a nonvolatile storage for a cache of fonts.

The LaserWriter IINTX SCSI consists of the 53C80 SCSI controller and an external 50-pin SCSI connector (Centronics-style). The 53C80 is connected directly to the 50-pin external connector. The LaserWriter IINTX controller does not provide termination of the SCSI signals or active power for external terminators. External terminator plugs are required and another device connected to the SCSI bus (typically a hard disk) must provide power for those terminators.

- **Video interface:** The LaserWriter IINTX video interface receives digital bits in the form of serial signals that correspond to either black or white dots of the image to be printed. The printer modulates a laser beam to create an image based upon the signals received by the video interface. The LaserWriter IINTX controller sends printer-parameter commands to the printer, and monitors the status returned by the print engine. While these signals are being exchanged, the controller processes the bit-mapped image. After the image is completed, it is shifted out of the controller RAM and sent to the printer by the 8-bit-wide video shift register through the video interface.

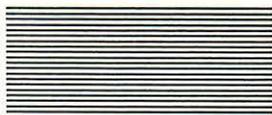
The LaserWriter IINTX controller sends video data to the print engine until the page image is complete or until a paper out, paper jam, or printer error occurs. Before and after sending page data to the printer, the controller can send a serial command and receive serial status back.

- **Video shift register and video control logic:** Two video data FIFOs share the video shift register. The video control logic Read circuit selects each FIFO alternately to present video data to the video shift register, and causes the video shift register to load the data. The Read circuit also signals VIA1 that a word has been shifted out of the FIFOs, which decrements a counter in the VIA1. The VIA1 counter is read by the MC68020 microprocessor, which compares the current word-count value with the previously read value. If enough space is available in the FIFOs, as signaled by the word-count value, one or more scan lines of data is transferred into the video data FIFO. This method of multiple-scan-line buffering releases the processor from waiting for each scan line to be shifted out to the FIFOs, so that it can take care of other I/O tasks, such as network communication.

When a Beam Detect signal is received from the printer, a bit-clock circuit that acts as a counter is resynchronized and starts the video clock for the video shift register and FIFO Read circuit to send the next scan line to the printer. This process helps to maintain image integrity by closing any gaps between the scan lines sent to the printer.

- **VIA1 and VIA2 interface:** The LaserWriter IINTX Versatile Interface Adapters (VIA1 and VIA2) control a variety of controller processes by monitoring requests between the controller and printer. VIA1 controls data transmission through the video interface to the print engine, and returns either Ready or Busy to the controller. VIA2 lights the printer status LEDs, monitors the settings of the controller communication-mode configuration switches, and reports on the status of the printer power-up self-test. For more information about the VIA, see Chapter 2 of *Inside Macintosh*, Volume III.
- **ZPRAM:** The serial-port configuration setting is stored in the battery-backed Zero Power RAM (ZPRAM). You can change the configuration information by sending PostScript control operators to the printer or by changing the mode-configuration switch settings. Whenever the mode-configuration switch settings are changed, the configuration stored in the ZPRAM is updated to match the new settings. If the switches are changed while the printer is powered off, the ZPRAM configuration data is updated when VIA2 reads the switch settings at power-on. The configuration information stored in the ZPRAM is maintained after the power has been turned off and then on.

If you believe that the controller is incorrectly configured, toggle any switch, wait 30 seconds, and then toggle the switch back to its normal position. Doing this forces the ZPRAM configuration information to match the switches and overrides any conflicting software-downloaded configuration.



## Chapter 6



# LaserWriter IIINTX Font Expansion Card Specification

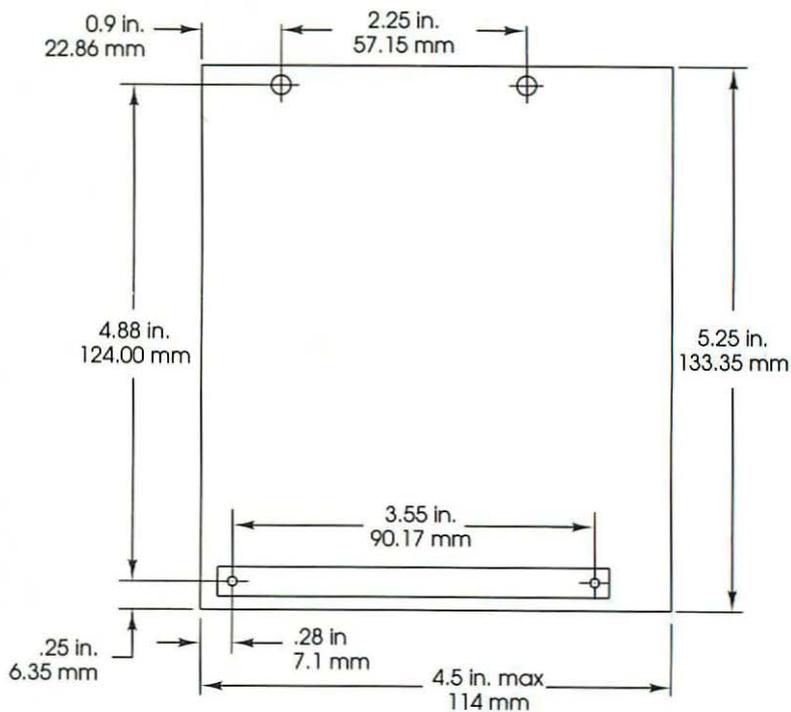
This chapter provides the specifications for hardware engineers interested in designing a font expansion card for the LaserWriter IINTX. The purpose of a LaserWriter IINTX font expansion card is to provide additional ROM storage for fonts. The fonts must be stored either as character bit maps or as character outlines, as defined in the *PostScript Language Reference Manual* and must conform to the Adobe cartridge-font definition. Contact Adobe Systems Incorporated for implementation details.

---

---

## Font expansion card dimensions

The dimensions for the font expansion card must be no larger than 5.25 inches by 4.5 inches (13.34 centimeters by 11.8-centimeters). The card is connected to the controller board and supported at one end by a 96-pin DIN connector. At the other end the card is raised above the controller board by two 0.625-in. (1.6-cm) nylon circuit board supports (standoffs). The standoffs fit into two 0.156-inch (3.96-mm) holes in the expansion card and two 0.187-inch (4.75-mm) holes in the controller board. The standoffs are not on the LaserWriter IINTX controller board and must be supplied with the font expansion card. Figure 6-1 shows the locations for the 96-pin connector and standoffs on the font expansion card.



**Figure 6-1**  
LaserWriter IINTX font expansion card dimensions

---



---

## Interface connection

The font expansion card design calls for the ROM to be arranged in parallel with the LaserWriter IINTX main ROM. The card interfaces with the LaserWriter IINTX control board through a 96-pin DIN connector. Table 6-1 presents a pin description of this connector.

**Table 6-1**Pin description of the 96-pin DIN connector to the LaserWriter II<sup>TX</sup> expansion card

Pin number	Row A	Row B	Row C
1	nc	+5V	d0
2	a2	+5V	d1
3	a3	+5V	d2
4	a4	+5V	d3
5	a5	+5V	d4
6	a6	+5V	d5
7	a7	+5V	d6
8	a8	+5V	d7
9	a9	nc	d8
10	a10	nc	d9
11	a11	nc	d10
12	a12	nc	d11
13	a13	nc	d12
14	a14	nc	d13
15	a15	nc	d14
16	a16	nc	d15
17	a17	dbrom0n	d16
18	a18	dbrom1n	d17
19	a19	dbrom2n	d18
20	a20	dbrom3n	d19
21	a21	nc	d20
22	a22	nc	d21
23	a23	nc	d22
24	a24	nc	d23
25	a25	gnd	d24
26	a26	gnd	d25
27	a27	gnd	d26
28	a28	gnd	d27
29	a29	gnd	d28
30	a30	gnd	d29
31	a31	gnd	d30
32	a32	gnd	d31

---

---

## Power consumption

The font expansion card cannot draw more than 600 milliamps of power from the LaserWriter IINTX controller board. The address and control lines must present a dynamic load of no more than 100pF, and a static load of no more than one 74ALS244 input. The data lines must present a load of no more than 50pF to the LaserWriter IINTX main-board ROM when tristate, and must be able to drive a dynamic load of 100pF and a static load of one 74F244s input.

---

---

## Timing

The allowable access time from the assertion of the expansion card chip select signal to stable data out signal must be no more than 150 ns for HTC data buffers and 175 ns for ALS data buffers. However, the address to data-access time is 223 ns.

The expansion card requires 1 more wait clock than the LaserWriter IINTX main ROM. The LaserWriter IINTX logic inserts the necessary wait clocks into the MC68020 CPU memory cycle to allow the expansion card to operate properly. Because fonts in the expansion card consist of data rather than instructions and can be fetched quickly, the additional clock cycle does not present any degradation in performance.

---

---

## Memory

The expansion card ROM memory address space is \$5100 0000 to \$5FFF FFFF. The font expansion card ROM must be accessed as long words. The LaserWriter IINTX controller design requires that a maximum of 200 ns CMOS ROM or EPROMs be used on the expansion card. The LaserWriter IINTX controller can support expansion card configurations with 1 MB to 8 MB of CMOS ROMs or EPROMs. Possible expansion card configurations of 4 MB or 8 MB are possible with surface-mounted ROMs.

The LaserWriter IINTX control board logic provides bank select signals on the 96-pin DIN connector for three banks of 0.5 MB ROM, and 1 data-buffer select signal for any expansion card access. These select signals are dbrom0n, dbrom1n, dbrom2n, and dbrom3n. (See Table 6-1 for the corresponding pin assignments).

