

LN03R SCRIPTPRINTER Programmer's Supplement

Order Number: EK-LN03R-SP-001

This manual provides information that programmers or system managers need when writing programs that use the facilities of the LN03R printer.

Revision/Update Information: This is a new document.

Operating System and Version: VAX/VMS, Version 4.4 or later
MicroVMS, Version 4.4 or later

Software Version: LN03R SCRIPTPRINTER Software V1.0

**digital equipment corporation
maynard, massachusetts**

First Printing April 1987

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.


No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Copyright ©1986 by Digital Equipment Corporation

All Rights Reserved.
Printed in U.S.A.

The **READER'S COMMENTS** form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem-10	PDP	VT
DECSYSTEM-20	PDT	
DECUS	RSTS	
DECwriter	RSX	

POSTSCRIPT is a registered trademark of Adobe Systems Incorporated, Palo Alto, California

TRANSCRIPT is a registered trademark of Adobe Systems Incorporated, Palo Alto, California

Tektronix is a trademark of Tektronix, Inc., Beaverton, Oregon

Times is a trademark and Helvetica is a registered trademark of Allied Corporation

LaserWriter and LaserWriter Plus are registered trademarks of Apple Computer Corporation

MS(TM)-DOS is a registered trademark of Microsoft Corporation

ML-S744

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by T_EX, the typesetting system developed by Donald E. Knuth at Stanford University. T_EX is a trademark of the American Mathematical Society.

Contents

PREFACE		xi
---------	--	----

CHAPTER 1	LN03R SCRIPTPRINTER PROFILE	1-1
1.1	PRINTER HARDWARE	1-2
1.1.1	Print Engine _____	1-3
1.1.2	Device Controller _____	1-4
1.1.3	Paper Input and Output Devices _____	1-4
1.2	HARDWARE REQUIRED ON HOST SYSTEM	1-4
1.3	PRINTER SOFTWARE	1-5
1.3.1	Translators _____	1-5
1.4	ANCILLARY INTERFACES	1-5
1.5	APPLICATION PROGRAMS	1-5

CHAPTER 2	PostSCRIPT SYSTEM PARAMETERS	2-1
2.1	ACCESSING SYSTEM PARAMETERS	2-2
2.2	PERSISTENT AND VOLATILE PARAMETERS	2-2
2.2.1	Changing Persistent Parameters _____	2-2
2.2.2	Changing Volatile Parameters _____	2-5
2.3	INTERACTIVE MODE	2-5
2.4	SCAN CONVERSION PARAMETER	2-6

2.5	PAPER HANDLING PARAMETERS	2-8
2.5.1	Paper Orientation _____	2-9
2.5.2	Margins _____	2-9
2.6	MISCELLANEOUS SYSTEM PARAMETERS	2-10
2.6.1	Page Counter _____	2-10
2.6.2	Printer Name _____	2-10
2.6.3	Test Page Status _____	2-10
2.6.4	Password _____	2-11
2.6.5	Default Timeouts _____	2-11
2.6.6	Scratch Memory _____	2-11
2.6.7	Name of Current Job _____	2-12
2.6.8	Software Revision _____	2-12
2.6.9	Encoding Vectors _____	2-12

CHAPTER 3	PostSCRIPT EXTENSIONS	3-1
------------------	------------------------------	------------

3.1	PostSCRIPT EXTENSION OBJECT TYPES	3-1
3.2	USING THE EXTENSIONS	3-2
3.2.1	Extensions Restrictions _____	3-2
	CHECKPASSWORD	3-3
	DEFAULTJOBTIMEOUT	3-4
	DEFAULTTIMEOUTS	3-5
	DOSTARTPAGE	3-6
	EESCRATCH	3-7
	EEXEC	3-8
	EXECUTIVE	3-9
	EXITSERVER	3-10
	IDLEFONTS	3-11
	JOBNAME	3-13
	JOBTIMEOUT	3-14
	MARGINS	3-15
	PAGECOUNT	3-16
	PAGESTACKORDER	3-17

PAPERSIZE	3-18
PRINTERNAME	3-19
PRODUCT	3-20
REVISION	3-21
SETDEFAULTJOBTIMEOUT	3-22
SETDEFAULTTIMEOUTS	3-23
SETDOSTARTPAGE	3-24
SETEESCRATCH	3-25
SETIDLEFONTS	3-26
SETJOBTIMEOUT	3-28
SETMARGINS	3-29
SETPASSWORD	3-31
SETPRINTERNAME	3-32
SETVMLIMIT	3-33
WAITTIMEOUT	3-34

CHAPTER 4	FONT METRICS	4-1
4.1	FONT METRIC FILE NAMES	4-1
4.2	FORMAT OF FONT METRIC FILES	4-3
4.2.1	Global Section _____	4-5
4.2.2	Character Metrics Section _____	4-7
4.2.3	Kerning Data Section _____	4-8
4.2.4	Composite Character Section _____	4-10

<hr/>		
CHAPTER 5	OPTIMIZING PERFORMANCE	5-1
5.1	BASIC CONCEPTS	5-1
5.2	SMALLER OPERATOR NAMES	5-2
5.3	AUTOMATIC SPACING	5-2
5.4	USE OF BIND OPERATOR	5-2
5.5	ELIMINATING UNNECESSARY PROCEDURE CALLS	5-2
5.6	SCALING THE COORDINATE SYSTEM	5-3
5.7	SHOWPAGE AND COPYPAGE	5-3
5.8	#COPIES FOR MULTIPLE COPIES	5-4
5.9	IDLE FONTS	5-4
<hr/>		
CHAPTER 6	PROGRAMMING FOR THE LN03R	6-1
6.1	PRINTING DEVICE CONTROL	6-1
6.1.1	Using POSTSCRIPT Extension Operators _____	6-2
6.1.2	Redefining Standard POSTSCRIPT Operators _____	6-2
6.2	POSTSCRIPT OPERATORS	6-3
6.2.1	Device-Dependent Programs _____	6-3
6.2.2	Preserving the Context State _____	6-3
6.2.3	Restrictions to Operators _____	6-4
6.3	VIRTUAL MEMORY	6-6

6.4	CHANGES TO STANDARD PostSCRIPT OPERATORS ON THE LN03R	6-9
6.4.1	Limit to Dictionary Stack Entries _____	6-9
6.4.2	findfont _____	6-9
6.4.3	gsave _____	6-9
6.4.4	Polygon Fill _____	6-9
6.4.5	quit _____	6-10
6.4.6	start _____	6-10
6.4.7	vmstatus _____	6-10
6.5	WRITING A PostSCRIPT SOFTWARE DRIVER	6-11
6.5.1	Useful Techniques _____	6-11
6.5.2	Techniques to Avoid _____	6-12
6.6	CONTROL CHARACTERS	6-12
6.7	PRINTABLE CHARACTERS	6-13

APPENDIX A	FONTS	A-1
-------------------	--------------	------------

APPENDIX B	ENCODING VECTORS	B-1
-------------------	-------------------------	------------

APPENDIX C	COMPATIBILITIES AND DIFFERENCES: LN03R AND PrintSERVER	C-1
40		

C.1	VIRTUAL MEMORY	C-1
C.2	OPERATORS	C-1
C.3	COMMUNICATIONS INTERFACE	C-2

APPENDIX D MS-DOS AND ULTRIX INSTALLATIONS **D-1**

D.1	USING THE LN03R WITH MS-DOS	D-1
D.1.1	Using Applications with MS-DOS _____	D-1
D.1.2	Transmitting Print Requests _____	D-2
D.2	USING THE LN03R WITH VMS VAXMATE SERVER	D-3
D.3	USING THE LN03R WITH ULTRIX	D-4
D.3.1	The TRANSCRIPT Application _____	D-4

APPENDIX E CHARACTER BITMAP DISTRIBUTION FORMAT **E-1**

E.1	TAPE FORMAT	E-1
E.2	FILE FORMAT	E-2
E.3	METRIC INFORMATION	E-3
E.4	AN EXAMPLE FILE	E-6

GLOSSARY **Glossary-1**

INDEX

EXAMPLES

2-1	Defining the DECMCS Encoding Vector _____	2-12
2-2	Defining the ISOLatin1 Encoding Vector _____	2-13
3-1	Setting Margins on the Printer _____	3-30
4-1	Sample Font Metric (AFM) File _____	4-4

FIGURES

1-1	The LN03R System _____	1-1
1-2	Major Hardware Systems _____	1-3
A-1	ITC Avant Garde _____	A-2
A-2	Courier _____	A-4
A-3	Helvetica _____	A-6
A-4	ITC Lubalin Graph _____	A-8
A-5	New Century Schoolbook _____	A-10
A-6	ITC Souvenir _____	A-12
A-7	Times _____	A-14
A-8	Symbol _____	A-16
B-1	Adobe Standard Encoding Vector _____	B-2
B-2	Symbol Encoding Vector _____	B-3
B-3	ISO Latin1 Encoding Vector _____	B-5
B-4	DEC MCS (DIGITAL Multinational Character Set) Encoding Vector _____	B-7
E-1	Example of a Descender _____	E-4
E-2	Example with the Origin Outside the Bounding Box _____	E-5

TABLES

2-1	LN03R Persistent Parameters _____	2-4
2-2	LN03R Volatile Parameters _____	2-5
2-3	Interactive Mode Control Characters _____	2-6
2-4	Font Numbers for Idle Scan Conversion _____	2-7
4-1	Font Metric Files _____	4-2
B-1	Characters Listed in the Order of Adobe Standard Encoding _____	B-8
B-2	Characters Listed in the Order of Symbol Encoding _____	B-9
B-3	Characters Listed in the Order of ISO Latin1 Encoding _____	B-11
B-4	Characters Listed in the Order of DEC MCS Encoding _____	B-13
B-5	Characters Listed in Alphabetical Order _____	B-15
C-1	Undefined PRINTSERVER 40 Operators _____	C-2



Preface

Intended Audience

This manual is for programmers creating POSTSCRIPT programs or applications that generate POSTSCRIPT programs.

In addition to programmers, system managers can use this manual to make adjustments to the LN03R printing environment.

You should be familiar with the POSTSCRIPT page description language and with the *POSTSCRIPT Language Reference Manual* and *POSTSCRIPT Language Tutorial and Cookbook*.

Structure of This Document

The manual contains the following chapters and appendixes:

- Chapter 1 introduces the LN03R by describing its major components.
- Chapter 2 explains the system-specific capabilities of the LN03R, organized by function.
- Chapter 3, the main reference section, describes system-specific POSTSCRIPT operators, listed in alphabetical order.
- Chapter 4 explains how your application can obtain font metric information for the fonts supplied with the LN03R.
- Chapter 5 explains how to improve the performance of POSTSCRIPT programs that you or your applications send to the LN03R.
- Chapter 6 gives guidelines for writing POSTSCRIPT programs that will run successfully in the DIGITAL printing environment.
- Appendix A describes and shows samples of the 29 fonts on the LN03R.
- Appendix B shows the encoding vectors for Adobe Standard Encoding, Symbol encoding, ISOLATIN1 encoding, and DECMCS encoding.

- Appendix C describes the compatibilities and differences between the LN03R and PRINTSERVER 40.
- Appendix D describes installing and using the LN03R in the MS-DOS and Ultrix environments.
- A glossary of terms and index is provided for reference.

Conventions Used in This Document

This document uses the following conventions:

Convention	Meaning
Boldface	Boldface words and letters used in formats, examples, and text specify PostScript keywords. In page descriptions, you should type the keyword exactly as shown.
<i>Italics</i>	When <i>italic</i> words and letters appear in command formats, substitute an appropriate word or value for the word or letter. When <i>italic</i> words appear in normal text, they indicate the first use of a new term.
[]	Square brackets in command formats indicate that the enclosed item is optional.
{ }	Braces in command formats enclose lists from which you must choose one alternative. The choices are listed vertically or separated by the vertical bar symbol ().
...	A horizontal ellipsis in command formats indicates that the preceding item(s) can be repeated one or more times.
.	A vertical ellipsis in examples indicates that code has been omitted.

Convention	Meaning
CTRL/x	This symbol indicates that you should press the key labeled CTRL while you simultaneously press another key, for example, CTRL/Z, CTRL/C, CTRL/O.
RET	This symbol indicates that you should press the RETURN key.

Associated Documents

The following documents are associated with the LN03R:

- *VAX/VMS Tools Reference Manual: LN03R SCRIPTPRINTER*
- *VAX/VMS Management/User's Guide: LN03R SCRIPTPRINTER*
- *VAX/VMS Software Installation Guide: LN03R SCRIPTPRINTER*
- *POSTSCRIPT Quick Reference Guide: LN03R SCRIPTPRINTER*
- *LN03R SCRIPTPRINTER Operator Guide*
- *LN03R SCRIPTPRINTER Installation Guide*
- *POSTSCRIPT Translators Reference Manual*
- *POSTSCRIPT Language Tutorial and Cookbook*
- *POSTSCRIPT Language Reference Manual*

The following document may also be useful for reference:

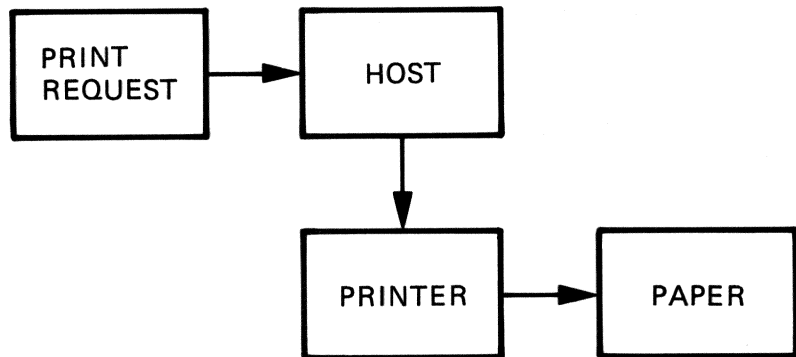
- *Font File Format User's Manual: Order No. EK-CFFFM-UG-001*



LN03R SCRIPTPRINTER Profile

The LN03R combines hardware and software into a printing system that many users can share. The major components of the system are the printer and the host, as shown in Figure 1-1.

Figure 1-1: The LN03R System



MLO-228-87

This chapter introduces the LN03R system by describing its major components. Section 1.1, which describes the printer, deals mainly with system hardware. The remaining sections describe the software.