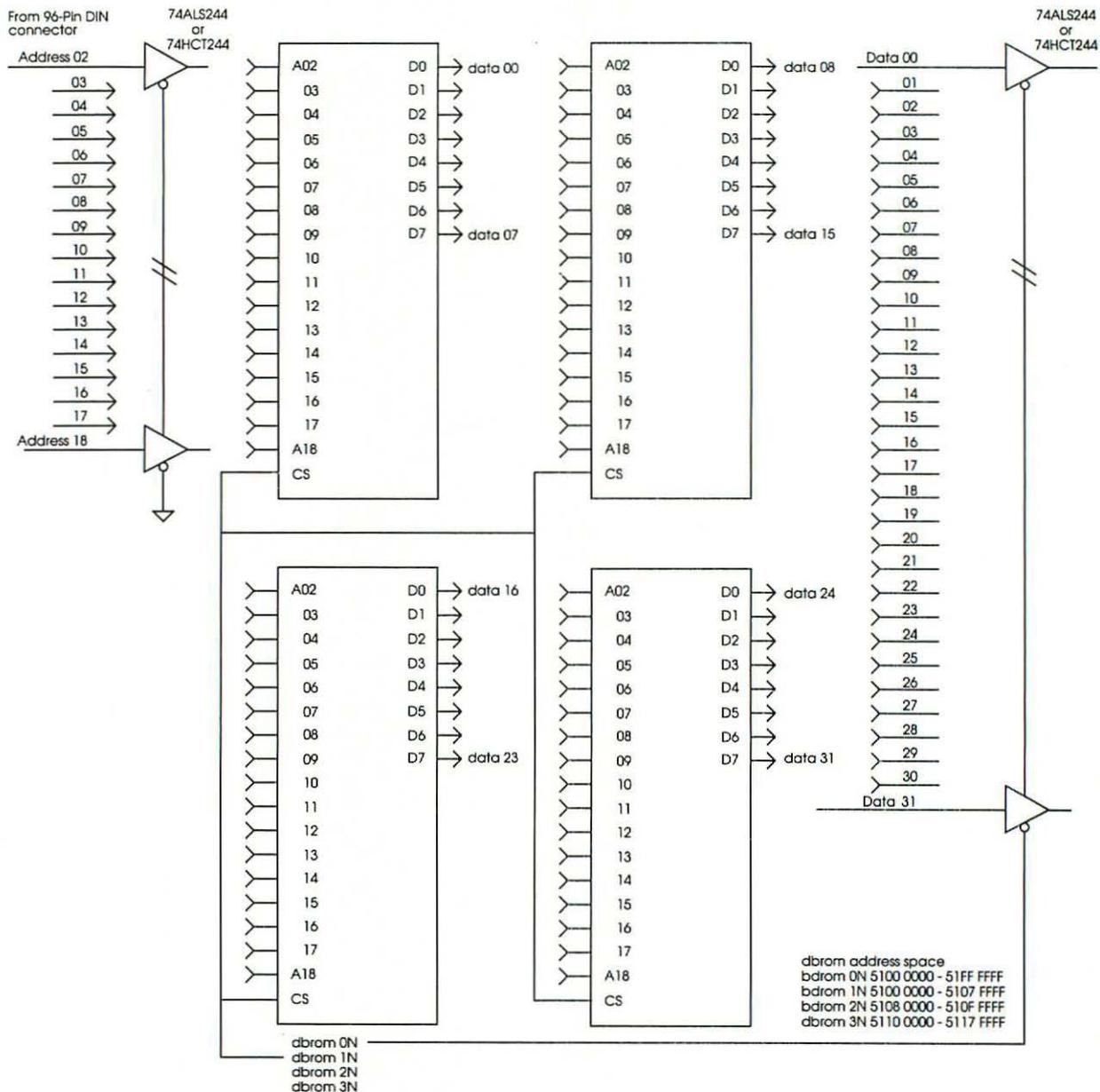
If you plan to design an expansion card with more than three 0.5 MB banks of ROM, you must provide a decoder on the expansion card to ensure only 1 bank is turned on when the expansion card is selected, and that no banks are turned on when no banks are selected. You should include the delay of this decoder in the timing calculations. The expansion card must not drive the bus when not selected. Figure 6-2 shows a simplified circuit design diagram for the font expansion card.

---

---

## Environment

The ROM address and data buffers must be CMOS devices to reduce the temperature and power consumption. Expansion card construction should be four layers to keep the Radio Frequency Interference (RFI) under control, and should provide adequate bypass capacitors for any expansion card devices. All address, data, and control lines should be as short as possible to reduce any noise on the transmission lines. If necessary, a 47-ohm resistor may be put in series with each of the expansion card ROM outputs and the 96-pin DIN connector.



**Figure 6-2**

A simplified circuit design diagram for the font expansion card



# Appendixes



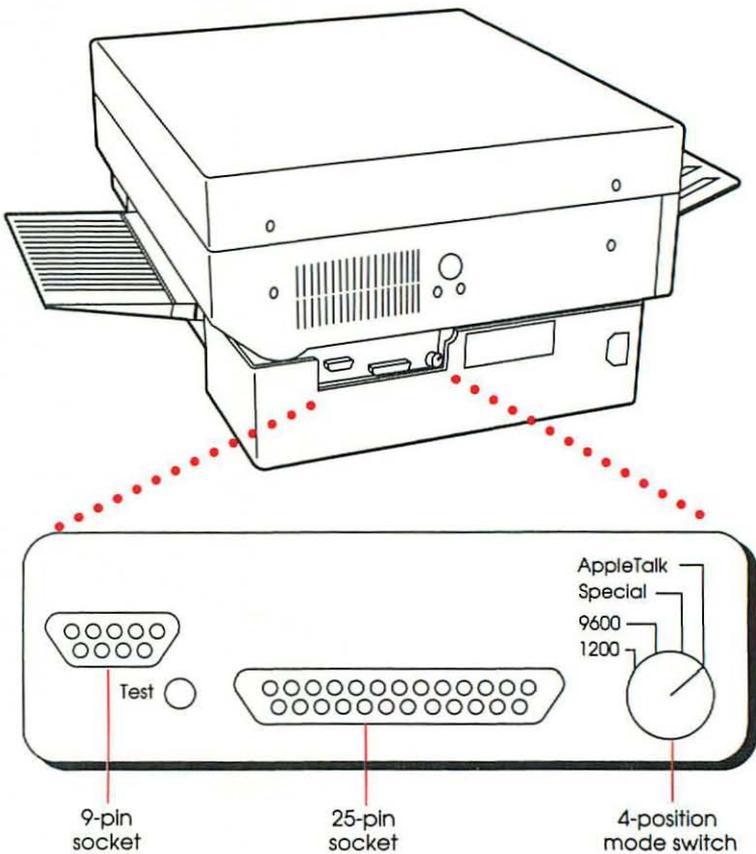
## Appendix A



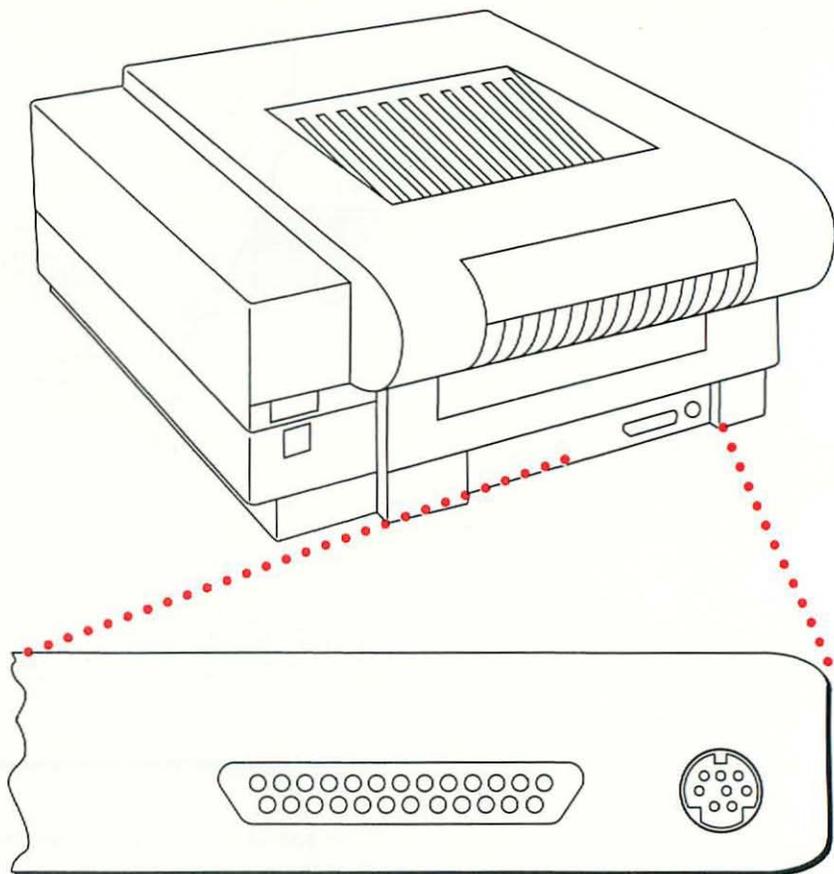
# Serial-Data Communication

The LaserWriter has two serial channels. On the original LaserWriter and LaserWriter Plus, one serial channel is wired to a DB-9 (RS-422) connector and the other to a DB-25 (RS-232C) connector, as shown in Figure A-1. On the LaserWriter IINTX and LaserWriter IINT controllers, one channel is wired to a MINI DIN-8 connector and the other to a DB 25 (RS-232C) connector, which is identical to the RS-232C connector on the original LaserWriter and LaserWriter Plus, as shown in Figure A-2. You can use either channel for conventional asynchronous serial communication. The MINI DIN-8 connector on the LaserWriter IINTX controller, and the 9-pin connector on the original LaserWriter and LaserWriter Plus are also used with the mode switch for connecting to AppleTalk, but serial communication and AppleTalk communication are incompatible and will never occur at the same time.

When the LaserWriter is in any of the serial input/output modes, it uses one of the two channels to send and receive serial data encoded in ASCII. Certain character codes serve special purposes, such as Command-D to mark the end of a file. At the beginning of a job, both channels are enabled with independent data-transmission rates and parity. The first channel to receive a character is chosen to execute the job. The other channel is not disabled; if characters start to arrive on it, they are buffered and that channel is selected when the current job is finished. When the EOF is received and the program terminates, the LaserWriter sends an EOF, ends the job, and, if possible, starts a new one.



**Figure A-1**  
The original LaserWriter and LaserWriter Plus serial connectors



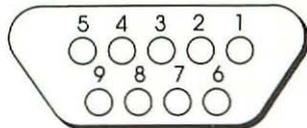
**Figure A-2**  
The LaserWriter II NT and LaserWriter II NTX serial connectors

---

---

## Using the RS-422 port

The original LaserWriter and LaserWriter Plus pinout descriptions for the 9-pin serial RS-422 port are shown in Table A-1. Figure A-3 shows the physical pin locations on the original LaserWriter and LaserWriter Plus 9-pin connector.



**Figure A-3**  
The 9-pin connector of the LaserWriter

**Table A-1**  
9-pin RS-422 port pin assignments

Pin	Signal	Description
1,3	SG	Signal ground
4	TxD+	Transmit data +
5	TxD-	Transmit data -
8	RxD+	Receive data +
9	RxD-	Receive data -

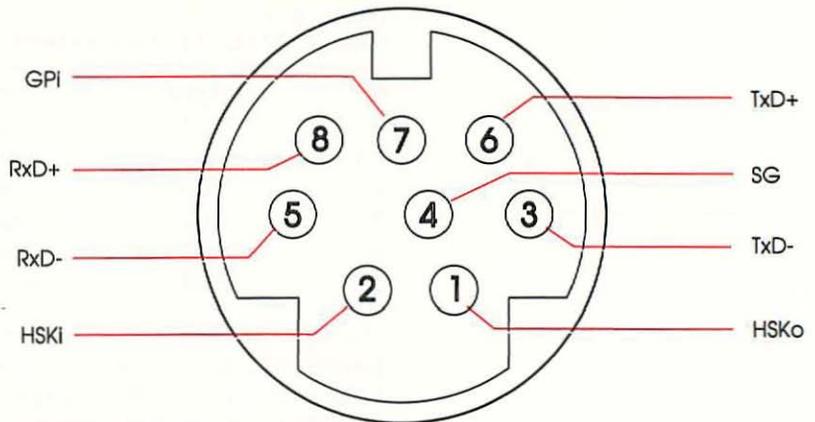
The original LaserWriter and LaserWriter Plus RS-422 pin assignments in Table A-1 are compatible with the Macintosh 128K, Macintosh 512K, and Macintosh 512K enhanced. The LaserWriter IINTX MINI DIN-8 pin assignments in Table A-2 are compatible with the MINI DIN-8 connector on the Macintosh Plus, Macintosh SE, and Macintosh II. You can connect any one of these Macintosh computers directly to a LaserWriter by using the appropriate Apple Personal Modem cable. This type of serial connection allows communication on a one-to-one interactive basis between a Macintosh using MacTerminal and the LaserWriter. See "Working Interactively" in Chapter 3 for a step-by-step explanation of how to do this.

The LaserWriter IINT and LaserWriter IINTX pinout descriptions for the MINI DIN-8 serial RS-422 port are shown in Table A-2. Figure A-4 shows the physical pin locations on the LaserWriter IINT and LaserWriter IINTX MINI DIN-8 connector.

**Table A-2**  
LaserWriter IINT and LaserWriter IINTX MINI DIN-8 RS-422 port pin assignments

Pin	Signal	Description	Comments
1	HSKo	Handshake output	Connected to the SCC Data Terminal Ready (DTR)
2	HSKi	Handshake input	Connected to the SCC Transmit/Receive Clock (TRxC)
3	TxD-	Transmit Data -	Connected to the SCC Transmit Data (TxD)
4	SG	Signal ground	Connected to logic and chassis ground
5	RxD-	Receive data -	Connected to the SCC Receive Data (RxD)
6	TxD+	Transmit data +	Connected to the SCC Transmit Data (TxD)
8	RxD+	Receive data +	Connected to the SCC Receive Data (RxD)

*Note:* Pin number 1 is not connected to LaserWriter IINT.



**Figure A-4**  
LaserWriter IInt and LaserWriter IIntX MINI DIN-8 connector

---



---

## Using the RS-232C port

The pin assignments for the LaserWriter, LaserWriter Plus, LaserWriter IInt, and LaserWriter IIntX 25-pin RS-232C port are shown in Table A-3.

**Table A-3**  
LaserWriter RS-232C port pin assignments

Pin	Signal	Description
1	Shield	Protective ground (RFI/ESD shield of cable)
2	TxD	Transmit Data
3	RxD	Receive Data
4	RTS	Request To Send (message ready)
5	CTS	Clear To Send (modem operational and remote connected)
6	DSR	Data Set Ready (power on modem)
7	GND	Signal ground
8	DCD	Data Carrier Detect (received tone from remote modem)
20	DTR	Data Terminal Ready (power on terminal [LaserWriter IIntX])
22	RI	Ring Indicator (telephone line is ringing)

---

---

## Cable configuration

Technically, the DB 25 RS232C connector on all LaserWriters is configured as **Data Terminal Equipment (DTE)**, which means that you can connect it directly to a modem or to a host computer that is configured as **Data Communication Equipment (DCE)** without any signal reversals (a straight-through cable). Connecting to another DTE device, such as a Macintosh or IBM-compatible computer, requires interposing a null modem. Making a null-modem cable simply involves connecting the Transmit Data (TxD) of each connector to the Receive Data signals (RxD) of the other. The following are details on the cable specifications for connecting a Macintosh 512K, Macintosh Plus, Macintosh SE, and Macintosh II to the LaserWriter.

Cable pin assignments for connecting the Macintosh 128K, 512K, and 512K enhanced to the LaserWriter, LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX are as follows:

Macintosh DB-9 plug		LaserWriter DB-25 plug	
Signal	Pin#	Pin#	Signal
GND	3	7	GND
TxD-	5	3	RxD
TxD+	4	8	DCD
HSK <sub>o</sub>	7	20	DTR
RxD+	8		
RxD-	9	2	TxD

Cable pin assignments for connecting the Macintosh Plus, Macintosh SE, and Macintosh II to LaserWriter, LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX are as follows:

Macintosh MINI DIN-8 plug		LaserWriter DB-25 plug	
Signal	Pin#	Pin#	Signal
GND	4	7	GND
TxD-	3	3	RxD
TxD+	6	8	DCD
HSK <sub>o</sub>	1	20	DTR
RxD+	8		
RxD-	5	2	TxD

---

## RS-232C cable configuration for IBM-compatible to LaserWriter

The following details the cable specifications for connecting an IBM-compatible host computer with a DB-25 RS232C serial connector to a LaserWriter.

---

IBM-compatible DB-25 Plug		LaserWriter DB-25 Plug	
Signal	Pin#	Pin#	Signal
Shield	1	1	Shield
TxD	2	3	RxD
RxD	3	2	TxD
RTS	4	4	RTS
CTS	5	5	CTS
DSR	6	8	DCD
GND	7	7	GND
		20	DTR

---

---

---

## Changing communication parameters

On the original LaserWriter, serial communication is always asynchronous, start-stop, with 8 data bits per character (the high-order bit may or may not be used for parity), 1 start bit, and 2 stop bits. There are three programmable parameters: channel (9-pin or 25-pin connector on the original LaserWriter and LaserWriter Plus; 8-pin and 25-pin connector on the LaserWriter IINTX and LaserWriter IINT), baud, and parity. On the original LaserWriter, switch setting 1200 establishes communication with standard parameters (1200 baud, parity ignored). The 9600 and Special switch settings use whatever parameters have been set previously. If no parameters have been set, the defaults are 9600 baud, parity ignored. If you are trying to make contact with a LaserWriter for the first time and you don't know which parameters have been set previously, you should start with the 1200 setting.

As an alternative to using the hardware switches to set the communication parameters, you can also change the channel, baud, and parity with the PostScript `statusdict` operators `setscbatch` for the 9600 switch setting and `setscinteractive` for the Special switch setting (Diablo 630 emulation) on the original LaserWriter, LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX. (The `scc` part of the call stands for *Serial Communications Controller*, which is the device that operates the two serial I/O connectors.)

---

## The operator `setscbatch`—LaserWriter and LaserWriter IINT

The PostScript operator `setscbatch` determines how serial communication is performed on the specified channel for batch jobs when the switch is in the 9600 position. The `setscbatch` operator takes three integer parameters to designate channel, baud, and parity:

*channel baud parity* `setscbatch`

In a PostScript program, these parameters are specified as follows.

### Channel

The integers 9 and 25 designate the 9-pin and 25-pin connectors. The LaserWriter IINT and LaserWriter IINTX use a MINI DIN-8 connector in place of the DB-9. However, the parameter specifying the 8-pin connector on the LaserWriter IINT and LaserWriter IINTX is still selected with the integer 9.

### Baud

This parameter is given as an integer, such as 1200 or 9600. The baud parameter may be an integer less than 100,000. However, the hardware can achieve only certain rates above 9600. These rates are 10,473, 11,520, 12,800, 14,400, 16,457, 19200, 23,040, 28,800, 38,400 and 57,600. The maximum data-transmission rate supported by the software is 57,600.

## Parity

This parameter is specified by an integer from 0 to 3:

- 0 Ignore—the high-order bit of each 8-bit character received is ignored, and the high-order bit of each character transmitted is 0.
- 1 Odd—the high-order bit of each 8-bit character received is checked for odd parity (a PostScript ioerror occurs if it is incorrect), and each character transmitted has odd parity.
- 2 Even—like odd, but for even parity.
- 3 None—all 8 bits of each character are treated as data, and no checking is performed.

You must set the data-transmission rate and parity independently for each of the two channels. The new data-transmission rate and parity do not take effect until the end of the current job. Setting the data-transmission rate to 0 disables the channel, but disabling both channels is not permitted.

The action of `sccinteractive` is almost identical to the operator of `setscbatch`, but `sccinteractive` sets serial-communication parameters to be used when the switch is in the Special position (emulation mode). The switch in combination with the operator selects either interactive or emulation mode.

To determine the currently selected values for the channel, baud, and parity parameters, use the operators `sccbatch` and `sccinteractive`.

The operator `sccbatch` takes a channel number (9 or 25) and returns the 9600 (batch) data-transmission rate and parity previously set for that channel:

```
channel sccbatch baud parity
```

The default is 9600 baud, parity ignored.

---

## The sefscinteractive operator

The `sccinteractive` operator is identical to `sccbatch`, except that it returns the values set for the Special switch (interactive or emulation mode) data-transmission rate and parity for the specified channel.

*channel sccinteractive baud parity*

The default values are 9600 baud, parity ignored—9600 and 0.

---

---

## Controlling communication

The serial communication protocol is minimal. Several character codes are reserved for communication functions and are not passed through to PostScript:

Control-C (\$03)	Interrupt—causes a PostScript <code>interrupt</code> operator to be executed (see the <i>PostScript Language Reference Manual</i> )
Control-D (\$04)	EOF End-of-file
Control-Q (\$11)	XON—start output
Control-S (\$13)	XOFF—stop output
Control-T (\$14)	Status query—when received over either channel, elicits a one-line status message over the same channel; this channel need not be the one through which the LaserWriter is receiving its current job
Return (\$0D)	End-of-line
Line feed (\$0A)	End-of-line (ignored if it immediately follows a Return)

The LaserWriter uses XON/XOFF flow control and expects the host to do likewise. For batch mode operation, XON/XOFF flow control is required; the RS-232C Data Terminal Ready (DTR) signal for flow control was not supported on the original LaserWriter and LaserWriter Plus. Failure to conform to the XON/XOFF flow-control protocol will result in an I/O error when you transfer files longer than about 5000 characters.

On the LaserWriter IINT and LaserWriter IINTX, both XON/XOFF and DTR flow control are supported. XON/XOFF works as described earlier for the LaserWriter. DTR flow control makes use of the DTR control signal available on the LaserWriter IINT and LaserWriter IINTX DB 25 RS-232C connector. (The DTR signal is not available on the MINI DIN-8 connector.) Normally the LaserWriter IINT and LaserWriter IINTX leave the DTR signal turned on, but when they need to stop the flow of characters from the host, they turn the DTR signal off. The host must immediately stop sending characters until the LaserWriter IINT or LaserWriter IINTX turns the DTR signal on again. Similarly, the host can use the Data Set Ready (DSR) signal to control the flow of data sent to it from the LaserWriter IINT or LaserWriter IINTX. Again, failure of the host to conform to the selected flow-control protocol—either XON/XOFF or DTR—may cause a data-buffer overflow in the LaserWriter IINT or LaserWriter IINTX, resulting in I/O errors.