1.1 Printer Hardware

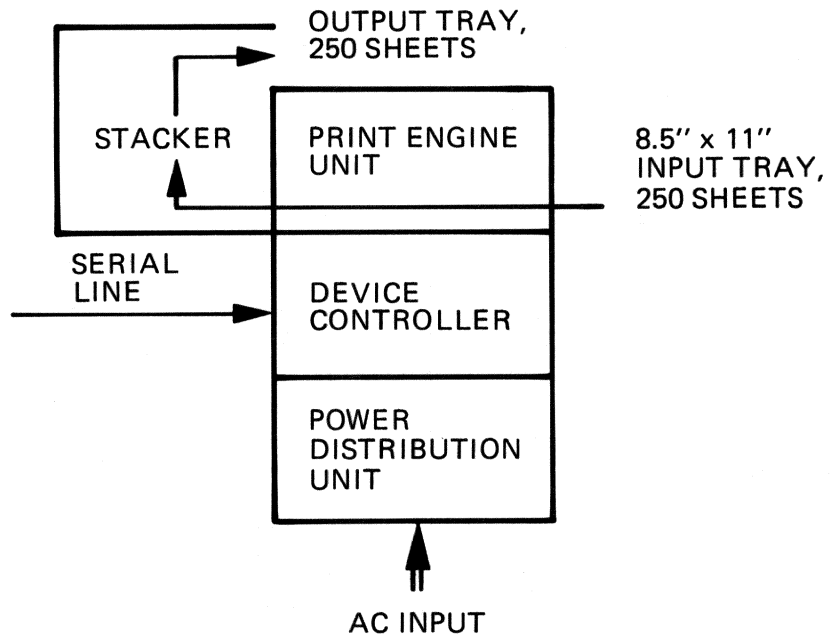
The LN03R printer is a laser output device that contains firmware to receive incoming print files and the following hardware systems:

- A device controller to process POSTSCRIPT file data (page descriptions) into bitmap images
- A print engine to produce marks on paper
- Paper input and output devices
- Power distribution unit

The printer composes the image in a full-page bitmap memory, then processes the image to produce a serial bit stream that is compatible with the print engine. The print engine, in turn, uses laser electrophotography to deposit the image on plain paper.

Figure 1-2 illustrates the LN03R subsystems.

Figure 1-2: Major Hardware Systems



MLO-229-87

1.1.1 Print Engine

The print engine produces images. Using a nonimpact printing technique, a low-power laser generates image-forming charges one line at a time on the surface of a belt. Then the printer applies a dry powder (toner) to the belt; the powder adheres only to the charged areas. The powder is then transferred to the paper and fused on.

The print engine accepts 1-bit/pixel data from bitmap memory and generates images having 300-pixel/inch resolution. Maximum printing speed for the print engine is eight A-size pages/minute, although the actual speed depends on the page composition time; complex pages reduce the throughput.

The print engine contains internal logic that synchronizes the printing process. This controller makes sure the paper is fed to the print engine at the proper time, the marks are made on the paper, and then the paper is ejected from the engine.

1.1.2 Device Controller

The device controller is a Motorola 68000-based microprocessor. It contains a PostScript interpreter, firmware in ROM, PostScript virtual memory and full-page bit-mapped memory in RAM, plus communications hardware to support the host serial port and print engine signals.

1.1.3 Paper Input and Output Devices

The LN03R contains a 250-sheet input tray and a 250-page output tray. A device called the *face-down stacker* feeds paper to the output tray.

A switch on the back of the LN03R should agree with the paper size being loaded into the print engine. This setting should also agree with the **papersize** operator as described in Chapter 3.

1.2 Hardware Required on Host System

The host system for the LN03R must have a hard disk drive with 2000 blocks of storage available for LN03R software and an RX50, TK50, or 1600 BPI magnetic tape for software distribution.

The host system must have an RS-232 or RS-423 serial interface to connect to the LN03R. Refer to the *LN03R SCRIPTPRINTER Installation Guide* for more information.

1.3 Printer Software

The LN03R operates with a VMS, MicroVMS, Ultrix, or MS-DOS operating system. The software configuration depends on the host's operating system. Refer to the software installation kit or the operating system documentation for more information.

1.3.1 Translators

Application programs that output data other than POSTSCRIPT must use a translator to convert output files into POSTSCRIPT. Translator facilities differ between operating systems. Refer to the *VAX/VMS Software Installation Guide: LN03R SCRIPTPRINTER* or to the documentation for your operating system.

1.4 Ancillary Interfaces

In addition to the primary hardware systems already discussed, there are additional interfaces into and out of the LN03R system:

- Host serial port—Receives POSTSCRIPT programs and control codes from the host system.
- Print engine operator panel—Allows the operator to control the print engine in order to recover from errors and change configuration.

1.5 Application Programs

Application programs running on host systems produce print files for the LN03R. Examples of such applications are:

- Text formatters
- Computer-aided design tools
- Spreadsheets
- Graphics editors

The output of these applications can be PostScript, ReGIS, Tektronix, or simple ANSI text, although PostScript is preferred.

NOTE

New applications should produce PostScript output for greatest efficiency and to make the best use of the LN03R's features. If your application is designed to produce files for the LN03R, those files should be PostScript page descriptions.

POSTSCRIPT System Parameters

POSTSCRIPT is a page description language that describes the appearance of text, graphical shapes, and scanned images on printed pages. The LN03R converts programs written in POSTSCRIPT to printed pages.

The POSTSCRIPT language is portable. A single program written in POSTSCRIPT produces an equivalent printed document, regardless of which model POSTSCRIPT printer is used. This portability is ensured only for programs that use standard (non-device-specific) POSTSCRIPT operators.

The LN03R has unique features that distinguish it from other POSTSCRIPT printers. Certain *system parameters*, such as password, job names, paper margins, paper sizes, page counts, and timeout values, are manipulated by the LN03R in ways that may apply differently or not at all to other POSTSCRIPT printers. This chapter describes the system parameters on the LN03R.

Accessing the LN03R system parameters require some of the LN03R extensions to the POSTSCRIPT language. Those extensions are discussed in this chapter and are fully described in Chapter 3.

2.1 Accessing System Parameters

POSTSCRIPT stores LN03R system parameters in a special dictionary named **statusdict**. This dictionary contains operator, string, and integer objects that constitute the POSTSCRIPT language extensions for the LN03R. By comparison, the **systemdict** and **userdict** dictionaries contain only the standard POSTSCRIPT operators and procedure definitions.

To alter a system parameter, execute **statusdict begin**. This pushes **statusdict** on the dictionary stack. You can then execute the operators and objects defined in **statusdict**. To exit **statusdict**, execute the **end** statement.

The following example illustrates this procedure using the **setvmlimit** operator:

```
statusdict begin
470 setvmlimit
end
```

2.2 Persistent and Volatile Parameters

The LN03R has two kinds of parameters: persistent and volatile. *Persistent* parameters are stored in nonvolatile memory (NVM) located in the printer, so their values persist even when the machine is turned off. Persistent parameters are sometimes called *nonvolatile* parameters.

Volatile parameters, on the other hand, remain in effect only during the execution of a job. When a job completes, volatile parameters revert to the default values.

2.2.1 Changing Persistent Parameters

To change a persistent parameter, a job must bypass the normal initialization sequence that begins and ends every job. This initialization sequence, called the *server loop*, consists of the system executing the **save** and **restore** operators to ensure that changes to virtual memory in one job do not persist into the next job. By escaping from the server loop, a job can make changes that persist until the printer is powered off.

To escape from the server loop, issue the following POSTSCRIPT statement:

```
serverdict begin password exitserver
```

where:

serverdict	Refers to a special dictionary. This dictionary is distinct from statusdict .
password	Is a string that provides control over the ability of user jobs to escape the server loop.
exitserver	Is the procedure in serverdict that exits the server loop.

CAUTION

1. Application programs should never exit the server loop.
2. From outside the server loop, you have direct access to all system parameters and you can damage the printer configuration beyond your ability to reconfigure or repair it.

To change persistent parameters, POSTSCRIPT programs require a *password*. The system manager sets the system password, and only a system manager should be permitted to change persistent parameters.

Your POSTSCRIPT program checks the system password with the **checkpassword** operator. If your application generates a POSTSCRIPT program that needs the password, the application should first query the user and check the password.

If the password you supply when exiting the server loop is incorrect, **exitserver** executes the error **invalidaccess**.

If the password is correct, **exitserver** performs an implicit **restore** and clears the operand and dictionary stacks, as if preparing to execute the next job. However, it does not perform another **save**. (Actually, the system does start a new job, but without sending the usual end-of-file indication on the communication channel. Therefore, the POSTSCRIPT program must not issue an **end** to match the **serverdict begin**.)

Having executed a successful **exitserver**, your POSTSCRIPT program can invoke the **statusdict** operators that change persistent parameters until the next end-of-file. Changes the program makes to the state of the POSTSCRIPT VM — such as creating new objects and storing values into dictionaries — persist until power-off. The initial states of all subsequent jobs contain the modified VM.

During execution of a program outside the server loop, VM is not protected from changes that could cause the system to malfunction. Also, VM resources consumed by the program remain until power is cycled to the LN03R (the LN03R is turned off and on).

Table 2-1 lists the persistent parameters of the LN03R and their extensions. The table lists, in parentheses, the object type of an extension if it is not an operator. For a complete description of these parameters, see Chapter 3.

Table 2-1: LN03R Persistent Parameters

Parameter	Extension
Communication parameters	setscbatch* , sccbatch
Device controller firmware ID	deviceid
Exit save/restore context	exitserver
Fonts to scan convert	setidlefonts* , idlefonts
Margin values	setmargins* , margins
Name of printer	setprintrname* , printrname
Name of product	product (string)
Page count	pagecount (integer)
Paper size and load direction	papersize
Page stacking order	pagestackorder
Password for system manager	checkpassword , setpassword*
Print engine model ID	deviceid
Printer error	printererror
Revision of machine-dependent POSTSCRIPT software	revision (integer)
Scratch memory values	seteescratch* , eescratch
Test page selection	setdostartpage* , dostartpage
Timeout values, default	setdefaulttimeouts* , defaulttimeouts , setdefaultjobtimeout* , defaultjobtime- out

* Indicates an operator that requires you to exit the server loop.

2.2.2 Changing Volatile Parameters

Some volatile parameters are string or integer values that you can read and write using POSTSCRIPT dictionary operators, such as **get** and **put**. You access the other volatile parameters by using **statusdict** operators. Table 2-2 lists the volatile parameters and the object type of an extension if it is not an operator. For a description of the volatile parameters, see Chapter 3.

Table 2-2: LN03R Volatile Parameters

Parameter	Extension
Interactive mode	executive
Name of current job	jobname (string)
Timeout values	jobtimeout , setjobtimeout , waittimeout
Virtual memory limit	setvmlimit

2.3 Interactive Mode

Normally, the LN03R printer operates in batch mode, consuming and executing a sequence of POSTSCRIPT programs received from the host system. The LN03R also supports *interactive* mode by which you interact with the LN03R's POSTSCRIPT interpreter by connecting an ANSI terminal to the printer with a standard RS-232 serial interface. Use a "null modem" cable, which reverses the Transmit Data and Receive Data signals, to connect the terminal to the host serial port of the LN03R printer. Characters typed at the terminal are transmitted to the printer. The printer then displays the typed characters (and other information) on the terminal screen.

Interactive mode should be used by application programmers as a POSTSCRIPT debugging tool and should never be invoked from within a POSTSCRIPT program.

The **executive** operator allows you to enter interactive mode. This operator is described in Chapter 3.

To enter interactive mode, enter:

```
executive
```

This entry does not echo to the screen, since the LN03R is in batch mode. After the carriage return, the LN03R enters interactive mode. The POSTSCRIPT interpreter issues the following prompt for user input:

PS>

You can then input POSTSCRIPT statements to the interpreter, which are echoed to the terminal on the LN03R output channel. When a POSTSCRIPT statement is terminated, the interpreter executes the statement.

The following control characters take the actions indicated while in interactive mode:

Table 2-3: Interactive Mode Control Characters

Character	Action
Delete	Backs up and erases one character
CTRL/H (Backspace)	Backs up and erases one character
CTRL/U	Erases the current line
CTRL/R	Redisplays the current line
CTRL/C	Aborts the statement and starts over

Interactive mode continues until you type `CTRL/D`, or until the **quit** or **exitserver** POSTSCRIPT operator is executed.

NOTE

Executing the POSTSCRIPT **stop** operator does not terminate interactive mode. The **stop** operator aborts the statement being executed and prompts for a new one.

2.4 Scan Conversion Parameter

To gain efficiency, the LN03R performs character scan conversion and cache storage during idle time. That is, when not processing a job, the LN03R scan converts characters in commonly-used fonts and point sizes and load the results into the font cache. Page descriptions using the characters in cache execute faster than if the cache fonts were not available.

You can select the fonts to scan convert, using the **setidlefonts** operator. Specify each font to be scan converted, using the following sequence of five integers:

```
font sx*10 sy*10 rot/5 nchars
```

where:

font Is one of the font numbers shown in Table 2-4.
s_x and *s_y* Are the scale factors for x and y.
rot Is the rotation in degrees.
nchars Is the number of characters to be converted.

Table 2-4: Font Numbers for Idle Scan Conversion

Number	Font	Number	Font
0	Courier	15	AvantGarde-Demi
1	Courier-Bold	16	AvantGarde-DemiOblique
2	Courier-Oblique	25	NewCenturySchlbk-Roman
3	Courier-BoldOblique	26	NewCenturySchlbk-Bold
4	Times-Roman	27	NewCenturySchlbk-Italic
5	Times-Bold	28	NewCenturySchlbk-BoldItalic
6	Times-Italic	35	LubalinGraph-Book
7	Times-BoldItalic	36	LubalinGraph-BookOblique
8	Helvetica	37	LubalinGraph-Demi
9	Helvetica-Bold	38	LubalinGraph-DemiOblique
10	Helvetica-Oblique	39	Souvenir-Demi
11	Helvetica-BoldOblique	40	Souvenir-DemiItalic
12	Symbol	41	Souvenir-Light
13	AvantGarde-Book	42	Souvenir-LightItalic
14	AvantGarde-BookOblique		

The system scan converts the specified number of characters of the following 94-character set:

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

For example, the statement

```
mark 9 100 120 18 52 setidlefonts
```

specifies 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.

The *mark* value indicates the bottom of the stack. You can specify multiple fonts by providing a group of five integers for each font. POSTSCRIPT reads in each group of five integers, up to a maximum of 150 integers (30 fonts) in the range 0 to 255. If you omit the integers, POSTSCRIPT does not perform idle-time conversion.