There is no way to limit the reserved control characters in order to pass them through as data to PostScript; and there is no way to transmit characters in the high ASCII range (128 to 255) when the high-order bit is being ignored or used for parity. Thus, the serial link is not a fully transparent channel, which causes no difficulty in normal use because PostScript consists entirely of printable characters. PostScript itself provides a means for encoding arbitrary characters in strings (the `\nnn` escape sequence). True binary data, such as images and encrypted programs, is transmitted in hexadecimal.

When a job terminates, the LaserWriter sends a Control-D EOF over the serial channel. Doing this allows the host application program to synchronize with the LaserWriter if desired and to correlate a given batch of output with the job that generated it. The application does not need to wait for one job to finish before beginning to send the input for the next job.



## Appendix B



# LaserWriter Technical Specifications

This appendix contains hardware and media specifications for the original LaserWriter, LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX models of the Apple laser printers.

The original LaserWriter and LaserWriter Plus laser-printing engines contain built-in laser-printer controllers. The LaserWriter IINT and LaserWriter IINTX laser printers consist of a printer engine configured with either a LaserWriter IINT or LaserWriter IINTX controller. The LaserWriter and LaserWriter Plus share many technical specifications with the laser-print engine configured with a LaserWriter IINT or LaserWriter IINTX laser-printer controller. Therefore, the specifications that follow are identical for each of the four models of laser printers, except where indicated by the specific printer model name.

### Physical specifications

#### LaserWriter

Width	18.5 in. (47.00 cm)
Depth	16.2 in. (41.20 cm) without paper trays 28.2 in. (71.63 cm) with paper trays
Height	11.5 in. (29.20 cm)
Weight	77 lbs. (34.70 kg)

**LaserWriter IINT and LaserWriter IINTX**

Width	20 in. (51.00 cm) without paper tray 26.5 in. (67.30 cm) with paper tray
Depth	18.9 in. (48.00 cm)
Height	8.5 in. (21.6 cm)
Weight	46 lbs. (21 kg)

**General specifications****RAM**

LaserWriter	1.5 MB of RAM
LaserWriter IINT	2.0 MB of RAM
LaserWriter IINTX	2.0 MB to 12.0 MB

**ROM**

LaserWriter	0.5 MB of ROM
LaserWriter Plus	1.0 MB of ROM
LaserWriter IINT	1.0 MB of ROM
LaserWriter IINTX	1.0 MB on main board and up to 8 MB of ROM on an expansion board

**Controller CPU**

LaserWriter	32-bit MC68000 microprocessor running at 11.16 MHz
LaserWriter IINT	32-bit MC68000 microprocessor running at 11.16 MHz
LaserWriter IINTX	32-bit MC68020 microprocessor running at 16.666 MHz

**Built-in serial ports**

LaserWriter	Two serial ports-one D25 for RS-232C and one D9 for RS-422/AppleTalk
LaserWriter IINT	Two serial ports-one D25 for RS-232C and one MINI DIN-8 for RS-422/AppleTalk
LaserWriter IINTX	Two serial ports-one D25 for RS-232C and one MINI DIN-8 for RS-422/AppleTalk

**Hard-disk support**

LaserWriter IINTX	An external SCSI interface for a hard disk (see Chapter 4 for more on of the disk-file system)
-------------------	--

**Print quality**

All models	Text and graphics at a resolution of 300 dpi
------------	--

**Printing speed**

All models	8 pages per minute maximum throughput; actual performance depends on the application
------------	--

## Fonts

LaserWriter	Times, Helvetica, Courier, and Symbol
LaserWriter Plus	Times, Helvetica, Helvetica Narrow, Courier, Symbol, Palatino, ITC Avante Garde Gothic, ITC Bookman, New Century Schoolbook, ITC Zapf Chancery, ITC Zapf Dingbats
LaserWriter IINT and LaserWriter IINTX	Times, Helvetica, Helvetica Narrow, Courier, Symbol, Palatino, ITC Avante Garde Gothic, ITC Bookman, New Century Schoolbook, ITC Zapf Chancery, ITC Zapf Dingbats

## Printer protocols supported

LaserWriter	PostScript, and a subset of the Diablo 630 printer
LaserWriter IINT	PostScript, and a subset of the Diablo 630 printer
LaserWriter IINTX	PostScript, a subset of the Diablo 630 printer, and a subset of the Hewlett-Packard LaserJet <sup>+</sup>

## Printing material feed

All models	Automatic from cassette; manual single-sheet feed
------------	---

## Printing materials

All models	16 lbs. to 24 lbs., single-sheet photocopy bond from cassette; 16 to 36 lbs., letterhead and colored stock; standard weight transparency material; envelopes and labels should be manual fed; an optional envelope cassette is available
------------	--

## Printing material sizes

LaserWriter	Letter, Legal, A4, and B5 sizes; cassettes for each paper size hold 100 sheets; output tray holds 20 sheets
-------------	---

LaserWriter IINT and LaserWriter IINTX	Letter, Legal, A4, and B5 sizes; cassettes for each paper size hold 200 sheets; output tray holds 100 sheets
--	--

### Maximum printable area

LaserWriter	Letter—8.0 by 10.78 in. Legal—6.75 by 12.84 in. A4—7.41 by 10.86 in. B5—7.69 by 10.16 in.
LaserWriter IINT and LaserWriter IINTX	Letter—8.0 by 10.78 in. Legal—8.0 by 13.78 in. A4—7.79 by 11.08 in. B5—6.45 by 9.76 in.

### Power consumption

All models	Standby	170 Watts average
	Operating	900 Watts maximum (115V)
		780 Watts maximum (220V)
		880 Watts maximum (240V)

### Acoustic noise

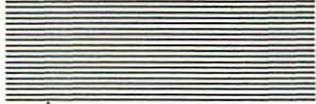
All models	60 dB(A) maximum during operation and 50 dB(A) during standby per ISO 17779, sound power measurement
------------	--

### Power requirements

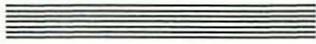
All models	US/Japan	90 to 126 VAC, 50 to 60 Hz
	European	198 to 264 VAC, 50 Hz

### Safety and environmental approval

All models	UL 478 5th edition CSA C22.2 No. 154 FCC, part 15, Class B CDRH, CFR Part 1040, Laser products
------------	---



## Appendix C



# Laser Printer Controllers

The laser-print engine provides a Canon standard laser-printer video interface for connection of one of the three Apple laser-printer controllers. Apple computers communicate with the controllers over a variety of interfaces. These interfaces are SCSI, RS-232-C, RS-422, and AppleTalk. The Macintosh Plus, Macintosh SE, and Macintosh II communicate with LaserWriter IISC over a SCSI channel (LaserWriter IISC is discussed separately in the Apple *LaserWriter IISC Reference*). Host computers communicate with LaserWriter IINT and LaserWriter IINTX through the RS-232-C serial, RS-422 serial, and Appletalk network channels.

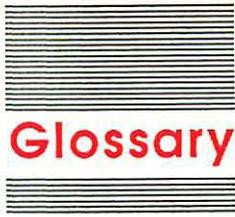
---

### Controller models

The Apple laser-printer-controller models for the laser-print engine are as follows:

- **LaserWriter IISC:** A single-user personal laser-printer controller. The LaserWriter IISC communicates with the host over the SCSI bus, and does not support serial RS-232-C, RS-422, or the AppleTalk network. The LaserWriter IISC uses QuickDraw and a set of specially designed Macintosh system fonts to print high resolution text and graphics output.

- **LaserWriter IINT:** A network laser-printer controller. The LaserWriter IINT has a built-in PostScript interpreter and supports both RS-232-C and RS-422 serial communication, as well as AppleTalk network Printer Access Protocol (PAP).
- **LaserWriter IINTX:** A configurable high-speed network laser-printer controller. The LaserWriter IINTX has a built-in PostScript interpreter, and supports both RS-232-C and RS-422 serial communication, as well as AppleTalk network PAP. The PostScript code and the LaserWriter IINTX's external SCSI port provide support for connecting a SCSI hard disk that can be used as additional virtual memory for caching downloadable fonts. RAM can be upgraded on the controller board. Additional ROM-based fonts can be added by installing a expansion board.



## Glossary

**alert message:** A warning or report of an error in the form of an **alert box**, a sound from the computer's speaker, or both.

**alert box:** A box that appears on the screen to give a warning or to report an error message during use of an application.

**ANSI:** Acronym for *American National Standards Institute*, which sets standards for many technical fields and is the most common standard for computer terminals.

**Apple key:** A key marked with an outlined Apple symbol; on older Apple II machines, it's called *Open Apple*. On some keyboards, the Apple key can also act as the **Command key**, and carries both the Apple symbol and the propeller-shaped Command symbol.

**Apple menu:** The farthest-left menu in the menu bar, indicated by an Apple symbol, from which you choose **desk accessories**.

**AppleTalk connector:** A piece of equipment consisting of a connection box, a short cable, and a 9-pin or 25-pin plug that allows a device to be part of an AppleTalk network. The LaserWriter IINT and LaserWriter IINTX use an 8-pin plug for the AppleTalk network connection.

**AppleTalk Link Access Protocol (ALAP):** The lowest-level protocol in the AppleTalk architecture, managing node-to-node delivery of frames on a single AppleTalk network.

**AppleTalk Network:** The AppleTalk connectors, cables, cable extenders, and software that link computers and peripheral devices, such as printers or file servers, together in a communication network.

**AppleTalk Transaction Protocol (ATP):** An AppleTalk Protocol that's a Datagram Delivery Protocol (DDP) client. It allows one ATP client to request another ATP client to perform some activity and to report the activity's result as a response to the requesting socket with guaranteed delivery. See **Datagram Delivery Protocol**.

**ascent line:** A horizontal line that coincides with the tops of the tallest characters in a **font**. See also **base line**, **descent line**, **x-height**.

**ASCII:** Acronym for *American Standard Code for Information Interchange*, pronounced *ASK-ee*. A code in which the numbers from 0 to 127 stand for text characters. ASCII code is used for representing text inside a computer and for transmitting text between computers or between a computer and a peripheral device.

**asynchronous:** Not synchronized by a mutual timing signal or clock. Compare **synchronous**.

**asynchronous transmission:** A method of data transmission in which the receiving and sending devices don't share a common timer, and no timing data is transmitted. Each information character is individually synchronized, usually by the use of start and stop bits. The time interval between characters isn't necessarily fixed. Compare **synchronous transmission**.

**ATP:** See **AppleTalk Transaction Protocol**.

**B-spline:** A mathematical description of a curve that is used in defining outlines of a character for use on the LaserWriter.

**baud:** A unit of data transmission speed: the number of discrete signal state changes per second. Often, but not always, equivalent to *bits per second*. Compare **bit rate**.

**binary:** The representation of numbers in the base-2 system, using only the two digits 0 and 1. For example, the numbers 0, 1, 2, 3, and 4 in base-10 notation are 0, 1, 10, 11, and 100 in binary notation.

**binary file:** A file whose data is to be interpreted in binary form. Machine-language programs and pictures are stored in binary files. Compare **text file**.

**bit:** A single binary digit that consists of either a zero or a one.

**bit encoding:** A method of electrically distinguishing a data unit on a network.

**bit image:** A collection of bits in memory that have a rectilinear representation. The display on the screen is a visible bit image.

**bit map:** A set of bits that represent the position and state of a corresponding set of items or pixels. A bit map can be converted to graphics data and sent to the printer. See **pixel**.

**bit-mapped character:** A character that exists in a computer file or in memory as a bit map, that is drawn as a pixel pattern on the screen, and that is sent to the printer as graphics data.

**bit-mapped font:** A font made up of bit-mapped characters. Fonts stored in a Macintosh system file are bit-mapped fonts, for example. Compare **internal font**.

**bit rate:** The speed at which bits are transmitted, usually expressed as *bits per second*, or *bps*. Compare **baud**.

**bits per second:** See **bit rate**.

**bottleneck:** A standard routine for performing an operation, such as any QuickDraw call for performing an operation on a graphics object.

**bps:** See **bit rate**.

**built-in fonts:** Fonts built into the LaserWriter printer. You install the Macintosh screen versions with the printer software and use them when creating or changing a document. The LaserWriter substitutes the built-in font for the screen version when you print.

**channel:** Short for **communication channel**.

**character:** Any symbol that has a widely understood meaning and thus can convey information. There are 256 possible characters, which correspond to the range of 8-bit binary numbers (from 0 to 255). Compare **control character**.

**character origin:** The point on a **base line** used as a reference location for drawing a character.

**character set:** The entire set of characters that can be printed by a device or displayed on the screen.

**character-set encoding:** A set of rules for naming and positioning characters in a font.

**character-set-encoding table:** Part of the style-mapping table. The table is used to specify **character-set encoding**.

**character style:** A set of stylistic variations, such as bold, italic, and underline. The empty set indicates plain text (no stylistic variations).

**character string:** Two or more characters read or sent in sequence; for example, Esc Z 3 Control-4 is a character string.

**Chooser:** A **desk accessory** installed along with the LaserWriter and Imagewriter II printer software. Chooser lets you print from any attached printer for which you have a printing **resource** on the startup disk. You also use Chooser to designate the port to which a printer is attached. Chooser replaces the older Choose Printer accessory.

**Clear To Send:** An RS-232-C signal from a DCE to a DTE that is normally kept false until the DCE makes it true, indicating that all circuits are ready to transfer data out. See **Data Communication Equipment, (DCE); Data Terminal Equipment, (DTE).**

**client:** An entity that uses a part of the AppleTalk network structure to communicate with another entity. For example, the application that opens a socket in a LaserWriter is that socket's client.

**command:** An instruction that can be sent to a device, such as a computer or printer, to cause it to perform a specific function.

**Command key:** A key that, when held down while another key is pressed, causes a command to take effect. When held down in combination with dragging the mouse, the Command key lets you drag a window to a new location without activating it. The Command key is marked with a propeller-shaped symbol.

**communication channel:** The logical path along which information is passed between the LaserWriter and a workstation.

**compile:** To convert a program (source file) written in a high-level programming language (such as BASIC) into a file of commands in a lower-level language (such as machine language) for later execution.

**control character:** A nonprinting character that controls or modifies the way information is printed or displayed. In the Apple II family, control characters have ASCII values between 0 and 31, and are typed from a keyboard by holding down the Control key while pressing some other key. In the Macintosh family, the Command key performs a similar function. See **Command key** and **control key.**

**control code:** One or more nonprinting characters—included in a text file—whose function is to change the way a printer prints the text. For example, a program may use certain control codes to turn boldface printing on and off. See **control character.**

**control key:** A general term for a key that controls the operation of other keys; for example, Caps Lock, Command, Control, Open Apple, Option, and Shift are control keys. When you hold down or engage a control key while pressing another key, the combination makes that other key behave differently. Also called a *modifier key.*

**coordinated font:** A font that has been assigned a valid **font class** and **font name.** Compare **noncoordinated font.**

**data bits:** The bits in a communication transfer that contain information. Compare **start bit, stop bit.**

**Data Carrier Detect (DCD):** An RS-232-C signal from a DCE (such as a modem) to a DTE (such as an Apple IIc) indicating that a communication connection has been established. See **Data Communication Equipment, (DCE); Data Terminal Equipment, (DTE).**

**Data Communication Equipment (DCE):** As defined by the RS-232-C standard, any device that transmits or receives information. This device is usually a modem.

**data format:** The form in which data is stored, manipulated, or transferred. Serial data transmitted and received typically has a data format of 1 start bit, 5 to 8 data bits, an optional parity bit, and 1 or 2 stop bits.

**Datagram Delivery Protocol (DDP):** An AppleTalk protocol managing socket-to-socket delivery of datagrams over AppleTalk networks.

**data set:** A device that modulates, demodulates, and controls signals transferred between business machines and communication facilities. See **modem**.

**Data Set Ready (DSR):** An RS-232-C signal from a DCE to a DTE indicating that the DCE has established a connection. See **Data Communication Equipment, (DCE); Data Terminal Equipment, (DTE)**.

**Data Terminal Equipment (DTE):** As defined by the RS-232-C standard, any device that generates or absorbs information, thus acting as an endpoint of a communication connection. A computer might serve as a DTE.

**Data Terminal Ready (DTR):** An RS-232-C signal from a DTE to a DCE indicating a readiness to transmit or receive data. See **Data Communication Equipment, (DCE); Data Terminal Equipment, (DTE)**.

**DCD:** See **Data Carrier Detect**.

**DCE:** See **Data Communication Equipment**.

**DDP:** See **Datagram Delivery Protocol**.

**demodulate:** To recover the information being transmitted by a modulated signal. For example, a conventional radio receiver demodulates an incoming broadcast signal to convert it into the sound emitted by the radio's speaker. Compare **modulate**.

**derived font:** A font whose characteristics are partially determined by modifying an **intrinsic font**. A derived font might be one whose characters are scaled from an intrinsic font to achieve a desired size, or are slanted to achieve an italic style.

**descent line:** A horizontal line that coincides with the bottoms of character descenders (such as the tail on a lowercase "p") extending farthest below the **base line**. See also **ascend line, font size, base height**.

**desktop publishing:** A system providing you with the ability to produce publication-quality documents. A Macintosh, an Apple LaserWriter, and page-formatting software provide this capability.

**device:** A hardware component of a computer system, such as a video monitor, a disk drive, or printer.

**dialog box:** A box that contains a message requesting more information from you. Sometimes, the message warns you that you're asking your computer to do something it can't do or that you're about to destroy some of your information. In these cases, the message is often accompanied by a beep.

**DIP switch:** A small switch that affects a printer function and can be operated manually. There are six switches in one DIP switch assembly on the back of the Apple LaserWriter IINTX laser printer. *DIP* stands for *dual in-line package*.

**download:** To send data to a device (in this case, the LaserWriter) for use by that device. Typically, the LaserWriter driver downloads user documents and the fonts necessary to print them to the LaserWriter.

**drag:** To position the pointer on something, press and hold the mouse button, move the mouse, and release the mouse button. When you release the mouse button, you either confirm a selection or move an object to a new location.

**DSR:** See **Data Set Ready**.

**DTE:** See **Data Terminal Equipment**.

**DTR:** See **Data Terminal Ready**.

**duplex transmission:** Simultaneous two-way, independent transmission of data between two computers or between a computer and a terminal.

**emulation mode:** A manner of operating in which one system imitates another.

**escape code:** A sequence of characters that begins with an Escape character and constitutes a complete command. Usually synonymous with **escape sequence**.

**escape sequence:** A sequence of keystrokes, beginning with the Esc key. In **escape mode**, escape sequences are used for positioning the cursor and controlling the display of text on the screen. Escape sequences are also used as codes to control printers.

**even/odd parity check:** In data transmission, a check that tests whether the number of 1 bits in a group of binary digits is even (even parity check) or odd (odd parity check).

**even parity:** In data transmission, the use of an extra bit set to 0 or 1 as necessary to make the total number of 1 bits an even number; used as a means of error checking. Compare **MARK parity**, **odd parity**.

**firmware:** Programs stored permanently in read-only memory (ROM).

**fixed-width font:** A font in which every character takes the same amount of space on the line. In a fixed-width font, the letter *I* takes as much horizontal space on a line as the letter *M*. Compare **proportional font**.

**flag:** A variable whose value (usually 1 or 0, standing for TRUE or FALSE) indicates whether some condition holds or whether some event has occurred. A flag is used to control the program's actions at some later time.

**font:** In typography, a complete set of type in one size and style of character. In computer usage, a collection of letters, numbers, punctuation marks, and other typographical symbols with a consistent appearance. The size can be changed readily. See **font scaling**.

**font class:** A mechanism for grouping fonts by their **style-implementation methods** and **character-set-encoding** schemes.

**font family:** A group including all the styles and sizes of the characters in that font. For example, the Geneva font family includes 9-point to 36-point characters in italic, bold, outlined, and other styles. A font family is defined in a 'FOND' resource.

**Font file:** A specific file used with the **Font Mover**. You copy fonts to and from this file to the System file of the disk that Font Mover is on.

**Font Mover:** An application available on your Macintosh *System Disk* and on the *LaserWriter Installation Disk* that allows you to add or remove fonts from a disk's system file.

**font name:** The name, such as *Geneva* or *Times*, given to a font family to distinguish it from other font families. The font name is the name by which the LaserWriter knows a font in a specific style.

**font rectangle:** The smallest rectangle enclosing all character images in a font, if the images were all superimposed over the same **character origin**. Also called the *font-bounding box*.

**font scaling:** A feature that allows Macintosh computers to create all sizes of a font from one size. Scaled fonts in larger sizes are usually not as attractive as the installed font.

**font size:** The size of a font of characters in **points**; equivalent to the distance between the **ascent line** of one line of text and the ascent line of the next line of single-spaced text. Examples of font size are 12 point and 18 point.

**Font Substitution option:** A feature that lets you choose whether or not the Macintosh automatically substitutes LaserWriter fonts for equivalent Macintosh fonts.

**format:** The general shape and appearance of the printer's output, including page size, character width and spacing, line spacing, and other factors.