The following characters are set at the factory for idle time scan conversion:

- Courier 10 point: full ASCII set
- Times-Roman and Helvetica 10, 12 and 14 point: alphanumerics and common punctuation
- Times-Bold and Helvetica-Bold 10, 12 and 14 point: lowercase letters

See the description of **setidlefonts** in Chapter 3 for more information.

2.5 Paper Handling Parameters

The LN03R has one slot and one paper input tray, which accommodates either 8.5" x 11" (A) or 210 x 297 mm (A4) paper sizes. A switch on the back of the printer indicates the paper size. The state of this switch is accessible to POSTSCRIPT programs through the use of the **papersize** operator, as described in Chapter 3.

The LN03R has one face-down output stacker on top of the machine, which holds 250 sheets.

2.5.1 Paper Orientation

The following parameters are affected when a paper size is selected for input:

- Paper orientation: Paper feeds into the LN03R short edge first.
- POSTSCRIPT coordinate system positioning: The POSTSCRIPT processor places its default coordinate system origin (0,0) at the lower left-hand corner of sheet, with the positive X axis extending right (along the short edge), and the positive Y axis extending vertically (along the long edge). This orientation is called "Portrait".
- POSTSCRIPT coordinate system unit size: Regardless of paper size, the default units for the POSTSCRIPT coordinate system are points (1/72 inch). There is no default scaling of the units based on paper size. Only the extents of the printable area change to match the paper size.
- POSTSCRIPT clipping values: The clipping path is the edges of the imageable area for the paper being used and is set to its default value each time a POSTSCRIPT job begins. The POSTSCRIPT clipping path is set by reading the paper size switch on the print engine when the interpreter is initialized. These values for A size paper are .25" (75 pixels) from the left and right sides, .18" (54 pixels) from the top and .28" (83 pixels) from the bottom of the paper. The values are the same for A4 paper with one exception: .35" from the right side.

NOTE

The LN03R print engine cannot image to the edges of the sheet.

2.5.2 Margins

Margin setting allows POSTSCRIPT programs to adjust the machine-specific paper registration. Only a Field Service technician or individual responsible for installation or repair should set the LN03R margins. This person sets margins at LN03R installation time or when performing major repair on the printer that affects paper registration.

NOTE

Most POSTSCRIPT programs should not use the margin control operators. The printer does not require machine registration adjustment.

The operators that allow you to access the margins are **setmargins** and **margins**.

2.6 Miscellaneous System Parameters

The remaining system parameters pertain to system status, ID, and current configuration. This section describes those parameters.

2.6.1 Page Counter

You examine the contents of the LN03R page counter to determine the number of pages printed since the printer controller was built. To do this, use the **pagecount** operator.

2.6.2 Printer Name

The printer name is a character string on the test page at printer start-up time. By default, the value is set to "LN03R".

You can use the **setprintername** operator to store a printer name. The maximum length of the string is 31 characters. To obtain the printer name, use the **printername** operator.

2.6.3 Test Page Status

The LN03R prints a test page when powered up. You can control whether or not this test page prints by using the **setdostartpage** operator. To determine if printing the test page is enabled, execute the **dostartpage** operator.

2.6.4 Password

As described in Section 2.2.1, you use the system password to escape the normal server loop and make changes to persistent parameters.

To set the system password, use the **setpassword** operator.

To compare a value with the current password, use the **checkpassword** operator.

2.6.5 Default Timeouts

A timeout facility limits how long the system remains in various states. Two timeout facilities are the *job* timeout and the *wait* timeout. At the beginning of a job, these timeouts are set to default values (initially 0 and 40 seconds, respectively).¹ A POSTSCRIPT program can be used to change the timeouts for a job, if desired. The operators for controlling timeouts are located in **statusdict** and are described in Chapter 3.

The job timeout limits the time a job has to execute, which protects the LN03R from being tied up by a POSTSCRIPT program that runs too long. The program can rejuvenate the timer any number of times during the job.

The wait timeout limits the time the system waits to receive additional input for a job in progress. This protects the LN03R from being indefinitely tied up by a host system failure or disconnection.

If a job or wait timeout expires, the POSTSCRIPT interpreter executes the **timeout** error from **errordict**. With the standard definition of **timeout**, a job running in batch mode terminates. The timeout facility is not ordinarily enabled when the LN03R is in interactive mode.

2.6.6 Scratch Memory

Scratch memory is an area in nonvolatile memory where additional persistent parameters can be stored. You can store values in scratch memory, using the **setescratch** operator. To obtain a value stored in scratch memory, use the **eescratch** operator.

¹ A third timeout, the *manual feed* timeout, is maintained by the POSTSCRIPT interpreter. However, its value is irrelevant in the LN03R, which has no manual feed option.

2.6.7 Name of Current Job

The job name is a string that appears as part of status responses generated during the print job. By default, the value is a null string. Your POSTSCRIPT program can set the job name by associating it with **jobname** in **statusdict**.

2.6.8 Software Revision

You can obtain the revision level of the machine-dependent portion of the POSTSCRIPT interpreter by executing the **revision** operator.

2.6.9 Encoding Vectors

The examples in this section demonstrate how the ISOLatin1 and DEC MCS *encoding vectors* are defined. See Appendix B for more information about the encoding vectors. The encoding vector DECMCSEncoding is defined by changing the vector ISOLatin1Encoding.

Example 2-1: Defining the DECMCS Encoding Vector

```
%! DECMCSEncoding.PS
%
% Create DEC Multinational Character Set (MCS) encoding vector.
/DECMCSEncoding ISOLatin1Encoding 256 array copy def
mark
  8#177 8#240 8#244 8#246 8#254 8#255 8#256 8#257
  8#264 8#270 8#276 8#320 8#336 8#360 8#376 8#377
counttomark
{DECMCSEncoding exch /questionmirror put}
repeat
% stack now contains  mark
  8#250 /currency
  8#327 /OE
  8#335 /Ydieresis
  8#367 /oe
  8#375 /ydieresis
counttomark -1 bitshift % divide by 2
{DECMCSEncoding 3 1 roll put}
repeat
% stack now contains  mark
cleartomark
```

Example 2-2: Defining the ISOLatin1 Encoding Vector

```
%! ISOLatin1Encoding.PS
%
% Create ISO Latin 1 encoding vector, if it does not already exist.

mark
/ISOLatin1Encoding
8#000 1 8#054 {StandardEncoding exch get} for
/minus % at position 8#055, was hyphen
8#056 1 8#217 {StandardEncoding exch get} for
/dotlessi % at position 8#220
8#301 1 8#317 {StandardEncoding exch get} for % all the accents
/space /exclamdown /cent /sterling
/currency /yen /brokenbar /section
/dieresis /copyright /ordfeminine /guillemotleft
/logicalnot /hyphen /registered /macron
/degree /plusminus /twosuperior /threesuperior
/acute /mu /paragraph /periodcentered
/cedilla /onesuperior /ordmasculine /guillemotright
/onequarter /onehalf /threequarters /questiondown
/Agrave /Aacute /Acircumflex /Atilde
/Adieresis /Aring /AE /Ccedilla
/Egrave /Eacute /Ecircumflex /Edieresis
/Igrave /Iacute /Icircumflex /Idieresis
/Eth /Ntilde /Ograve /Oacute
/Ocircumflex /Otilde /Odieresis /multiply
/Oslash /Ugrave /Uacute /Ucircumflex
/Udieresis /Yacute /Thorn /germandbls
/agrave /aacute /acircumflex /atilde
/adieresis /aring /ae /ccedilla
/egrave /eacute /ecircumflex /edieresis
/igrave /iacute /icircumflex /idieresis
/eth /ntilde /ograde /oacute
/ocircumflex /otilde /odieresis /divide
/oslash /ugrave /uacute /ucircumflex
/udieresis /yacute /thorn /ydieresis
/ISOLatin1Encoding where not {256 array astore def} if
cleartomark
```



POSTSCRIPT Extensions

This chapter describes the LN03R extensions to the POSTSCRIPT language. The extensions described in this chapter are in addition to the standard POSTSCRIPT language used by all POSTSCRIPT products. For the standard POSTSCRIPT language, see the *POSTSCRIPT Language Reference Manual*. Before using the extensions described in this chapter, be certain you are familiar with the information in Chapter 2.

3.1 POSTSCRIPT Extension Object Types

While most of the extensions for the LN03R are operators, some are strings, integers, or boolean values. For example, the **jobname** extension is a string object whose value represents the name of the job that the system uses in all status reports concerning the job. You can read and write extensions that are not operators, using the normal POSTSCRIPT dictionary operators, such as **get** and **put**.

The four categories of extensions are operators, procedures, integers, and strings. The extensions described in this chapter are operators or procedures, except for two strings (**jobname** and **product**) and one integer (**revision**). Unless noted otherwise, all extensions are defined in **statusdict**.

3.2 Using the Extensions

Each operator description shows syntax in the following format:

*operand*₁...*operand*_n **operator** *result*₁...*result*_m

The operands precede the operator, with *operand*_n being the top element on the operand stack. The operator pops these objects from the operand stack and consumes them. An example of an operator that expects an operand (the password) is **checkpassword**. After executing, the operator leaves *result*₁ through *result*_m on the operand stack, with *result*_m being the top element. For the **checkpassword** operator, TRUE is returned if the operand entered is correct and FALSE if it is incorrect.

A dash in an operand position indicates that the operator expects no operands. Likewise, a dash in a result position indicates that the operator returns no results. This has the syntax:

— **operator** —

3.2.1 Extensions Restrictions

The POSTSCRIPT extensions are used by operating system software, such as symbionts, device control libraries, and daemons. While it is possible to also use them in application programs, it is not recommended; the results can be unpredictable or undesirable. Also, the extensions are device-specific and, their use in an application can make the application unusable on POSTSCRIPT printers other than the LN03R.

checkpassword

Determines if a string value is equal to the current system password. You need the password to escape from the server loop and make changes to persistent parameters or to virtual memory. See Section 2.2.1 for details.

The initial default password is "LN03R".

Format *string checkpassword boolean****string***

The value to test against the current system password.

boolean

TRUE if the string is equal to the current system password or FALSE if it is not equal, the wrong type, or missing.

defaultjobtimeout

defaultjobtimeout

Returns the value of the job default timeouts. This value is set by the **setdefaultjobtimeout** or **setdefaulttimeout** operator. If the job default timeout is not set, its value is 0 (that is, the default is to disable job timeouts).

Format — **defaultjobtimeout** *job*

job

The returned job default timeout.

Error

stackoverflow

The operand stack is full, preventing the operator from adding the returned value to the stack.

defaulttimeouts

Returns the value of the *job*, *manualfeed*, and *wait* timeouts set by the **setdefaulttimeouts** operator.

Format — **defaulttimeouts** *job manualfeed wait*

job

The default value for *jobs*, in seconds. Standard value is zero.

manualfeed

The default value *manualfeed*, in seconds. This value is ignored by the LN03R.

wait

The default value for the *wait* timeout, in seconds. Standard value is 40.

Error

stackoverflow

The operand stack is full, preventing the operator from adding the returned value to the stack.

dostartpage

dostartpage

Determines whether a test page will be printed at power on. You control printing of a power-on test page by invoking the **setdostartpage** operator. The initial default is TRUE.

Format — **dostartpage** *boolean*

boolean

The TRUE or FALSE value of the start page parameter.

Error

stackoverflow

The operand stack is full, preventing the operator from adding the returned value to the stack.

eescratch

Returns the value of the specified array element stored in nonvolatile memory for scratch use. If the specified scratch value is not set, its value is 0.

NOTE

eescratch index locations 0 to 31 are reserved for DIGITAL software.

Format

index eescratch value

index

An integer in the range of 0 to 63_{10} , which is an index into the array stored in nonvolatile memory.

value

An integer in the range of 0 to 255_{10} which is the data contained in the array in nonvolatile memory.

Errors***rangecheck***

An operand value is not within the allowed range of values.

stackunderflow

The operand stack is empty when the operator expects a value.

typecheck

An operand value is not the correct data type.

eexec

eexec

Decodes an encrypted file or string data and executes it as a standard input stream.

Format *file eexec* —

file

The object on the operand stack to be passed through an encryption filter. The action is terminated by an end-of-file condition (CTRL/D) or by closing the current input file.

Errors

stackunderflow

The operand stack is empty when the operator expects a value.

typecheck

An operand value is not the correct data type.

executive

Puts the LN03R into interactive mode. Do not use this operator in a POSTSCRIPT program. For more information about interactive mode, see Section 2.3.

Format — **executive** —

exitserver

exitserver

Escapes from the server loop in order to change persistent parameters. See Section 2.2.1 for details on changing persistent parameters.

exitserver is defined in **serverdict**, not in **statusdict**.

Format *password* **exitserver** —

password
A string set by **setpassword**.

Error

invalidaccess
Password is incorrect or missing.

idlefonts

Returns the characters that the LN03R scan converts during idle time. These characters are set by the **setidlefonts** operator. If **setidlefonts** was not called, this operator returns the list of default idle fonts set at the factory. See Section 2.4 for details on idle-time scan conversion.

Format — **idlefonts** *mark [font s_x s_y rot $nchars$]....*

mark

A position marker (character) that delimits the bottom of the list of scan-converted characters. The group of five integers following the mark is repeated for each font containing characters to be scan converted during idle time.

Idle-time font caching is performed only for fonts present in VM during idle time (those that are built into the system or have been down-line loaded permanently outside the server loop).

font

An integer that indicates the font of the scan-converted characters. See Table 2-4 the list of values.

s_x , s_y

Integers representing the scale factor for the x and y coordinates of the scan-converted characters, multiplied by 10.

rot

The rotation, in degrees, of the scan-converted characters, divided by 5. For example, characters rotated by 90 degrees have a **rot** value of 18.

$nchars$

The number of characters to be converted, beginning with the first character and advancing through the following set:

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

Error

stackoverflow

The operand stack is full, preventing the operator from adding the returned value to the stack.

jobname

Specifies the name of the job. This object is a string and not an operator. The name is an ordinary data value, stored in **statusdict**, that you access, using either the **get** or **put** operator.

If defined by a POSTSCRIPT program, the job name appears as part of status responses generated during the remainder of the job. The default is a null object.

Format — **jobname** *string****string***

After a **get** operation, this string object contains the job name. For a **put** or **def** operation, this string object contains the name of the job to be set.