**frame:** Information used to encapsulate data to be sent on a network.

**framing error:** In serial-data transfer, the absence of the expected stop bits at the end of a received character.

**frequency:** In alternating current (AC) signals, the number of complete cycles transmitted per second. Frequency is usually expressed in hertz (cycles per second), kilohertz (kilocycles per second), or megahertz (megacycles per second). In acoustics, the frequency of vibration determines musical pitch. Compare **duration**.

**full duplex:** A four-wire communication circuit or protocol that allows two-way data transmission between two points at the same time. Compare **half duplex**.

**function:** In a programming language, an instruction that converts data from one form to another. The functions `CHR` or `CHR$`, for example, convert an ASCII code number into its corresponding character.

**half duplex:** A two-wire communication circuit or protocol designed for data transmission in either direction, but not both directions simultaneously. Compare **full duplex**.

**hardware:** In computer terminology, the physical machinery, as opposed to *software*, which is the program instructions.

**hardware reset:** The act of resetting the printer to its default settings by turning the printer off and back on. A hardware reset clears any data in the print buffer. Compare **software reset**.

**hexadecimal:** The representation of numbers in the base-16 system, using the ten digits 0 through 9 and the six letters A through F. For example, the decimal numbers 0, 1, 2,..., 10, 11,..., 15, 16, 17 would be shown in hexadecimal notation as 00, 01, 02,..., 0A, 0B,..., 0F, 10, 11. Each hexadecimal digit corresponds to a sequence of 4 bits. Hexadecimal numbers are usually preceded by a dollar sign (\$).

**high ASCII characters:** ASCII characters with decimal values of 128 to 255. Called *high ASCII* because their high bit (first binary digit) is set to 1 (for *on*) rather than 0 (for *off*).

**high-order byte:** The more significant half of a memory address or other 2-byte quantity. In the 6502 microprocessor used in the Apple II family of computers, the **low-order byte** of an address is usually stored first, and the high-order byte is stored second. In the 68000 microprocessors used in the Macintosh family, the high-order byte is stored first.

**host computer:** The computer that receives information from and sends data to terminals over telecommunication lines. The computer that is in control in a data communication network. The host computer may be a mainframe computer, minicomputer, or microcomputer.

**index:** (1) A number used to identify a member of a list or table by its sequential position. (2) A list or table whose entries are identified by sequential position. (3) In machine-language programming, the variable component of an indexed address, contained in an index register and added to the base address to form the effective address.

**interface:** (1) The point at which independent systems or diverse groups interact. The devices, rules, or conventions by which one component of a system communicates with another. Also, the point of communication between a person and a computer. (2) The part of a program that defines constants, variables, and data structures, rather than procedures.

**internet:** Two or more networks connected by *bridges*. A bridge is the physical piece of hardware that connects independent networks, so that they can communicate through electronic mail, and share resources, such as network printers.

**intrinsic font:** A font whose characteristics are entirely defined in a 'FONT' or 'NFNT' resource. The plain-style font of any family is an intrinsic font. Other styles may or may not be intrinsic. Compare **derived font**.

**LaserWriter Namer:** An application on the *LaserWriter Installation Disk* that you use to name and rename LaserWriter printers.

**leading:** The amount of blank vertical space between the **descent line** of one line of text and the **ascent line** of the next line of single-spaced text. In early typesetting, strips of lead were placed between lines of type for spacing; hence the name *leading* (pronounced LED-ing).

**low ASCII character:** A character with decimal equivalent between 0 and 127, inclusive. Called *low ASCII* because the high bit (leftmost binary digit) is set to 0 rather than 1. The low ASCII characters make-up the standard ASCII character set.

**MARK parity:** A bit of value 1 appended to a binary number for transmission. The receiving device checks for errors by looking for this value on each character. Compare **even parity**, **odd parity**.

**mode switch:** A four-position switch on the back of the LaserWriter that lets you place the LaserWriter in another mode of operation for programming the printer, for emulating a Diablo 630, and so on.

**modem eliminator:** A short cable for connecting two **Data Terminal Equipment (DTE)** devices together without a **Data Communication Equipment (DCE)** device.

**modem port:** A socket on the back of the computer marked by a telephone icon.

**modulate:** To modify or alter a signal so as to transmit information. For example, conventional broadcast radio transmits sound by modulating the amplitude (amplitude modulation, or *AM*) or the frequency (frequency modulation, or *FM*) of a carrier signal.

**monospace font:** Any font in which the width of characters is always the same. (*Mono* means *one*.) For example, in a font used to show program listings, the letter M is the same width as the letter I. Thus, MMMMM is the same width as IIIII.

**network-visible:** Having a network address. Processes in nodes can be visible to the network—nodes themselves are not.

**node:** A device that's attached to and communicates by means of an AppleTalk network.

**odd parity:** In data transmission, the use of an extra bit set to 0 or 1 as necessary to make the total number of 1 bits an odd number; used as a means of error checking. Compare **even parity**, **MARK parity**.

**page setup dialog:** The dialog in which a Macintosh user specifies such printing characteristics as page size and orientation. This dialog is accessed through the File menu. Also called the **style dialog**.

**PAP client:** A process that uses PAP (Printer Access Protocol) calls to transmit and receive information on a network.

**parallel interface:** An **interface** in which several bits of information (typically 8 bits, or 1 byte) are transmitted simultaneously over different wires or channels. Compare **serial interface**.

**parity:** Having the quality that the number of like bits (ones or zeros) is even (**even parity**) or odd (**odd parity**). A **parity bit** may be added to each byte to assure that the proper parity condition is met, and to allow for error checking. For example, if the most significant bit is the parity bit and even parity is set, then the number 4 is transmitted as 10000100 while 5 is transmitted as 00000101. If the number 10000101 is received and even parity is set, then an error is indicated, since there is an odd number of ones (or zeros) in this number. In **MARK parity**, the most significant bit of every byte is set to 1. In **space parity**, the most significant bit of every byte is set to 0. See **even parity**, **MARK parity**, **odd parity**, **parity bit**.

**parity bit:** A bit used to check for errors during data transmission. Depending on the number of 1 bits in a transmission, the parity bit is set to 1 or 0 to make the total number of 1 bits even or odd.

**picture element:** The smallest element of resolution in the rendition of an image. A picture element on the LaserWriter is  $\frac{1}{300}$  inch on a side. Also called a **pixel**.

**pixel:** Same as **picture element**.

**point:** A unit of measurement for type; 12 points equal 1 pica, and 6 picas equal 1 inch; thus, 1 point equals approximately  $\frac{1}{72}$  inch.

**port:** A socket on the back panel of the computer where you can plug in a cable to connect a peripheral device, another computer, or a network.

**printer driver:** A program that controls the exchange of information between the printer and the computer. You must have a separate printer driver for each type of printer that you want to use. See also **print manager**.

**printer font:** A bit-mapped font intended for use by the printer rather than for use on the screen. For the LaserWriter IINTX and LaserWriter IINT, printer fonts are those whose characters are defined (via B-splines) as sets of outlines to be filled. These fonts have several advantages for printing, including standard appearance despite scaling and transportability between devices of different resolutions.

**Printing Manager:** A firmware or software program that provides routines for controlling printing. Your program calls the printing manager routines, and the printing manager calls the **printer driver** for the appropriate printer; the printing manager makes it unnecessary for your program to include a separate printing routine for each type of printer you want to support.

**printer port:** A socket on the back of your computer marked by a printer icon. The printer port, in addition to being a connection for a printer, also serves as the usual attachment point for an AppleTalk connector on Macintosh computers.

**print job:** All processes performed by the LaserWriter in printing a single user's document. The printer driver prepares the user's document as a single job and downloads it to the LaserWriter for execution.

**print job dialog:** The dialog in which the Macintosh user specifies characteristics of a single **print job**, such as the number of copies to print.

**proportional font:** Any font in which different characters have different widths; thus, the space taken up by words having the same number of letters varies. For example, the letter *M* is wider than the letter *I*, so that MMMMM produces a wider string than IIIII. See **monospace font**.

**protocol:** A formal set of rules for sending and receiving data on a communication line.

**read-only memory (ROM):** Memory whose contents can be read, but not changed. Information is placed into ROM once, during manufacture; it then remains there permanently, even when the computer's power is turned off. See also **firmware**.

**re-encoded font:** A font in which characters have been "borrowed" from another font.

**register:** A location in memory where a small amount of information, such as a single byte, is stored temporarily under program control.

**reset:** To restore all the default settings for a device with one action or command. The LaserWriter can be reset by turning it off and back on (a **hardware reset**).

**resource:** Data or code stored in a **resource file** on the Macintosh.

**resource file:** The **resource fork** of a Macintosh file.

**resource fork:** The part of a file that contains data used by an application (such as menus, fonts, and icons).

**ROM:** See **Read-only memory**.

**RS-232 cable:** Any cable that is wired in accordance with the RS-232 standard, which is the common serial data-communication interface standard.

**sans serif:** Without serifs; *serifs* are fine lines that finish off the main strokes of a letter—like the little "feet" on the bottom of the horizontal strokes in the letter *M* in Times Roman typeface. The Avant Garde font used for chapter and section headings in this book is a sans-serif typeface; the Avant Garde *M* looks like this: M.

**screen font:** A font defined to yield the best possible appearance on the CRT screen. This type of font is defined by a bit map, and therefore is subject to degradation as a result of scaling.

**serial interface:** An interface in which information is transmitted sequentially, 1 bit at a time, over a single wire or channel.

**serial port:** The connector for a peripheral device that uses a **serial interface**.

**server:** A **network-visible** entity in a server node.

**server node:** A network node that performs services for other nodes on the network. An example is a LaserWriter printer.

**socket:** One end of an AppleTalk connection.

**socket listener:** The routines in an application that monitor a socket opened by that application.

**start bit:** One or two bits that indicate the beginning of a character in a string of serially transmitted characters.

**stop bits:** One or two bits indicating the end of a character in a string of serially transmitted characters.

**style:** A stylistic variation of a font, such as italic, underline, shadow, or outline.

**style dialog:** Same as the **page setup dialog**.

**style-implementation method:** A set of rules for obtaining a specific style of a font. For example, the italic style is often derived from a plain font by slanting its characters.

**style-name table:** Part of the style-mapping table. Used to specify the **font name**.

**synchronization:** A timing scheme used by the network's physical level to ensure that transmitted data is received properly.

**synchronous:** A mode of data transmission in which a constant time interval exists between transmission of successive bits, characters, or events. Compare **asynchronous**.

**tickler packet:** A packet sent on a periodic basis to ensure that a connection remains open.

**virtual memory:** The random access memory used in the LaserWriter for creating the image of the page.

**wait timeout:** The time period the LaserWriter will wait for input before terminating a print job and closing a communication connection.

**workstation:** An individual work area that includes one or more devices on a network.

**workstation node:** A computer device on a network; for example, a Macintosh computer.

**WYSIWYG :** Acronym for *what you see is what you get*; the concept that the screen image should match the printed image as closely as possible.