Errors

jobname does not cause errors. However, the **get**, **put**, and **def** operators can cause errors. See the *POSTSCRIPT Language Reference Manual* for details.

jobtimeout

jobtimeout

Returns the amount of time remaining before the batch job timeout occurs. This value can be set by **setjobtimeout**, or by the default, **setdefaultjobtimeout**. A value of 0 means that the job will never time out.

Format — **jobtimeout** *integer*

integer

The amount of time remaining before the batch job timeout occurs.

Error

stackoverflow

The operand stack is full, preventing the operator from adding the returned value to the stack.

margins

Returns the value of the two margin parameters. The margin parameters are set by the **setmargins** operator. Both values are integers that specify distances in device space (hardware pixels). The initial defaults for both margins are 0.

Format — **margins** *top left****top***

The value of the top margin.

left

The value of the left margin.

Error***stackoverflow***

The operand stack is full, preventing the operator from adding the returned value to the stack.

pagecount

pagecount

Returns the number of pages (including test pages) printed since the controller was built. This value, stored in a page counter in the LN03R, cannot be changed.

This value may not be the same as that shown on the mechanical page counter on the side of the LN03R.

Format — **pagecount** *integer*

integer

The number of pages printed since the printer was built.

Error

stackoverflow

The operand stack is full, preventing the operator from adding the returned value to the stack.

pagestackorder

Determines the stacking order of the paper. This operator exists for compatibility with the PRINTSERVER 40 POSTSCRIPT printer.

Format — **pagestackorder** *boolean*

boolean

Returns FALSE if the first page printed faces the back of the second page, and TRUE if the first page faces away from the second. The LN03R will always return TRUE, since paper is always stacked face down.

Error

stackoverflow

The operand stack is full, preventing the operator from adding the returned value to the stack.

papersize

papersize

Returns the paper size as selected by the paper size switch on the back of the LN03R. This switch corresponds to the size of the paper in the paper input tray. The POSTSCRIPT interpreter uses the paper size switch to determine the default clipping region for the page. This operator exists for compatibility with the PRINTSERVER 40 POSTSCRIPT printer.

Format — **papersize** *boolean name*

boolean

Returns the load orientation of the paper. If the paper is loaded short edge first, the value is TRUE; if the long edge is first, the value is FALSE. The LN03R always returns TRUE, since paper feeds short edge first.

name

The name of the tray, which corresponds to one of the following strings:

Paper Size Selected	String Returned
letter (8.5" x 11")	lettertray
A4 (210mm x 297mm)	a4tray

These strings are the names of the operators used for paper tray selection by DIGITAL POSTSCRIPT printers with multiple input trays, such as the PRINTSERVER 40. The LN03R has one input tray.

Error

stackoverflow

The operand stack is full, preventing the operator from adding the returned value to the stack.

printername

Returns the name of the printer. The operator stores the name in a string you supply. This value is set by **setprintername**.

Format *string printername substring****string***

A string of characters, including spaces, that contains as many characters as the saved printer name. The operator places the printer name into the space provided by this string.

substring

The returned value, which is a substring of the string parameter after the operation. The initial default value of the substring is LN03R SCRIPTPRINTER.

Errors***invalidaccess***

An object's access attribute is violated.

stackunderflow

The operand stack is empty when the operator expects a value.

typecheck

An operand value is not the correct data type.

rangecheck

Operand string is too small to contain the printer name.

product

product

Returns the name of the product, which is an ordinary data value, stored in **statusdict**, that you access with the standard **get** operator.

The value of **product** is "SCRIPTPRINTER" and cannot be changed.

Format — **product** *string*

string

After the **get** operation, this string contains the product name stored in the **product** string.

Error

product does not cause errors. However, the **get** operator can cause errors. See the *POSTSCRIPT Language Reference Manual* for details.

revision

Returns the revision level of the machine-dependent portion of the POSTSCRIPT interpreter. The revision level is a data value, stored in **statusdict**, that you access with the **get** operator.

Format — **revision** *integer****integer***

The revision number of the machine-dependent portion of the POSTSCRIPT interpreter. The standard value is 1.

Error

revision does not cause errors. However, the **get** and **put** operators can cause errors. See the *POSTSCRIPT Language Reference Manual* for details.

setdefaultjobtimeout

setdefaultjobtimeout

Sets the job default timeout value. The operator does not take effect until the start of the next job.

Format *job* **setdefaultjobtimeout** —

job

A positive integer representing the default timeout value for jobs, in seconds. The standard value is zero, which disables timeouts for jobs. Positive integers greater than 14 are permitted.

Errors

rangecheck

An operand value is not in the allowed range of values.

stackunderflow

The operand stack is empty when the operator expects a value.

typecheck

An operand value is not the correct data type.

setdefaulttimeouts

Sets the values of the default job, manualfeed, and wait timeouts. To disable timeouts, enter 0.

Format *job manualfeed wait* **setdefaulttimeouts** —***job***

A positive integer that sets the default value for jobs, in seconds. The standard value is zero. Positive integers greater than 14 are permitted.

manualfeed

A positive integer that sets the default value for manual feed, in seconds. This value is ignored by the LN03R.

wait

A positive integer that sets the default value for the wait timeout, in seconds. The standard value is 40.

Errors***invalidaccess***

Attempted to set persistent parameters without first exiting the server loop.

rangecheck

An operand value is not in the allowed range of values.

stackunderflow

The operand stack is empty when the operator expects a value.

typecheck

An operand value is not the correct data type.

setdostartpage

setdostartpage

Sets the power-on default for either printing or not printing a test page.

Format *boolean setdostartpage* —

boolean

A TRUE value causes the printer to print a test page on power up. A FALSE value suppresses printing of a test page at power up.

Errors

invalidaccess

Attempted to set persistent parameters without first exiting the server loop.

stackunderflow

The operand stack is empty when the operator expects a value.

typecheck

An operand value is not the correct data type.

seteescratch

Writes a value into an array in nonvolatile memory for storing persistent data. System managers can use the scratch memory to store data needed at a particular site. For example, a system manager can maintain several different page counters for different types of sheets. Locations in the eescratch array can store the counter data.

Format *index value seteescratch* —

index

An integer value in the range 0 to 63₁₀ that is an index into the array stored in nonvolatile memory.

NOTE

eescratch index locations 0 to 31 are reserved for DIGITAL software.

value

An integer value in the range 0 to 255₁₀ that is the data that you are writing to the array in nonvolatile memory.

Errors

invalidaccess

Attempted to set persistent parameters without first exiting the server loop.

rangecheck

An operand value is not in the allowed range of values.

stackunderflow

The operand stack is empty when the operator expects a value.

typecheck

An operand value is not the correct data type.

setidlefonts

setidlefonts

Specifies fonts to be scan converted during idle time. Input to the operator consists of up to 150 integer values, in the range 0 to 255, on the operand stack. The integers are read by POSTSCRIPT in groups of five. Each group contains information on the font, scale factor, rotation, and characters in the font to scan convert. A mark immediately below the bottom integer on the stack delimits the series of integers.

An empty list of integers (that is, a mark that is not followed by integer values) specifies that no characters will be scan converted. See Section 2.4 for details on idle-time scan conversion.

Format *mark [font s_x s_y rot nchars]... setidlefonts —*

mark

A position marker that delimits the bottom of the list of scan-converted characters. The group of five integers following the mark is repeated for each font containing characters to be scan converted during idle time.

font

An integer value that indicates the font of the scan-converted characters. See Table 2-4 for the list of possible values.

s_x , s_y

Integers representing the scale factor for the x and y coordinates of the scan-converted characters, multiplied by 10. For example, a 12-point size is specified by the integer 120.

rot

The rotation, in degrees, of the scan-converted characters, divided by 5. For example, to rotate a character by 90 degrees, specify 18.

nchars

The number of characters to be converted, beginning with the first character and advancing through the following set:

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

Errors***invalidaccess***

Attempted to set persistent parameters without first exiting the server loop.

limitcheck

An operand value is not in the allowed range of values.

typecheck

An operand value is not the correct data type.

unmatched mark

An expected mark is not on the operand stack.

setjobtimeout

setjobtimeout

Sets the timeout for the current job. You can obtain the default job timeout by executing the **defaultjobtimeout** operator.

Format *integer setjobtimeout* —

integer

The timeout, in seconds, that you want to set for the current job. Specify 0 to disable timeouts.

Errors

rangecheck

An operand value is not in the allowed range of values.

stackunderflow

The operand stack is empty when the operator expects a value.

typecheck

An operand value is not the correct data type.

setmargins

Add the specified value to the top and/or left margins. A positive integer widens the margin, a negative integer narrows it. The operator adds two numbers from the operand stack to the current values of the top and left margins, measured in pixels (1/300 inch). The result is a change in the alignment of the image area on the output page. The range of adjustment is limited by the print engine's adjustment ability. The standard values for 8.5" x 11" paper are .25" (75 pixels) from the left and right sides, .18" (54 pixels) from the top and .28" (83 pixels) from the bottom. The values are the same for A4 paper except for the right side: .35" (101 pixels).

When a value is out of the adjustment range, the closest value within the range is used. The limits on margin adjustment for 8.5" x 11" paper are, in pixels:

top: [-25, 1185]
left: [-16, 80]

System managers or technicians should use **setmargins** only during system installation to correct alignment errors. Do not use **setmargins** to set the dimensions of the image area.

This operator does not take effect until the next POSTSCRIPT job executes.

Format *top left setmargins* —***top***

The number of hardware pixels to add to the top margin (short edge of paper resting closest to front of the printer).

left

The number of hardware pixels to add to the left margin (long edge of paper closest to left side of printer).

setmargins

Errors

invalidaccess

Attempted to set persistent parameters without first exiting the server loop.

rangecheck

An operand value is not in the allowed range of values.

stackunderflow

The operand stack is empty when the operator expects a value.

typecheck

An operand value is not the correct data type.

Example

Example 3-1: Setting Margins on the Printer

```
%
% This program will set the X and Y margins on an LNO3R.
%
(LNO3R) % edit this line to equal your system exit password
/Courier findfont 10 scalefont setfont
100 720 moveto
serverdict begin
statusdict begin
dup checkpassword
{
  (Setting new margins. Old margins were: ) show margins exch
  10 string cvs show ( ) show
  10 string cvs show ( ) show
  showpage exitserver
} {(Sorry, wrong password!) show} ifelse
statusdict begin
1 32 setmargins % *** Edit this line. First number is top margin,
                % *second number is left margin. Both are in pixels.
end
```

setpassword

Sets the password. The password must be enclosed in parentheses. The password is stored in nonvolatile memory.

Format *old new setpassword boolean****old***

A string corresponding to the current password. The initial default is "LN03R."

new

A string corresponding to the new password.

boolean

TRUE is returned if *old* is the correct password and the password is changed to *new*. FALSE is returned if *old* is incorrect and no action is taken.

Errors***stackunderflow***

The operand stack is empty when the operator expects a value.

typecheck

An operand value is not the correct data type.

setprintername

setprintername

Sets the name of the printer. The entry must be enclosed in parentheses. The printer name is stored internally and is printed on the test page at printer startup.

Format *string setprintername* —

string

The name of the printer. The string can be 31 characters long.

Errors

invalidaccess

Attempted to set persistent parameters without first exiting the server loop.

limitcheck

Implementation limit exceeded.

stackunderflow

The operand stack is empty when the operator expects a value.

typecheck

An operand value is not the correct data type.

setvmlimit

Allocates the number of bytes to store in virtual memory (VM). By default, the VM limit on the LN03R is set to the maximum bytes available, which is 470K bytes. Applications can reduce the VM limit to ensure transportability among POSTSCRIPT printers.

The **setvmlimit** values are set by the standard POSTSCRIPT operator **vmstatus**. See Section 6.6.7 for more information about **vmstatus** as modified for the LN03R.

Format *limit setvmlimit* —***limit***

The maximum number of bytes for VM, which is 470K bytes. If you specify a value larger than the maximum VM limit, the maximum value is used.

Errors***limitcheck***

An operand value is not in the allowed range of values.

vmerror

Attempted to use more VM than the limit allows.

waittimeout

waittimeout

Returns the wait timeout, which is the seconds the LN03R waits to receive additional characters from the host before aborting the job by executing a *timeout*. At the beginning of a job, the system initializes **waittimeout** to the timeout returned by **defaulttimeouts**; but a POSTSCRIPT program may change the timeout to any positive integer. (In interactive mode, the wait timeout is 0.)

The initial default is 40.

Format — **waittimeout** *integer*

integer

The number of seconds before timeout.

Errors

stackoverflow

The operand stack is full, preventing the operator from adding the returned value to the stack.

Font Metrics

Adobe Font Metrics files are ASCII-encoded files that allow application programs to format text containing characters defined in the fonts provided with the LN03R. Font metrics consist of such information as:

- Character widths
- Kerning information (by character pairs)
- Composite character information (for forming character composites, such as accented letters)

This chapter describes the font metric files and shows the contents and form of the files.

4.1 Font Metric File Names

The font metrics for each font reside in a file with the AFM extension. (AFM stands for *Adobe Font Metrics*.) You can determine the name of a font metric file by taking the contents of the FontName field and changing the hyphens (-) to underscores (_). For example, suppose a FontName field contains the following string:

```
DEC-COURIER-BOLDOBLIQUE
```

The name of its font metric file is:

```
DEC_COURIER_BOLDOBLIQUE.AFM
```

The names of the font metric files for the 29 PostScript fonts in to the LN03R are as follows:

Table 4-1: Font Metric Files

AVANTGARDE_BOOK.AFM
AVANTGARDE_BOOKOBLIQUE.AFM
AVANTEGARDE_DEMI.AFM
AVANTEGARDE_DEMIOBLIQUE.AFM
COURIER.AFM
COURIER_BOLD.AFM
COURIER_OBLIQUE.AFM
COURIER_BOLD OBLIQUE.AFM
HELVETICA.AFM
HELVETICA_BOLD.AFM
HELVETICA_OBLIQUE.AFM
HELVETICA_BOLD OBLIQUE.AFM
LUBALINGRAPH_BOOK.AFM
LUBALINGRAPH_BOOKOBLIQUE.AFM
LUBALINGRAPH_DEMI.AFM
LUBALINGRAPH_DEMIOBLIQUE.AFM
NEWCENTURYSCHLBK_ROMAN.AFM
NEWCENTURYSCHLBK_ITALIC.AFM
NEWCENTURYSCHLBK_BOLD.AFM
NEWCENTURYSCHLBK_BOLDITALIC.AFM
SOUVENIR_DEMI.AFM
SOUVENIR_DEMIITALIC.AFM
SOUVENIR_LIGHT.AFM
SOUVENIR_LIGHTITALIC.AFM
SYMBOL.AFM
TIMES_BOLD.AFM
TIMES_BOLDITALIC.AFM
TIMES_ITALIC.AFM
TIMES_ROMAN.AFM

4.2 Format of Font Metric Files

The AFM file has the following features:

- It is ASCII encoded and human readable. Applications can reformat the information, if necessary. (For example, you can produce \TeX , TFM, or PL files.)
- The file is extensible. You can add nonstandard fields.
- Applications can parse the format, skipping over items that are not applicable.
- A POSTSCRIPT program can generate the format by using the information in a POSTSCRIPT font.