# Index

## A

a4 (page type) 85  
a4small (page type) 85  
a4tray (page type) 86  
accessing the LaserWriter 59–65  
acoustic noise specifications of the  
    LaserWriter family 150  
AppleTalk 11, 57–58  
appletalktype 105  
applications, developing 12  
asynchronous, defined 54

## B

b5 (page type) 85  
b5tray (page type) 86  
batch mode 59, 64–65  
    defined 54  
baud parameter (setsccbatch)  
    143  
begin 90  
Bézier curves 18  
bind 108  
bit-mapped fonts 10, 14,  
    16–18, 20  
    matching with printer fonts  
    24–25  
    versus printer fonts 20  
bold text (Diablo 630 emulator) 71  
B-splines 18  
built-in fonts 14

## C

cable configuration 141–142  
cachestatus 109

cartstatus 48  
channel, defined 7  
channel parameter (setsccbatch)  
    143  
character font selection (LaserJet<sup>+</sup>  
    emulator) 73–74  
character height (LaserJet<sup>+</sup>  
    emulator) 74  
character-set encoding 25–26  
character-set-encoding table 29–30  
character widths (LaserJet<sup>+</sup>  
    emulator) 76  
checkpassword 99–100  
classifying fonts 25–33  
client 57  
clipping region (LaserJet<sup>+</sup> emulator)  
    74  
Command key 60  
communication  
    controlling 145–146  
    serial-data 136–146  
    transparent (LaserJet<sup>+</sup> emulator)  
    77  
communication channel, defined 7  
communication parameters,  
    changing 142–143  
connecting the LaserWriter  
    to IBM-compatible host computer  
    142  
    to Macintosh 59–65, 141  
control characters (Diablo 630  
    emulator) 67  
control codes (LaserJet<sup>+</sup> emulator)  
    78–82

controller boards 116–126,  
    151–152  
    functional block diagrams  
    118–119, 122–126  
    print engine and 117–122  
controller CPU specifications 148  
coordinated fonts 21–24  
#copies 113  
copy 107  
copypage 113  
CPU  
    LaserWriter 148  
    LaserWriter IINT 122, 148  
    LaserWriter IINTX 123, 148  
    specifications 148  
currentcacheparams 109, 112  
currentpacking 107, 111

## D

Data Communication Equipment  
    (DCE) 141  
Data Terminal Equipment (DTE)  
    141  
DCE (Data Communication  
    Equipment) 141  
def 90  
defaulttimeouts 99  
deletefile 48  
derived fonts 15  
devdismount 48  
device-resolution images 109–110  
devmount 48–49  
devstatus 49  
Diablo 630 emulator 66–71  
    bold text 71

- changing print parameters with
  - 68–70
  - control characters 67
  - deviations from protocol 70–71
  - double-striking 71
  - emulation mode, and 66–71
  - end of document 70
  - invoking 66–67
  - paper positioning 71
  - persistent parameters for 101
  - proportional fonts 71
  - unsupported feature and commands 71
- dimension specifications (font expansion card) 128–129
- disk file system (LaserWriter IIINTX)
  - 41–51
  - finding fonts on 46
- diskonline 49
- disk operators (LaserWriter IIINTX)
  - 46–51
- diskstatus 47, 49
- document, end of (Diablo 630 emulator) 70
- dostartpage 97
- dosysstart 44, 94
- double-striking (Diablo 630 emulator) 71
- downloading fonts 14, 19, 34–41
  - for all printer drivers 35
  - defined 5
  - file format 37–39
  - from non-Macintosh hosts 39–41
  - with later printer drivers 34
  - longevity and (LaserJet<sup>+</sup> emulator) 78
  - permanently 19, 36, 39
  - temporarily 19, 35–36, 40
- DRAM
  - LaserWriter IIINT 122
  - LaserWriter IIINTX 123–124
- drivers 11
  - classifying fonts for version 3.0 or later 26–33
- DTE (Data Terminal Equipment) 141
- DTR flow control 145–146
- dynamic unloading of fonts 39

## E

- EEROM 68, 69, 70
- eerom 70, 102
- eescratch 68, 69, 101
  - location assignments 69
- emulation mode
  - defined 55
  - Diablo 630 emulator and 66–71
  - LaserJet<sup>+</sup> emulator and 72–82
- end 20, 90
- end of document (Diablo 630 emulator) 70
- environmental specifications 150
  - font expansion card 132–133
- erasepage 113
- errordict 87, 91
- executive 60
- exitserver 44, 91

## F

- FIFO
  - LaserWriter IIINT 123
  - LaserWriter IIINTX 125
- filenameforall 50
- file system operators (LaserWriter IIINTX) 46–51
- findfont 46
- flush 21
- FMSwapFont 28
- 'FOND' resource 17–20, 22, 26, 28, 30, 32–38
  - structure of 27
- FontBBox 26
- font-bounding-box table 26
- font cache (LaserWriter IIINTX)
  - 41–51, 109
- font classes 25
- font expansion card
  - pin assignments 130
  - specifications 128–133
- font families 6, 15
- fontlist 20
- font list, obtaining 20–22
- Font Manager 10
- font names 25
- font numbers 89
- 'FONT' resource 17, 19, 28, 33, 34, 36

- fonts 6, 14–51
  - basic concepts 15–25
  - bit-mapped 10, 14, 16–18, 20, 24–25
  - built-in 14
  - classifying 25–33
  - coordinated 21–24
  - defined 15
  - derived 15
  - downloading 14, 19, 34–41
  - dynamic unloading of 39
  - finding on disk file system 46
  - intrinsic 15
  - matching bit-mapped with printer 24–25
  - missing 31
  - naming 33–34
  - noncoordinated 21–24
  - outline 10
  - printer 14, 18–20, 24–25
  - processing of 17–18
  - proportional (Diablo 630 emulator) 71
  - reencoded 25
  - selection (LaserJet<sup>+</sup> emulator) 73–74
  - specifications 149
- font scan-conversion 88–89
- forall 106, 107
- functional block diagrams (controller boards) 118–119, 122–126

## G

- general specifications of the LaserWriter family 148
- get 102, 106
- getinterval 106
- 'GNRL' resource type 38

## H

- hard disk specifications of the LaserWriter family 148
- hardwareiomode 95
- Hewlett-Packard LaserJet<sup>+</sup> emulator. *See* LaserJet<sup>+</sup> emulator
- hyphen (in font names) 33

- I
  - IBM-compatible host computer, connecting to 142
  - idlefonts 100
  - idle-time font scan-conversion 88-89
  - image 109-110
  - images, device-resolution 109-110
  - immediately evaluated names 107-108
  - initgraphics 113
  - initializedisk 47, 50
  - interactive mode 59, 60-64
    - defined 54
    - line-editing characters 60
  - interface connection specifications (font expansion card) 129-130
  - internet 57
  - intrinsic fonts 15
  - invalidaccess 91
- J, K
  - jobname 104
  - jobtimeout 103
- L
  - LaserJet<sup>+</sup> emulator 72-82
    - character font selection 73-74
    - character height 74
    - character widths 76
    - clipping region 74
    - control codes 78-82
    - deviations from protocol 73-82
    - font selection 73-74
    - line printer fonts 77
    - longevity of downloaded fonts 78
    - paper size interactions 75
    - pitch 73
    - printing orientation 73
    - print pitch 73
    - stroke weight 74
    - symbol set 73, 75
    - transparent communications 77
    - typeface and typeface size 74, 77
  - Laser Prep dictionary 36
  - LaserWriter 2, 3
    - accessing 59-65
    - driver 11
    - fonts 6
    - ROM features 5
    - serial connectors 137
    - switch settings 55
  - LaserWriter Font Utility 41-42
  - LaserWriter Plus 3
    - fonts 6
    - ROM features 5
    - serial connectors 137
    - switch settings 55
  - LaserWriter IINT 4-5
    - basic operation of 84-89
    - controller board 116-126, 152
    - DRAM 122
    - fonts 6
    - ROM features 5
    - serial connectors 138
    - switch settings 56
    - VIA 122
    - video control logic 123
    - video shift register 123
  - LaserWriter IINTX 4-5
    - basic operation of 84-89
    - controller board 116-126, 152
    - disk file system 41-51
    - disk operators 46-51
    - DRAM 123-124
    - file system operators 46-51
    - font cache 41-51, 109
    - font expansion card specifications 128-133
    - fonts 6
    - ROM features 5
    - serial connectors 138
    - switch settings 56
    - VIA1 126
    - VIA2 126
    - video control logic 125
    - video shift register 125
  - LaserWriter Ilsc 3
    - controller board 151
  - leading (of a font) 15
  - legal (page type) 85
  - legaltray (page type) 86
  - letter (page type) 85
  - lettersmall (page type) 85
  - lettertray (page type) 86
  - line-editing characters (interactive mode) 60
  - line printer font (LaserJet<sup>+</sup> emulator) 77
  - load 108
  - longevity of downloaded information (LaserJet<sup>+</sup> emulator) 78
  - 'LWPN' resource 37
  - 'LWRT' creator 37
- M
  - Macintosh, connecting to 59-65, 141
    - manualfeed 86-87, 104
    - manual feed 86-87
    - manualfeedtimeout 87, 103
    - margins 98
    - maximum printable area specifications 150
    - mechanical-component location, in print engine 121
    - memory specifications (font expansion card) 131-132
    - missing fonts 31
    - modes. *See* batch mode; emulation mode; interactive mode
    - mode switch settings. *See* switch settings
- N
  - names, immediately evaluated 107-108
  - naming fonts 33-34
  - 'NFNT' resource 17
  - nodes 57
  - noncoordinated fonts 21-24
  - non-Macintosh hosts, downloading fonts from 39-41
  - nonvolatile parameters. *See* persistent parameters
  - note (page type) 85
- O
  - 128K ROM Font Manager, font processing by 18

operating modes. *See* batch mode;  
emulation mode; interactive  
mode  
operators, new (PostScript)  
110–112  
currentcacheparams 112  
currentpacking 111  
packedarray 111  
setcacheparams 112  
setpacking 110  
outline fonts 10