An AFM file consists of four sections that describe different properties of a font. The *global section* contains information on the font. The *character metrics section* contains the metric information for each character. The *kerning data section* contains information about different types of character spacing (kerning). The *composite character section* contains data about characters that are made from other characters.

The following example shows a sample AFM file, illustrating the four sections of the format. An explanation of each section follows.

Example 4-1: Sample Font Metric (AFM) File

```
Comment ***** Global Section ****
StartFontMetrics 2.0
Comment Copyright (c) 1986 Adobe Systems Incorporated. All rights reserved.
FontName Times-Roman
FullName Times Roman
FamilyName Times
Weight Medium
ItalicAngle 0.0
IsFixedPitch false
UnderlinePosition -98
UnderlineThickness 54
Version 001.000

Notice Times is a trademark of Allied Corporation.
EncodingScheme AdobeStandardEncoding
FontBBox -167 -252 1004 904
CapHeight 673
XHeight 445
Descender -219
Ascender 686

Comment ***** Character Metrics Section ****
StartCharMetrics 210
C32 ; WX 250 ; N space ; B 0 0 0 0 ;
C33 ; WX 333 ; N exclam ; B 128 -17 240 673 ;
C34 ; WX 408 ; N quotedbl ; B 46 445 313 685 ;
.
.
Comment---> Lines omitted for brevity
.
.
C37 ; WX 833 ; N e ; B 24 -17 416 469 ;
C37 ; WX 833 ; N f ; B 21 0 382 685 ; L i f i ; L i f l ;
C37 ; WX 833 ; N g ; B 24 -220 469 468 ;
.
.
Comment---> Lines omitted for brevity
.
.
C37 ; WX 833 ; N islash ; B 0 0 283 685 ;
C37 ; WX 833 ; N oslash ; B 30 -104 469 566 ;
C37 ; WX 833 ; N oe ; B 30 -10 684 462 ;
EndCharMetrics
```



```
Comment ***** Kern Data Section ****
StartKernData
StartTrackKern 3
TrackKern -3 6 1 72 -5
TrackKern -2 6 2 72 -3
TrackKern -1 6 7 72 -1
EndTrackKern
StartKernPairs 2
KPX A V -18
KPX A O -10
EndKernPairs
EndKernData

Comment ***** Composite Character Section ****
StartComposites 1
CC Aacute2; PCC A O 0; PCC acute 30 300;
EndComposites
EndFontMetrics
```

4.2.1 Global Section

The following global font keys are the same as those in the **FontInfo** dictionary for any font. The *POSTSCRIPT Language Reference Manual* describes the fonts' meanings. Numeric values are in the character coordinate system (1000 units/em).

NOTE

The following format descriptions use *number* as defined in the *POSTSCRIPT Language Reference Manual*. A number can be a signed integer, a real value, or a radix value.

FontName *string*

string is the name of the font as presented to the POSTSCRIPT **findfont** operator.

FullName *string*

string is the print name of the font.

FamilyName *string*

string is the font family name.

Weight *string*

string is the weight of the font.

ItalicAngle *realnumber*

realnumber is the angle in degrees, counterclockwise from the vertical, of the vertical strokes of the font.

IsFixedPitch *boolean*

boolean value indicates whether the font is monospaced.

UnderlinePosition *number*

number is the distance beneath the baseline at which to place an underline.

UnderlineThickness *number*

number is the thickness of an underline stroke.

Version *string*

string is a font version identifier.

Notice *string*

string is a font name trademark or copyright notice.

Comment *string*

string is a string that an application reading the AFM file should ignore.

EncodingScheme *string*

string indicates the default encoding vector for the font. The most common vector is **AdobeStandardEncoding**. Special fonts can also specify **FontSpecific**.

FontBBox *llx lly urx ury*

The four integers represent the lower left-hand corner and upper right-hand corner of the font bounding box:

llx is an integer value representing the lower left-hand x coordinate.

lly is an integer value representing the lower left-hand y coordinate.

urx is an integer value representing the upper right-hand x coordinate.

ury is an integer value representing the upper right-hand y coordinate.

This bounding box is of flattened paths, not of the Bezier curve descriptions.

CapHeight *number*

number represents the top of an uppercase H.

XHeight *number*

number represents the top of a lowercase x.

Descender *number*

number is a value representing the bottom of a lowercase p.

Ascender *number*

number is a value representing the top of a lowercase d.

4.2.2 Character Metrics Section

The character metrics section of an AFM file begins with the key **StartCharMetrics** and ends with the key **EndCharMetrics**.

Each character's metric information consists of a list of keys and values, separated by semicolons. The characters in the font are in numerically ascending character code. Unencoded characters follow the encoded characters and are identified by character codes of -1. Each character metric must fit on one line.

Key descriptions follow.

StartCharMetrics *integer*

integer indicates the number of characters that the font metric file describes for the font. This integer is valuable for applications to perform a consistency check when parsing the data.

C *number*

number is the decimal value of the default POSTSCRIPT character code (-1 if unencoded).

WX *width-x*

width-x is the character width in the x coordinate. (The value of y is 0.)

W *width-x width-y*

width-x and *width-y* constitute the character width vector.

N *name*

name is the name of the character.

B *llx lly urx ury*

The four integers represent the lower left-hand corner and upper right-hand corner of the character bounding box:

llx represents the lower left-hand x coordinate.

lly represents the lower left-hand y coordinate.

urx represents the upper right-hand x coordinate.

ury represents the upper right-hand y coordinate.

L *successor ligature*

successor is the name of a character that joins with the current character to form a character named *ligature*. Characters may have more than one ligature entry.

4.2.3 Kerning Data Section

The kerning data section is optional; it may or may not be present. The section begins with the key **StartKernData** and ends with **EndKernData**. Two categories of kerning data exist: *track kerning* and *character pair kerning*. One or both sections can be in a font metric file.

To specify track kerning, supply the minimum and maximum point sizes of the characters you are kerning, as well as the minimum and maximum kerning amounts.

The format of keys for kerning data follows.

StartTrackKern *integer*

integer indicates how many tracks are present. This key begins a section that provides track kerning information. The section must have a matching **EndTrackKern** key.

TrackKern *degree min_pt min_kern max_pt max_kern*

degree is an integer. Increasingly negative degrees represent tighter track kerning and increasingly positive degrees represent looser track kerning.

min_pt is the minimum point size for kerning, below which the *min_kern* value is used.

min_kern is the minimum track kerning value in character coordinate units (1000 units/em).

max_pt is the maximum point size for kerning, above which the *max_kern* value is used.

max_kern is the maximum track kerning value in character coordinate units (1000 units/em).

EndTrackKern

This key ends a section that provides track kerning information. It must have a matching **StartTrackKern** key.

StartKernPairs

This key starts the kerning pairs section. The integer following tells how many pairs to expect.

EndKernPairs

This key ends the kerning pairs section.

KP *name1 name2 deltax deltax*

name1 is the first character in the kerning pair.

name2 is the second character in the kerning pair.

deltax is the kerning amount in the x direction.

deltay is the kerning amount in the y direction.

KPX *name1 name2 deltax*

name1 is the first character in the kerning pair.

name2 is the second character in the kerning pair.

deltax is the kerning amount in the x direction (y is 0).

4.2.4 Composite Character Section

The composite character section is optional. Composite characters consist of characters that already exist in a font. Character metrics data for composite characters resides in the Character Metrics Section, along with noncomposite characters.

The composite character section begins with the key **StartComposites** and ends with the key **EndComposites**. The data for each composite character consists of a list of keys and values, separated by semicolons. Each composite character description must fit on one line.

The key description follows:

CC *name integer*

name is the name of the composite character.

integer is the number of parts that comprise the composite character.

PCC *name deltax deltay*

This key composes one of the parts of a composite character.

name is the character that will be used to make up the composite character.

deltax is the displacement from the origin in the x direction.

deltay is the displacement from the origin in the y direction.

All units are in the character coordinate system. There is a **PCC** entry for each part of the composite character.

Optimizing Performance

The performance of the LN03R depends on several factors, including the size and load of the host computer, speed of the communications interface, POSTSCRIPT processing, the image drawing hardware, and the print engine. This chapter describes methods for optimizing the POSTSCRIPT processing.

5.1 Basic Concepts

Three processes account for most of the POSTSCRIPT execution time:

- **Low-level raster manipulations**

This process consists of painting character bitmaps and filling trapezoids located at arbitrary bit boundaries. Usually, this process dominates all others if all the characters are in the font cache.

- **Character scan conversion**

This compute-intensive operation pushes high-level character definitions through the full POSTSCRIPT graphics machinery. This process requires extensive arithmetic, both fixed and floating point.

- **POSTSCRIPT input scanning and interpretation**

This process includes parsing the input stream, constructing tokens, looking up names, and pushing and popping stacks. The time consumed depends on the type of page description and the programming style. Text that consists of strings and calls to simple POSTSCRIPT procedures consumes little time in the interpreter; text that executes complex POSTSCRIPT code for each mark placed on the page more time.

The POSTSCRIPT processor can print 3000 characters a page at a rate of 8 pages a minute, using characters in the bitmap cache. However, since several factors determine this performance, make sure that your POSTSCRIPT code produces high performance.

5.2 Smaller Operator Names

You can replace standard operator names with names that use fewer characters for file-saving and/or communication-saving sake.

5.3 Automatic Spacing

Use the automatic spacing of the **show** operator rather than explicitly positioning every character in your POSTSCRIPT program. Also, use **widthshow** for sequences of words with constant space in between.

5.4 Use of bind Operator

Applying the **bind** operator to procedure bodies relieves the interpreter of looking up the names of POSTSCRIPT operators each time they are executed. This technique works only if the procedure body does not depend upon late-binding.

5.5 Eliminating Unnecessary Procedure Calls

Check for unnecessary nested procedure calls and “unroll” definitions where possible. For instance, the contents of some procedure bodies can be placed in-line.

5.6 Scaling the Coordinate System

Arrange the current transformation matrix so that the `POSTSCRIPT` matrix machinery does most of the arithmetic work. Where appropriate, specify numeric constants as reals rather than integers.

Applications can unnecessarily perform the `findfont`, `makefont` and `setfont` steps twice (once to change the typeface and again to change the size). This indicates an error in the application software's design. Check for double calls when font changes occur.

5.7 `showpage` and `copypage`

Inappropriately using the `copypage` operator can degrade performance in the LN03R.

`showpage` produces better throughput performance. The operator prints the current page, erases it, and initializes the graphics state.

On the other hand, `copypage` prints the page, but does not erase it or initialize the graphics state. `copypage` is more specialized in that it adds new marks to an existing page. You most often use `copypage` to incrementally build up a page.

The `showpage` operator is logically equivalent to

```
copypage erasepage initgraphics
```

Using `copypage` in your `POSTSCRIPT` programs for printing pages can degrade page throughput. `showpage` produces better results because it performs the printing and erasing in parallel, whereas `copypage erasepage` does it serially.

Additionally, you should not use `copypage` to defeat the automatic `initgraphics` performed by `showpage`. That is, to print and erase the current page, but leave the graphics state unchanged, you should enter

```
gsave showpage grestore
```

You should *not* enter

`copypage erasepage`

NOTE

The *POSTSCRIPT Language Tutorial and Cookbook* includes an example that uses the incorrect technique.

5.8 #copies for Multiple Copies

The correct way to print multiple copies of a page is to associate the number of copies with the name **#copies** prior to invoking **showpage** or **copypage**, as described in the *POSTSCRIPT Language Reference Manual*. Executing **showpage** or **copypage** for each copy is inefficient.

5.9 Idle Fonts

You can improve the page composition speed by using the **setidlefonts** extension to scan convert the appropriate fonts. See Section 2.4, as well as the description of **setidlefonts** in Chapter 3, for details.

Programming for the LN03R

System software that controls POSTSCRIPT printers may send POSTSCRIPT device control procedures to the printer ahead of the user's POSTSCRIPT program. This chapter provides information you need to know to write programs that successfully execute in the LN03R.

6.1 Printing Device Control

POSTSCRIPT procedures for device control may be generated by system software that controls a POSTSCRIPT printer. The function of these procedures is determined by the host operating system and printer control software. Some functions performed by device control procedures are:

- Report error conditions to the user
- Report print job accounting information to the user
- Control the portion of a print job that produces output
- Provide custom character set encodings

The device control procedures are sent to the printer ahead of the user's POSTSCRIPT program and terminate when the print job does. These procedures modify the behavior of print job in two ways:

1. By using POSTSCRIPT extension operators
2. By redefining standard POSTSCRIPT operators

6.1.1 Using POSTSCRIPT Extension Operators

POSTSCRIPT extension operators are used to control the printer. These operators are unique to the printer model or class of printers. The POSTSCRIPT extension operators for the LN03R are described in Chapter 3.

To ensure device-independent page descriptions, application software should not refer to the operator extensions in a POSTSCRIPT program. When the extensions must be used, use those that produce the desired output on the largest set of different printer models.

6.1.2 Redefining Standard POSTSCRIPT Operators

Printer control software may redefine standard POSTSCRIPT operators to modify a print job's behavior. This programming technique is called *layering*; it is made possible by the way POSTSCRIPT performs implicit name searches. Layering may also be used by document-formatting utilities that include portions of one POSTSCRIPT program in a second POSTSCRIPT program. See Section 6.2 for more information about programming context layers.

Layering inserts a *context layer* between the user's POSTSCRIPT program and the standard POSTSCRIPT interpreter. The context layer consists of the POSTSCRIPT stacks and the graphics state. The context layer modifies the behavior of the user's POSTSCRIPT program, which is executed (layered) over it. Although the context layer modifies the behavior of the user program, it is transparent to the user program. In fact, the user program may run on top of several different context layers, all equally unaware of each other's existence.