## P

packedarray 107, 111  
packed arrays 106–107  
pagecount 92  
pagestackorder 92  
pagetype 98  
page types 84–86  
a4 85  
a4small 85  
b5 85  
legal 85  
letter 85  
lettersmall 85  
paper path, in print engine 121  
paper positioning (Diablo 630  
emulator) 71  
paper-size interactions (LaserJet<sup>+</sup>  
emulator) 75  
parameters. *See* persistent  
parameters; system  
parameters; volatile parameters  
parity (setscbatch) 144  
PasswordIncorrect 91  
passwords 90–91  
permanent downloading of fonts  
19, 36, 39  
persistent parameters 68, 91–102  
changing 90–91  
checkpassword 99  
defaulttimeouts 99  
dostartpage 97  
dosystart 94  
eescratch 100  
for Diablo 630 emulator 101  
hardwareiomode 95  
idlefonts 100  
margins 98

pagecount 92  
pagestackorder 92  
pagetype 98  
printername 93  
ramsize 93  
scbatch 97  
setdefaulttimeouts 99  
setdostartpage 97  
setdosystart 93  
seteescratch 100  
sethardwareiomode 95  
setidlefonts 100  
setmargins 97  
setpagestackorder 92  
setpagetype 98  
setpassword 99  
setprintername 93  
setscbatch 95  
setsoftwareiomode 94  
softwareiomode 94  
physical specifications of the  
LaserWriter family 147–148  
picture elements 16  
pin assignments  
font expansion card 130  
RS-232C port 140  
RS-422 port 139–140  
pitch (LaserJet<sup>+</sup> emulator) 73  
pixels 16  
ports. *See* RS-232C port; RS-422  
port  
'POST' resource 37, 38  
PostScript 9, 84–113  
changes to 106–113  
new operators of 110–112  
power specifications 150  
font expansion card 131  
print 44  
print engine  
controller boards and 117–122  
mechanical-component location in  
121  
paper path in 121  
printererror 104–105  
printer fonts 14, 18–20  
matching with bit-mapped fonts  
24–25  
versus bit-mapped fonts 20  
printername 93  
printer protocol specifications 149

printer switch settings. *See* switch  
settings  
printing environment 6–11  
working in 54–82  
Printing Manager 10–11, 12  
printing material specifications  
149–150  
printing orientation (LaserJet<sup>+</sup>  
emulator) 73  
printing speed specifications 148  
print parameters 68  
print pitch (LaserJet<sup>+</sup> emulator) 73  
print quality specifications 148  
product 105  
proportional fonts (Diablo 630  
emulator) 71  
PSDump 36, 37  
put 102, 107  
putinterval 107

## Q

QuickDraw 9–10  
quit 60

## R

ramsize 93  
RAM specifications of the  
LaserWriter family 148  
rangecheck 86  
reencoded fonts 25  
renamefile 50  
restore 41, 78, 90, 91  
revision 105  
Rez 38  
RMaker 38  
ROM  
features 5  
LaserWriter IINT 122  
LaserWriter IINTX 124  
specifications 148  
RS-232C port 140  
RS-422 port 138–140

## S

safety specifications 150  
sans serif 26  
save 41, 78, 90, 91

- scaling 19
- sccbatch 97
- SCC interface
  - LaserWriter IINT 123
  - LaserWriter IINTX 124
- SCSI (LaserWriter IINTX) 124
- serial connectors 137-138
- serial-data communication 136-146
- serial port specifications 148
- server nodes 57
- setcachedevice 109
- setcachelimit 109
- setcacheparams 109, 112
- setdefaulttimeouts 99
- setdostartpage 97
- setdosysstart 93
- seteescratch 60, 68, 100
- SetFontLock 28
- sethardwareiomode 95
- setidlefonts 88-89, 100
- setjobtimeout 103
- setmargins 97-98
- setpacking 107, 110-111
- setpagestackorder 92
- setpagetype 98
- setpassword 91, 99
- setprintername 93
- setscbatch 44, 95-96, 143-144
- setscinteractive 145
- setsoftwareiomode 94
- setuserdiskpercent 47, 51
- showpage 87, 113
- 64K ROM Font Manager, font processing by 17
- size (of a font) 15
- socket listener 57
- sockets 57, 58
- software environment of the LaserWriter printers 7-11
- softwareiomode 66, 94
- specifications of the LaserWriter family 147-150
- status 51
- statusdict 47, 70, 84, 86-88, 90-91, 102
- stop 91
- storefile 45
- stroke weight (LaserJet+ emulator) 74

- style (of a font) 16
- style-implementation method 25
- style-mapping table 28-29
- style-name table 30-33
- switch settings 55-57, 60
- symbol set (LaserJet+ emulator) 73, 75
- synchronous, defined 54
- Sys/Start file 43-46
- systemdict 47, 90
- system parameters 90-105

## T

- technical specifications of the LaserWriter family 147-150
- temporary downloading 19, 35-36, 40
- timeouts 87
- timing specifications (font expansion card) 131
- toner 116
- transfer corona wire 116
- transparent communications (LaserJet+ emulator) 77
- typeface and typeface size (LaserJet+ emulator) 74, 77

## U

- underscore character (in font names) 33
- unsupported features and commands (Diablo 630 emulator) 71
- userdict 90
- userdiskpercent 47, 51

## V

- version 108
- vertical bar character (in font names) 21, 33
- VIA (LaserWriter IINT) 122
- VIA1 (LaserWriter IINTX) 126
- VIA2 (LaserWriter IINTX) 126
- video control logic
  - LaserWriter IINT 123
  - LaserWriter IINTX 125
- video interface 120-122

- video shift register
  - LaserWriter IINT 123
  - LaserWriter IINTX 125
- virtual memory 6
- volatile parameters 102-105
  - appletalktype 105
  - jobname 104
  - jobtimeout 103
  - manualfeed 104
  - manualfeedtimeout 103
  - printererror 104
  - product 105
  - revision 105
  - setjobtimeout 103
  - waittimeout 104

## W

- waittimeout 104
- workstation nodes 57

## X, Y

- XON/XOFF flow control 145-146

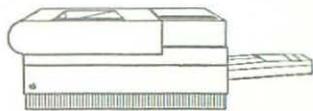
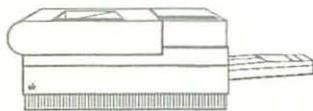
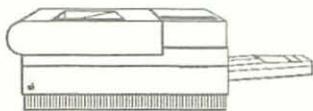
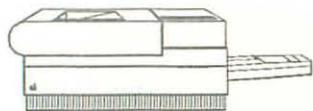
## Z

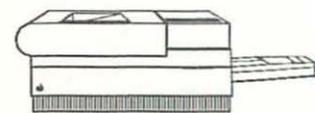
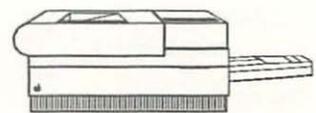
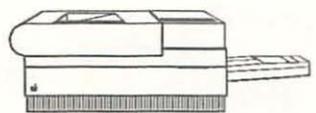
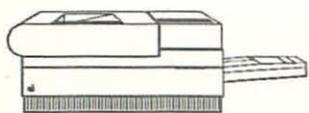
- ZPRAM 56-57, 68, 70, 126

## THE APPLE PUBLISHING SYSTEM

This Apple manual was written, edited, and composed on a desktop publishing system using the Apple Macintosh® computers and Microsoft® Word. Proof and final pages were created on the Apple LaserWriter® IINT. Illustrations were created using Adobe® Illustrator. POSTSCRIPT®, the LaserWriter's page-description language, was developed by Adobe Systems Incorporated.

Text type is ITC Garamond® (a downloadable font distributed by Adobe Systems). Display type is ITC Avant Garde Gothic®. Bullets are ITC Zapf Dingbats®. Program listings are set in Apple Courier, a monospaced font.





# The Apple Technical Library

## The Official Publications from Apple Computer, Inc.

The Apple Technical Library offers programmers, developers, and enthusiasts the most complete technical information available on Apple® computers, peripherals, and software. The Library consists of technical manuals for the Apple II family of computers, the Macintosh® family of computers, and their key peripherals and programming environments.

Manuals for the Apple II family include technical references to the Apple IIe, Apple IIc, and Apple IIgs® computers, with detailed descriptions of the hardware, firmware, ProDOS® operating system, and built-in programming tools that programmers and developers can draw upon. In addition to a technical introduction and programmer's guide to the Apple IIgs, there are tutorials and references for Applesoft BASIC and Instant Pascal programmers.

Manuals for the Macintosh family, known collectively as the Inside Macintosh Library, provide complete technical references to the Macintosh 512K, Macintosh 512K Enhanced, Macintosh Plus, Macintosh SE, and Macintosh II computers. Individual volumes provide technical introductions and programmer's guides to the Macintosh, as well as detailed information on hardware, firmware, system software, and programming tools. The Inside Macintosh Library offers the most detailed and complete source of information available for the Macintosh family of computers.

In addition, titles in the Apple Technical Library offer references to the wide range of important printers, communications standards, and programming environments—such as the Standard Apple Numerics Environment (SANE™)—to help programmers and experienced users get the most out of their computer systems.

> \$19.95 FPT  
USA

## The Official Publications from Apple Computer, Inc.

The Apple Technical Library has been developed to answer technical questions regarding Apple® products that support both the Macintosh® and Apple II families of computers. Addison-Wesley and Apple Computer, Inc., have been busy producing this selection of manuals to provide comprehensive and definitive information on Apple products that add value to both Macintosh and Apple II systems.

Whether you want to learn about guidelines for designing a system interface; need information about communications network protocols, unique programming environments, or internal hardware configurations; or simply want a functional overview of one of Apple's peripheral products, the Apple Technical Library is the best source for answers to many questions posed by developers, professional programmers, and serious enthusiasts.

These books will enable developers of Apple software and hardware to take full advantage of the many advanced features of Apple system and programming tools. The easy-to-read text and detailed indexes will enable you to find what you need quickly. Concise, thorough explanations make learning time more productive.



## LaserWriter® Reference

The Official Publication from Apple Computer, Inc.

Written for the software development community, this manual discusses the hardware and firmware features of the Apple® LaserWriter®, LaserWriter Plus, LaserWriter II<sub>NT</sub>, and LaserWriter II<sub>NTX</sub> printers. The typical user of this reference should be an application or device-level programmer familiar with the PostScript® page description language. The material contained in this reference can be used by Apple and non-Apple CPU programmers who wish to provide application support for any of Apple's PostScript printers.

Key topics include the following:

- A description of the LaserWriter hardware and software components that work together in the Macintosh® printing environment.
- An introduction to font terminology and an overview of the font resources used to structure and identify fonts for the Macintosh computer and LaserWriter family of printers.
- A description of the operational modes of the LaserWriter, LaserWriter Plus, LaserWriter II<sub>NT</sub>, and LaserWriter II<sub>NTX</sub>.
- A description of the unique features of the PostScript interpreter resident in the new LaserWriter II<sub>NT</sub> and LaserWriter II<sub>NTX</sub> models.
- A functional overview describing the interaction between the laser-printer controller and the print engine.
- A specification for designing a LaserWriter II<sub>NTX</sub> font-expansion card.
- A description of the serial-communication ports on the LaserWriter printers.

If you are working in the Macintosh software environment, you should also have the complete *Inside Macintosh* library, which includes Volumes I, II, III, IV, and V.

### Apple Computer, Inc.

20525 Mariani Avenue  
Cupertino, California 95014  
(408) 996-1010  
TLX 171-576

Addison-Wesley Publishing Company, Inc.



ISBN 0-201-19258-6