The *current context* is the POSTSCRIPT state resulting from all the layers present when a program runs: user, device control, system management, and standard POSTSCRIPT.

POSTSCRIPT programs that violate the implicit name search order can cause unexpected results, as they override and possibly conflict with underlying context layers.

POSTSCRIPT programs cannot assume the order of objects on the various POSTSCRIPT stacks at the start of a job, because the stack order may have been modified by an underlying context layer.

Start each program with a **save** operator and end each program with a **restore** operator to ensure that you save the context of the program.

6.2 POSTSCRIPT Operators

These guidelines help make programs transportable and help eliminate potential conflict among underlying context layers.

NOTE

Application programs should never exit the server loop.

6.2.1 Device-Dependent Programs

Device-dependent POSTSCRIPT programs can change system parameters. Because these programs perform device control and can exit the server loop, they do not always follow the operator restrictions.

Normally, only system managers should create programs that change system parameters.

6.2.2 Preserving the Context State

POSTSCRIPT programs should, in general, preserve the context state that exists at the start of the program. However, programs should not make assumptions about that context state.

Follow these guidelines to preserve the state of the underlying context layer:

1. Execute a **save** operator at the beginning of the program and a **restore** at the end.

This ensures that the state of virtual memory and composite objects are restored to their previous values when the program completes.

2. Do not alter the contents of a string object that was defined in a previous context layer, because the **restore** operator does not restore the contents of strings. You should not, however, depend on the fact that the **restore** operator does not restore the contents of strings.
3. Do not assume a preexisting stack state at the start of the program.

For example, do not assume an object is on top of a particular stack at the start of the program.

4. Keep the order of stack objects that existed before the start of the program.
5. Do not clear the stack.

6.2.3 Restrictions to Operators

Application software should restrict the use of the following POSTSCRIPT operators in a layered environment, as described:

- **banddevice**

Do not use this device-dependent operator.

- **clear**

This operator clears the entire operand stack, possibly including state that an underlying context layer was trying to preserve. Instead, use **mark** and **cleartomark** or pop the exact number of objects that were pushed.

- **closefile**

Only close files in the module that opened them. Do not close the standard input file, because this closes all instances of the input stream, thus preventing execution of the rest of the file.

- **erasepage**

This operator ignores the clipping path. Use code like the following instead:

```
gsave
clippath
1 setgray
fill
grestore
```

- **framedevice**

Do not use this device-dependent operator.

- **initclip**

Do not assume that the default clipping path is in effect when the program starts. Use **gsave** and **grestore** to preserve the original clipping path.

- **initgraphics**

The current context may have redefined any or all of the graphics state. Use **gsave** and **grestore** to preserve the original graphics state.

- **initmatrix**

The current context may have redefined the current transformation matrix (CTM). Instead of using **initmatrix**, set a variable to the CTM at the start of the included file. For example, use this:

```
/origmatrix matrix currentmatrix def
.
.
origmatrix setmatrix % revert to original CTM
```

Use **gsave** and **grestore** rather than **initmatrix** in such cases.

- **renderbands**

Do not use this device-dependent operator.

- **setmatrix**

Do not make assumptions about the initial CTM. In general, use **concat** instead of **setmatrix**.

- **systemdict**

Do not reference operators through **systemdict**. Use the dictionary stack, in case they have been redefined. Do not use:

```
systemdict /moveto get /m exch def
```

Instead, use:

```
/moveto load /m exch def
```

- **transform**

The **transform** operators (**transform**, **itransform**, **dtransform**, **idtransform**) take an X,Y pair on the stack and “transform” them back and forth between the user space and device space coordinate systems. They are implemented as matrix operations, and if they are used at all should only be used in pairs.

- **userdict**

Do not assume that **userdict** is on top of the dictionary stack. Programs should not reference **userdict** explicitly. Instead, they should create their own local dictionaries. Do not use:

```
/x 1 def
/y 2 def
/z 3 def
```

Instead, use:

```
3 dict begin
/x 1 def
/y 2 def
/z 3 def
.
.
end % pop local dictionary off the dictionary stack
```

- **quit**

In some environments, **quit** may prevent later jobs from being printed. Use **stop** instead.

6.3 Virtual Memory

By default, the LN03R sets the limit of available virtual memory (VM) to the maximum amount, 470K bytes. To set virtual memory lower than the default, use the **setvmlimit** operator. Setting a lower VM limit may be necessary for transporting applications.

To make sure a file does not exceed virtual memory limits, enclose each page with a **save/restore** pair. For programs that require large amounts of VM, such as complex graphics applications, you may need to take other steps to prevent VM limitchecks.

A program can reuse a composite object rather than create a new one each time the object is needed. A program can also use **save** and **restore** frequently. Remember that the **restore** operator not only recovers VM but also restores the graphics state (as for a **grestore**) and restores variables to the values they had at the time of the **save**.

Several POSTSCRIPT operators consume VM each time they are executed. Avoid using these operators within a loop. In this example, the *100 string* allocates VM for a 100-element string at each iteration of the loop:

```
1 1 100
  { myarray exch get 100 string cvs show } for
```


Instead, move the 100 string out of the loop. This example allocates VM for the string once instead of 100 times:

```
/mystring 100 string
1 1 1000
  { myarray exch get mystring cvs show } for
```

The following operators consume VM:

- **[]**
Initialized arrays constructed with bracketed pairs consume VM once an execution. Move the brackets outside of loops.
- **array**
Arrays constructed with the array operator consume VM once an execution.
- **dict**
The **dict** operator consumes VM each time it is executed. However, **dict** is rarely executed from within a loop.
- **matrix**
A matrix is an array of six elements, usually real numbers. Each call to the matrix operator consumes VM.

This example allocates VM for six matrixes, although only two are needed:

```
10 10 matrix scale
90 matrix rotate
matrix concatmatrix
/ctm_1 exch def

5 5 matrix scale
90 matrix rotate
matrix concatmatrix
/ctm_2 exch def
```

The following example consumes less VM. In this case, only three matrixes are allocated in VM, and the scratch matrix can be reused later.

```
/scratch_matrix matrix def
/ctm_1 matrix def
/ctm_2 matrix def

10 10 ctm_1 scale
90 scratch_matrix rotate
ctm_1 concatmatrix
```

```
5 5 ctm_2 scale
90 scratch_matrix rotate
ctm_2 concatmatrix
```

- **string**

The **string** operator consumes VM each time it is executed. When possible, reuse strings and keep the **string** operator outside of loops.

- **()**

A string in parentheses consumes VM only once, at scan time, so it can be left inside loops. However, be careful of modifying such a string. If the string is modified the first time through the loop, the modified version is used in each subsequent iteration.

The following example tries to enter several different names into an entry on a form. It fails, because there is only one copy of the string (Name:_____) in VM.

```
name_array
{
(Name:_____ ) 6
2 -1 roll putinterval
show
12 0 rmoveto
} forall
.
.
.
```

If the first two names entered were Zimmerman and Jones, the form would look something like this:

```
Name:_Zimmerman__ Name:_Jonesrman__
```

To get the desired results, use the **copy** operator to create a new string for each name:

```
name_array
{
(Name:_____ ) 1 copy exch pop 6
2 -1 roll putinterval
show
12 0 rmoveto
} forall
```

- **< >**

The angle bracket construct for entering hexadecimal strings consumes VM only at scan time, not at execution time.

6.4 Changes to Standard POSTSCRIPT Operators on the LN03R

Some standard POSTSCRIPT operators operate differently when used with the LN03R. This section describes changes and restrictions to those standard POSTSCRIPT operators.

6.4.1 Limit to Dictionary Stack Entries

The *POSTSCRIPT Language Reference Manual* states that the maximum number of entries on the dictionary stack (**dict stack**) is 20. However, on the LN03R, POSTSCRIPT uses two entries outside the server loop, so a maximum of 18 entries on **dict stack** is available to user programs.

NOTE

If other libraries also use dictionary stack entries, fewer are available for user programs.

6.4.2 **findfont**

findfont executes the **invalidfont** error when the requested font is not in VM. (In other POSTSCRIPT printers, **findfont** may select a font for you when the requested font is not in VM).

findfont is defined in **systemdict**, not in **statusdict**.

6.4.3 **gsave**

The *POSTSCRIPT Language Reference Manual* states that you can have a maximum of 31 active **gsave** commands before you get a **limitcheck** error. However, the LN03R uses two active **gsave** commands outside the server loop, so the maximum number for programs is 29. Note that if other libraries also use active **gsave**, fewer are available for user programs.

6.4.4 **Polygon Fill**

Using polygon fill (the **fill**, **eofill**, **clip**, and **eoclip** operators) can cause a **limitcheck** error even if fewer than 1500 points are in the path.

6.4.5 quit

The **quit** operator, when executed inside the server loop, is identical in function to the POSTSCRIPT **stop** operator. When executed outside the server loop, **quit** results in the following message:

Flushing: to end of file

If executed in interactive mode, **quit** terminates interactive mode or causes the innermost invocation of **executive** to be exited.

If executed in batch mode, **quit** functions identically to the **stop** operator.

6.4.6 start

The **start** operator implements the behavior of the top-level server control. When invoked, another instance of the POSTSCRIPT server loop is created. When used inside the server loop, **start** generates an *invalidaccess* error. When exited outside the server loop, a power-up reset sequence results.

User programs should not invoke **start**, as it may cause unexpected side effects. For example, the **exitserver** operator would not exit the top-level server loop when invoked, but rather the one that was "created" by the user program upon executing **start**.

6.4.7 vmstatus

For the LN03R, **vmstatus** returns three integers that describe the state of the POSTSCRIPT virtual memory. The values returned are defined as follows:

1. **LEVEL**: The current depth of "save" nesting. It describes the number of "saves" that have not yet been matched with a "restore".
2. **USED**: The number of bytes of virtual memory currently in use.
3. **MAXIMUM**: The maximum number of bytes of virtual memory available to the POSTSCRIPT program. This value is changed by the **setvmlimit** operator, as described in Chapter 3.

6.5 Writing a POSTSCRIPT Software Driver

POSTSCRIPT is a full programming language, but is an *interpreted* language designed for high-level, device-independent page representation. It is not designed for floating-point number processing, or extensive string manipulations. It is best to use the language according to its design. This section lists programming techniques to consider and avoid when writing a POSTSCRIPT software driver.

6.5.1 Useful Techniques

- Write a very clearly defined prolog, which will contain primitive procedures needed by your application, and make it human readable. The application will then generate a “script”, which will represent the document data.
- Use a local dictionary to define the prolog, and use it within procedure calls or when beginning a document. For more information, see the descriptions of **dict**, **begin**, and **end** in the *POSTSCRIPT Language Reference Manual*.
- Put your prolog in a separate data resource or file and/or allow users to supply alternate ones. Allow the POSTSCRIPT output to be redefined to the disk if requested.
- Adhere carefully to Adobe’s document structuring conventions, as outlined in the *POSTSCRIPT Language Reference Manual*.
- Use the **load** operator when you want to use an existing operator in another procedure definition. For example:

```
/s /showpage load def
```

- For changing among fonts, save font dictionaries instead of procedures that execute **findfont**. For example,

```
/music /Sonata findfont 24 scalefont def
```

will process faster than

```
/music { /Sonata findfont 24 scalefont } def
```

- Use the **bind** operator in all prolog procedure definitions. This eliminates unnecessary name lookups and increases POSTSCRIPT’s efficiency. For example:

```
/proc { add 2 div exch round } bind def
```

- Assume that the `POSTSCRIPT` environment you inherit at the beginning of your job is the standard one. Avoid “defensive” programming techniques such as using the `initgraphics`, `grestoreall`, `erasepage`, and `setmatrix` operators.
- Use the `save` and `restore` operators around logical graphic and text objects, and/or complete pages, to reclaim state and memory, and to accomplish font management.

6.5.2 Techniques to Avoid

- Avoid compromising efficiency for generality. Try to minimize the amount of computation performed as the `POSTSCRIPT` file is interpreted. Scale the coordinate system when appropriate. Do not compute line breaks of column justification in `POSTSCRIPT`.
- Avoid interactive dialogs with printers. Use font (`.AFM`) files and printer description (`.APD`) files.
- Avoid system-level dependencies and operators (see Section `op_restrict` and avoid using `exitserver` in simple drivers.
- Avoid setting device-level functions or parameters, especially not the transformation matrix. Use the `concat` operator instead.

6.6 Control Characters

The only characters in the ASCII control set that you can use in a print file are the white space characters recognized by `POSTSCRIPT`. The `LN03R` uses `CTRL/C`, `CTRL/D`, and `CTRL/T` for print job synchronization. These characters should only be used by system software, which controls the `LN03R`. Applications should not include these characters in `POSTSCRIPT` page descriptions.

`POSTSCRIPT` uses the following white space characters as separators. ASCII codes are in decimal.

- Carriage Return (ASCII code 13)
- Horizontal Tab (ASCII code 9)
- Line Feed (ASCII code 10)
- Space (ASCII code 32)

6.7 Printable Characters

ASCII characters that fall between code 33 and 126 comprise the printable characters for `POSTSCRIPT` programs. Using character codes outside these limits may prevent the `POSTSCRIPT` program from being successfully transmitted across a 7-bit communications interface. Use backslash octal notation in strings to represent character codes that fall outside the recommended set.



Appendix A

Fonts

This appendix describes the 29 fonts on the LN03R. These include four type faces of ITC Avant Garde, a fixed-pitch Courier, Helvetica, ITC Lubalin Graph, New Century Schoolbook, ITC Souvenir, and Times. The appendix also includes a Symbol font containing mathematical and special characters. Each font includes sample text and a description of the character set. The sample text is 10-point text with 12-point line spacing.

The following five characters are in all the available fonts, except Symbol. However, none of the character encodings have codes for these characters:

Scaron
scaron
Zcaron
zcaron
trademark

Figure A-1: ITC Avant Garde

10 point text on 12 point linespacing

- AvantGarde-Book** Each single letter is a small, well-balanced figure in itself. There are bad types, too; however, in a good type-face each letter rests complete in itself.
- Romano Guardini
- AvantGarde
-BookOblique* *The contemporary typographer regards his work from the design point of view and concentrates on the true essence of his task, to create graphic design.*
- Emerich Kner
- AvantGarde-Demi** **Machines exist; let us then exploit them to create beauty - a modern beauty, while we are about it. For we live in the twentieth century.**
- Aldous Huxley
- AvantGarde
-DemiOblique* *Neither may the clarity of the single letter be given up for the sake of rhythm, nor may formal beauty be sacrificed to mere clarity or misconceived utility.*
- Walter Tiemann

8 point The graphic signs called letters are so completely blended with the stream

9 point The graphic signs called letters are so completely blended with th

10 point The graphic signs called letters are so completely blended w

11 point The graphic signs called letters are so completely ble

12 point The graphic signs called letters are so completely

14 point The graphic signs called letters are so com

18 point The graphic signs called letters a

24 point The graphic signs called I

Figure A-2: Courier

10 point text on 12 point linespacing

- Courier Each single letter is a small, well-balanced figure in itself.
- Romano Guardini
- Courier-Oblique* *The contemporary typographer regards his work from the design point of view and concentrates on the true essence of his task, to create graphic design.*
- Emerich Kner
- Courier-Bold** **Machines exist; let us then exploit them to create beauty - a modern beauty, while we are about it. For we live in the twentieth century.**
- Aldous Huxley
- Courier -BoldOblique** ***Neither may the clarity of the single letter be given up for the sake of rhythm, nor may formal beauty be sacrificed to mere clarity or misconceived utility.***
- Walter Tiemann

- 8 point The graphic signs called letters are so completely blended w
- 9 point The graphic signs called letters are so completely bl
- 10 point The graphic signs called letters are so complete
- 11 point The graphic signs called letters are so com
- 12 point The graphic signs called letters are so c
- 14 point The graphic signs called letters a
- 18 point The graphic signs called l
- 24 point The graphic signs ca

Figure A-3: Helvetica

10 point text on 12 point linespacing

Helvetica

Each single letter is a small, well-balanced figure in itself. There are bad types, too; however, in a good type-face each letter rests complete in itself.

– Romano Guardini

Helvetica-Oblique

The contemporary typographer regards his work from the design point of view and concentrates on the true essence of his task, to create graphic design.

– Emerich Kner

Helvetica-Bold

Machines exist; let us then exploit them to create beauty – a modern beauty, while we are about it. For we live in the twentieth century.

– Aldous Huxley

***Helvetica
-BoldOblique***

Neither may the clarity of the single letter be given up for the sake of rhythm, nor may formal beauty be sacrificed to mere clarity or misconceived utility.

– Walter Tiemann

8 point The graphic signs called letters are so completely blended with the stream of writ
9 point The graphic signs called letters are so completely blended with the strea
10 point The graphic signs called letters are so completely blended with t
11 point The graphic signs called letters are so completely blended w
12 point The graphic signs called letters are so completely ble
14 point The graphic signs called letters are so comple
18 point The graphic signs called letters are s
24 point The graphic signs called le

Figure A-4: ITC Lubalin Graph

10 point text on 12 point linespacing

- LubalinGraph-Book** Each single letter is a small, well-balanced figure in itself. There are bad types, too; however, in a good type-face each letter rests complete in itself.
- Romano Guardini
- LubalinGraph-BookOblique* *The contemporary typographer regards his work from the design point of view and concentrates on the true essence of his task, to create graphic design.*
- Emerich Kner
- LubalinGraph-Demi** **Machines exist; let us then exploit them to create beauty - a modern beauty, while we are about it. For we live in the twentieth century.**
- Aldous Huxley
- LubalinGraph-DemiOblique* *Neither may the clarity of the single letter be given up for the sake of rhythm, nor may formal beauty be sacrificed to mere clarity or misconceived utility.*
- Walter Tiemann

8 point The graphic signs called letters are so completely blended with the strea
9 point The graphic signs called letters are so completely blended with t
10 point The graphic signs called letters are so completely blended w
11 point The graphic signs called letters are so completely ble
12 point The graphic signs called letters are so completel
14 point The graphic signs called letters are so co
18 point The graphic signs called letters a
24 point The graphic signs called

Figure A-5: New Century Schoolbook

10 point text on 12 point linespacing

NewCenturySchlbk
-Roman

Each single letter is a small, well-balanced figure in itself. There are bad types, too; however, in a good type-face each letter rests complete in itself.

– Romano Guardini

NewCenturySchlbk
-Italic

The contemporary typographer regards his work from the design point of view and concentrates on the true essence of his task, to create graphic design.

– Emerich Kner

NewCenturySchlbk
-Bold

Machines exist; let us then exploit them to create beauty – a modern beauty, while we are about it. For we live in the twentieth century.

– Aldous Huxley

NewCenturySchlbk
-BoldItalic

Neither may the clarity of the single letter be given up for the sake of rhythm, nor may formal beauty be sacrificed to mere clarity or misconceived utility.

– Walter Tiemann

8 point The graphic signs called letters are so completely blended with the stream of w

9 point The graphic signs called letters are so completely blended with the st

10 point The graphic signs called letters are so completely blended wit

11 point The graphic signs called letters are so completely blende

12 point The graphic signs called letters are so completely bl

14 point The graphic signs called letters are so compl

18 point The graphic signs called letters ar

24 point The graphic signs called l

Figure A-6: ITC Souvenir

10 point text on 12 point linespacing

Souvenir-Light

Each single letter is a small, well-balanced figure in itself. There are bad types, too; however, in a good type-face each letter rests complete in itself.

- Romano Guardini

Souvenir-LightItalic

The contemporary typographer regards his work from the design point of view and concentrates on the true essence of his task, to create graphic design.

- Emerich Kner

Souvenir-Demi

Machines exist; let us then exploit them to create beauty - a modern beauty, while we are about it. For we live in the twentieth century.

- Aldous Huxley

Souvenir-DemiItalic

Neither may the clarity of the single letter be given up for the sake of rhythm, nor may formal beauty be sacrificed to mere clarity or misconceived utility.

- Walter Tiemann

8 point The graphic signs called letters are so completely blended with the stream of write

9 point The graphic signs called letters are so completely blended with the stream

10 point The graphic signs called letters are so completely blended with the

11 point The graphic signs called letters are so completely blended wi

12 point The graphic signs called letters are so completely blend

14 point The graphic signs called letters are so complete

18 point The graphic signs called letters are s

24 point The graphic signs called lett

Figure A-7: Times

10 point text on 12 point linespacing

Times-Roman

Each single letter is a small, well-balanced figure in itself. There are bad types, too; however, in a good type-face each letter rests complete in itself.

– Romano Guardini

Times-Italic

The contemporary typographer regards his work from the design point of view and concentrates on the true essence of his task, to create graphic design.

– Emerich Kner

Times-Bold

Machines exist; let us then exploit them to create beauty – a modern beauty, while we are about it. For we live in the twentieth century.

– Aldous Huxley

Times-BoldItalic

Neither may the clarity of the single letter be given up for the sake of rhythm, nor may formal beauty be sacrificed to mere clarity or misconceived utility.

– Walter Tiemann

8 point The graphic signs called letters are so completely blended with the stream of written thou
9 point The graphic signs called letters are so completely blended with the stream of wr
10 point The graphic signs called letters are so completely blended with the strea
11 point The graphic signs called letters are so completely blended with t
12 point The graphic signs called letters are so completely blended w
14 point The graphic signs called letters are so completely b
18 point The graphic signs called letters are so c
24 point The graphic signs called lette

Figure A-7 (Cont.): Times

abcdefghijklmnopqrstuvwxyz 1234567890
 ABCDEFGHIJKLMNOPQRSTUVWXYZ ½¼¾
 áâãäåçéêëèíîïñóôöðøšúûüýÿž [i;²]
 ÁÂÃÄÅÇÉÊËÈÍÎÏÑÓÔÖØŠÚÛÜÝŸŽ
 ìøðþβŁØÐPææfiſlÆŒ(_@&†‡§¶)©®™µ¹²³
 ±+=--- — ¬□\$¢¥£ƒ<.* /||\^#>%~%₀
 «?!»°•<,, “” , ‘ ’ ;:; , ” ’ >... { ~ ~ ~ , ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ }

*abcdefghijklmnopqrstuvwxyz 1234567890
 ABCDEFGHIJKLMNOPQRSTUVWXYZ ½¼¾
 áâãäåçéêëèíîïñóôöðøšúûüýÿž [i;²]
 ÁÂÃÄÅÇÉÊËÈÍÎÏÑÓÔÖØŠÚÛÜÝŸŽ
 ìøðþβŁØÐPææfiſlÆŒ(_@&†‡§¶)©®™µ¹²³
 ±+=--- — ¬□\$¢¥£ƒ<.* /||\^#>%~%₀
 «?!»°•<,, “” , ‘ ’ ;:; , ” ’ >... { ~ ~ ~ , ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ }*

**abcdefghijklmnopqrstuvwxyz 1234567890
 ABCDEFGHIJKLMNOPQRSTUVWXYZ ½¼¾
 áâãäåçéêëèíîïñóôöðøšúûüýÿž [i;²]
 ÁÂÃÄÅÇÉÊËÈÍÎÏÑÓÔÖØŠÚÛÜÝŸŽ
 ìøðþβŁØÐPææfiſlÆŒ(_@&†‡§¶)©®™µ¹²³
 ±+=--- — ¬□\$¢¥£ƒ<.* /||\^#>%~%₀
 «?!»°•<,, “” , ‘ ’ ;:; , ” ’ >... { ~ ~ ~ , ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ }**

*abcdefghijklmnopqrstuvwxyz 1234567890
 ABCDEFGHIJKLMNOPQRSTUVWXYZ ½¼¾
 áâãäåçéêëèíîïñóôöðøšúûüýÿž [i;²]
 ÁÂÃÄÅÇÉÊËÈÍÎÏÑÓÔÖØŠÚÛÜÝŸŽ
 ìøðþβŁØÐPææfiſlÆŒ(_@&†‡§¶)©®™µ¹²³
 ±+=--- — ¬□\$¢¥£ƒ<.* /||\^#>%~%₀
 «?!»°•<,, “” , ‘ ’ ;:; , ” ’ >... { ~ ~ ~ , ^ ~ ~ ~ ~ ~ ~ ~ ~ ~ }*

Figure A-8: Symbol

Sample Symbol uses

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

$$\gamma p \supset \Gamma * P$$

$$\neg(p \vee q) = \neg p \wedge \neg q$$

$$\neg(p \wedge q) = \neg p \vee \neg q$$

$$\varepsilon = \min_{x > 0} (x \mid (1 + x) \neq 1)$$

$$w(\xi' - \xi'') = \sum_{i=1}^m |C_i \cap \{\xi'\}| \cdot |C_i \cap \{\xi''\}| \text{ if } \xi', \xi'' \in L \text{ and } \xi' \neq \xi''$$

$$\bigcup_{i=1}^n Z_i(t) \subseteq M$$

$$Z_i(t) \cap Z_j(t) = \emptyset \quad (i \neq j)$$

proposition	true if and only if
$(\forall u) s(p)$	$S \cap T_p' = \emptyset$
$(\exists u) s(p)$	$S \cap T_p \neq \emptyset$
$(\forall u) s(\sim p)$	$S \cap T_p = \emptyset$
$(\exists u) s(\sim p)$	$S \cap T_p' \neq \emptyset$
$\neg((\forall u) s(p))$	$S \cap T_p' \neq \emptyset$
$\neg((\exists u) s(p))$	$S \cap T_p = \emptyset$

$$kp^{k/2} t^{-1} I_k(at) \Leftrightarrow \left| \frac{s + \sqrt{s^2 - 4\lambda\mu}}{2\lambda} \right|^{-x}$$

Figure A-8 (Cont.): Symbol

12 point

αβχδεφγηικλμνοπθρστυωξψζφωθς ∫ { ([] } [[]] (())
ΑΒΧΔΕΦΓΗΙΚΛΜΝΟΠΘΡΣΤΥΩΞΨΖΥ Ζ } } | | | |
= ≠ ≡ ~ ≈ ≅ < > ≤ ≥ ∧ ∨ ∴ - + ± × ÷ · ∙ ∞ ∫ ∪ ∩ ∪
∀ ∃ ∂ ∩ ∪ ∩ ∩ ⊆ ⊂ ⊃ ⊄ ∈ ∉ ∅ ⊗ ⊕ ∞
↔ ← ↑ → ↓ ⇔ ⇐ ⇑ ⇒ ⇓ ° ' " ∟ ∞ ∑ ∏ ∫
... < ◇ ® © ™ ® © ™ ∑ ∏ > ∠ ⊥ ∇ • ♣ ♦ ♥ ♠
! # % & * , . 0 1 2 3 4 5 6 7 8 9 : ; ? f √ _ | | - / / -

16 point

αβχδεφγηικλμνοπθρστυωξψζφωθς
ΑΒΧΔΕΦΓΗΙΚΛΜΝΟΠΘΡΣΤΥΩΞΨΖΥ
= ≠ ≡ ~ ≈ ≅ < > ≤ ≥ ∧ ∨ ∴ - + ± × ÷ · ∙ ∞
∀ ∃ ∂ ∩ ∪ ∩ ∩ ⊆ ⊂ ⊃ ⊄ ∈ ∉ ∅ ⊗ ⊕ ∞
↔ ← ↑ → ↓ ⇔ ⇐ ⇑ ⇒ ⇓ ° ' " ∟ ∞ ∑ ∏ ∫
... < ◇ ® © ™ ® © ™ ∑ ∏ > ∠ ⊥ ∇ • ♣ ♦ ♥ ♠
! # % & * , . 0 1 2 3 4 5 6 7 8 9 : ; ? f √ _ | | - / / -
∫ { ([] } [[]] (())
| { } | | | |
∫ ∪ ∩ ∪

Encoding Vectors

The LN03R includes four encoding vectors: Adobe Standard Encoding, Symbol Encoding (used only for the Symbol font), ISO Latin1 Encoding, and DEC MCS Encoding. When you use the extended **findfont** operator, which is part of the device control library, these encodings are available.

This appendix describes the encoding vectors on the LN03R and lists the characters in each encoding vector in numerical and alphabetical order.

ISO Latin1 Encoding Vector

The ISO Latin Alphabet Number 1 standard (ISO 8859/1) and the 8-bit ASCII standard (ANSI X3.4-1977 and X3.134.2) both specify the character set for text and data processing in the languages of western Europe and the western hemisphere.

The character codes octal 040 through 176 in the ISO Latin1 Encoding vector and Adobe Standard Encoding vector conform to the American National Standard Code for Information Interchange (ASCII), X3.4-1977. The ASCII Standard permits several choices between coded characters that look alike, which is especially useful in monospaced fonts. For proportionally-spaced fonts, you must choose. For example, the ASCII and ISO standards permit octal 055 to be either *hyphen*, *minus sign*, or both. Also, the ASCII standard permits octal 140 to be *left single quotation mark*, *grave accent*, or both and permits octal 047 to be *apostrophe*, *right single quotation mark*, *acute accent*, or any combination.

Character codes octal 040 through 176 for the ISO Latin1 Encoding vector are identical to the Adobe Standard Encoding vector, except for character code 055, which is *hyphen* in the Standard Encoding vector and is *minus sign* in the ISO Latin1 Encoding vector. ISO Latin1 Encoding has *hyphen* at code table position 255 octal. Having both *hyphen* and *minus* characters (with different coding) is more useful than having two hyphen characters.

The character codes octal 240 through 377 in the ISO Latin1 Encoding vector conform to the ANSI and ISO standards. The ANSI and ISO standards are for "final form" interchange with character imaging devices, such as POSTSCRIPT, where the data is not to be reformatted after interchange. The standards are also for "revisable form" interchange with text

processing systems, where the data may or may not be reformatted after interchange.

The characters at octal 240 and 255 are *non-breaking space* and *soft hyphen*, respectively, which have identical presentation, as *space* (octal 40), and *hyphen*. These are for revisable interchange. Therefore, for final form interchange, such as POSTSCRIPT, octal 240 is defined as *space* (the same as octal 40), and octal 255 is defined as *hyphen*.

Note that dotlessi and accents (octal 220 to 237) are not part of the ISO Latin Alphabet Number 1 or the 8-bit ASCII standards. They are encoded in the ISO Latin1 Encoding vector because POSTSCRIPT requires that accents and component characters used by encoded composite characters also occur somewhere in the encoding vector. In other ANSI and ISO standards, octal 220 to 237 are assigned to control functions, not graphic characters. However, POSTSCRIPT does not include these control functions.

Figure B-1: Adobe Standard Encoding Vector

<i>octal</i>	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		!	"	#	\$	%	&	'
\05x	()	*	+	,	-	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	@	A	B	C	D	E	F	G
\11x	H	I	J	K	L	M	N	O
\12x	P	Q	R	S	T	U	V	W
\13x	X	Y	Z	[\]	^	_
\14x	'	a	b	c	d	e	f	g
\15x	h	i	j	k	l	m	n	o
\16x	p	q	r	s	t	u	v	w
\17x	x	y	z	{		}	~	
\20x								
\21x								
\22x								
\23x								
\24x		ı	ø	£	/	¥	f	§
\25x	□	'	“	«	<	>	fi	fl
\26x		-	†	‡	.		¶	•
\27x	,	„	”	»	...	%		ı
\30x		`	´	^	-	-	˘	˙
\31x	¨		°	˚		˛	˜	˜
\32x	—							
\33x								
\34x		Æ		ı				
\35x	Ł	Ø	Œ	º				
\36x		æ				ı		
\37x	ı	ø	œ	ß				

Figure B-2: Symbol Encoding Vector

<i>octal</i>	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		!	∀	#	∃	%	&	ə
\05x	()	*	+	,	-	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	≡	A	B	X	Δ	E	Φ	Γ
\11x	H	I	ϑ	K	Λ	M	N	O
\12x	Π	Θ	P	Σ	T	Υ	ς	Ω
\13x	Ξ	Ψ	Z	[∴]	⊥	—
\14x		α	β	χ	δ	ε	φ	γ
\15x	η	ι	φ	κ	λ	μ	ν	ο
\16x	π	θ	ρ	σ	τ	υ	ϖ	ω
\17x	ξ	ψ	ζ	{		}	~	
\20x								
\21x								
\22x								
\23x								
\24x		Υ	'	≤	/	∞	f	⊕
\25x	♦	♥	♠	↔	←	↑	→	↓
\26x	◦	±	"	≥	x	∞	∂	•
\27x	÷	≠	≡	≈	...		—	└
\30x	ℵ	ℑ	℔	∅	⊗	⊕	∅	∩
\31x	∪	⊃	⊇	⊂	⊆	⊆	∈	∉
\32x	∠	∇	⊗	⊙	™	∏	√	·
\33x	¬	∧	∨	↔	⇐	↑	⇒	↓
\34x	◇	⟨	⊗	⊙	™	Σ	(
\35x	({	(
\36x)				})	
\37x)					})	

Figure B-3: ISO Latin1 Encoding Vector

<i>octal</i>	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		!	"	#	\$	%	&	'
\05x	()	*	+	,	-	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	@	A	B	C	D	E	F	G
\11x	H	I	J	K	L	M	N	O
\12x	P	Q	R	S	T	U	V	W
\13x	X	Y	Z	[\]	^	_
\14x	`	a	b	c	d	e	f	g
\15x	h	i	j	k	l	m	n	o
\16x	p	q	r	s	t	u	v	w
\17x	x	y	z	{		}	~	
\20x								
\21x								
\22x	¡	¢	£	¤	¥	¦	§	¨
\23x	©	ª	«	¬	®	¯	°	±
\24x	°	±	²	³	´	µ	¶	·
\27x	¸	¹	º	»	¼	½	¾	¿
\30x	À	Á	Â	Ã	Ä	Å	Æ	Ç
\31x	È	É	Ê	Ë	Ì	Í	Î	Ï
\32x	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×
\33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
\34x	à	á	â	ã	ä	å	æ	ç
\35x	è	é	ê	ë	ì	í	î	ï
\36x	ð	ñ	ò	ó	ô	õ	ö	÷
\37x	ø	ù	ú	û	ü	ý	þ	ÿ

DEC MCS (DIGITAL Multinational Character Set) Encoding Vector

Character codes octal 040 through 176 (the ASCII character set) of the DEC MCS Encoding vector are the same as the ISO Latin1 Encoding vector and, therefore, are the same as the Adobe Standard Encoding, except for code table position octal 055. Position octal 055 is minus in ISO Latin1 Encoding and DEC MCS Encoding, but it is hyphen in Adobe Standard Encoding.

Note that dotlessi and accents (octal 220 to 237) are not part of the ISO Latin Alphabet Number 1 or the 8-bit ASCII standards. They are encoded because `POSTSCRIPT` requires that all accents and component characters used by encoded composite characters also occur somewhere in the encoding vector. In ANSI and ISO standards, octal 220 to 237 are assigned to control functions, not graphic characters. However, `POSTSCRIPT` does not include these control functions.

Figure B-4: DEC MCS (DIGITAL Multinational Character Set) Encoding Vector

<i>octal</i>	0	1	2	3	4	5	6	7
\00x								
\01x								
\02x								
\03x								
\04x		!	"	#	\$	%	&	'
\05x	()	*	+	,	-	.	/
\06x	0	1	2	3	4	5	6	7
\07x	8	9	:	;	<	=	>	?
\10x	@	A	B	C	D	E	F	G
\11x	H	I	J	K	L	M	N	O
\12x	P	Q	R	S	T	U	V	W
\13x	X	Y	Z	[\]	^	_
\14x	'	a	b	c	d	e	f	g
\15x	h	i	j	k	l	m	n	o
\16x	p	q	r	s	t	u	v	w
\17x	x	y	z	{		}	~	
\20x								
\21x								
\22x	ı	ˆ	˜	˘	˙	˚	˛	˜
\23x	¨		•	ˆ		˘	˙	˚
\24x		ı	¢	£		¥		§
\25x	□	©	ª	«				
\26x	°	±	²	³		µ	¶	·
\27x		ı	º	»	¼	½		¿
\30x	À	Á	Â	Ã	Ä	Å	Æ	Ç
\31x	È	É	Ê	Ë	Ì	Í	Î	Ï
\32x		Ñ	Ò	Ó	Ô	Õ	Ö	Ø
\33x	∅	Ù	Ú	Û	Ü	Ý		ß
\34x	à	á	â	ã	ä	å	æ	ç
\35x	è	é	ê	ë	ì	í	î	ï
\36x		ñ	ò	ó	ô	õ	ö	ø
\37x	ø	ù	ú	û	ü	ý		

Table B-1: Characters Listed in the Order of Adobe Standard Encoding

<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>
040		space	136	^	asciicircum	271	„	quotedblbase
041	!	exclam	137	_	underscore	272	”	quotedblright
042	"	quotedbl	140	`	quoteleft	273	»	guillemotright
043	#	numbersign	141	a	a	274	…	ellipsis
044	\$	dollar	…	…	…	275	‰	perthousand
045	%	percent	172	z	z	277	¿	questiondown
046	&	ampersand	173	{	braceleft	301	˘	grave
047	'	quoteright	174		bar	302	˙	acute
050	(parenleft	175	}	braceright	303	ˆ	circumflex
051)	parenright	176	~	asciitilde	304	˜	tilde
052	*	asterisk	241	¡	exclamdown	305	˘	macron
053	+	plus	242	¢	cent	306	˘	breve
054	,	comma	243	£	sterling	307	¨	dotaccent
055	-	hyphen	244	/	fraction	310	¨	dieresis
056	.	period	245	¥	yen	312	•	ring
057	/	slash	246	f	florin	313	˘	cedilla
060	0	zero	247	§	section	315	˘	hungarumlaut
…	…	…	250	¤	currency	316	˘	ogonek
071	9	nine	251	'	quotesingle	317	˘	caron
072	:	colon	252	“	quotedblleft	320	—	emdash
073	;	semicolon	253	«	guillemotleft	341	Æ	AE
074	<	less	254	‹	guilsingleft	343	ª	ordfeminine
075	=	equal	255	›	guilsingright	350	Ł	Lslash
076	>	greater	256	fi	fi	351	Ø	Oslash
077	?	question	257	fl	fl	352	Œ	OE
100	@	at	261	–	endash	353	º	ordmasculine
101	A	A	262	†	dagger	361	æ	ae
…	…	…	263	‡	daggerdbl	365	ı	dotlessi
132	Z	Z	264	·	periodcentered	370	ı	lslash
133	[bracketleft	266	¶	paragraph	371	ø	oslash
134	\	backslash	267	•	bullet	372	œ	oe
135]	bracketright	270	,	quotesingbase	373	ß	germandbls

Table B-2: Characters Listed in the Order of Symbol Encoding

<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>
040		space	106	Φ	Phi	144	δ	delta
041	!	exclam	107	Γ	Gamma	145	ε	epsilon
042	∇	universal	110	H	Eta	146	φ	phi
043	#	numbersign	111	I	Iota	147	γ	gamma
044	∃	existential	112	ϑ	theta l	150	η	eta
045	%	percent	113	K	Kappa	151	ι	iota
046	&	ampersand	114	Λ	Lambda	152	φ	phil
047	ε	suchthat	115	M	Mu	153	κ	kappa
050	(parenleft	116	N	Nu	154	λ	lambda
051)	parenright	117	O	Omicron	155	μ	mu
052	*	asteriskmath	120	Π	Pi	156	ν	nu
053	+	plus	121	Θ	Theta	157	ο	omicron
054	,	comma	122	P	Rho	160	π	pi
055	-	minus	123	Σ	Sigma	161	θ	theta
056	.	period	124	T	Tau	162	ρ	rho
057	/	slash	125	Υ	Upsilon	163	σ	sigma
060	0	zero	126	ς	sigma l	164	τ	tau
...			127	Ω	Omega	165	υ	upsilon
071	9	nine	130	Ξ	Xi	166	ω	omega l
072	:	colon	131	Ψ	Psi	167	ω	omega
073	;	semicolon	132	Z	Zeta	170	ξ	xi
074	<	less	133	[bracketleft	171	ψ	psi
075	=	equal	134	∴	therefore	172	ζ	zeta
076	>	greater	135]	bracketright	173	{	braceleft
077	?	question	136	⊥	perpendicular	174		bar
100	≡	congruent	137	—	underscore	175	}	braceright
101	A	Alpha	140	√	radicalex	176	~	similar
102	B	Beta	141	α	alpha	241	Υ	Upsilon1
103	X	Chi	142	β	beta	242	'	minute
104	Δ	Delta	143	χ	chi	243	≤	lessequal
105	E	Epsilon						

Table B-2 (Cont.): Characters Listed in the Order of Symbol Encoding

<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>
244	/	fraction	302	℞	Rfraktur	340	◇	lozenge
245	∞	infinity	303	∅	weierstrass	341	⟨	angleleft
246	f	florin	304	⊗	circlemultiply	342	Ⓔ	registersans
247	♣	club	305	⊕	circleplus	343	©	copyrightsans
250	♦	diamond	306	∅	emptyset	344	™	trademarksans
251	♥	heart	307	∩	intersection	345	∑	summation
252	♠	spade	310	∪	union	346	{	parenlefttp
253	↔	arrowboth	311	⊃	proversuperset	347		parenleftex
254	←	arrowleft	312	⊇	reflexsuperset	350		parenleftbt
255	↑	arrowup	313	⊈	notsubset	351		bracketlefttp
256	→	arrowright	314	⊂	propersubset	352		bracketleftex
257	↓	arrowdown	315	⊆	reflexsubset	353		bracketleftbt
260	°	degree	316	∈	element	354	{	bracelefttp
261	±	plusminus	317	∉	notelement	355		braceleftmid
262	″	second	320	∠	angle	356		braceleftbt
263	≥	greaterequal	321	∇	gradient	357		braceex
264	×	multiply	322	®	registerserif	361	}	angleright
265	∝	proportional	323	©	copyrightserif	362		integral
266	∂	partialdiff	324	™	trademarkserif	363	{	integraltp
267	•	bullet	325	∏	product	364		integralex
270	÷	divide	326	√	radical	365		integralbt
271	≠	notequal	327	·	dotmath	366	}	parenrighttp
272	≡	equivalence	330	¬	logicalnot	367		parenrightex
273	≈	approxequal	331	∧	logicaland	370		parenrightbt
274	…	ellipsis	332	∨	logicalor	371		bracketrighttp
275		arrowvertex	333	↔	arrowdblboth	372		bracketrightex
276	—	arrowhorizex	334	⇐	arrowdblleft	373		bracketrightbt
277	↵	carriagereturn	335	⇑	arrowdblup	374		bracerighttp
300	ℵ	aleph	336	⇒	arrowdblright	375	}	bracerightmid
301	Œ	Ifraktur	337	⇓	arrowdbldown	376		bracerightbt

Table B-3: Characters Listed in the Order of ISO Latin1 Encoding

octal	char	name	octal	char	name	octal	char	name
040		space	101	A	A	233	¸	cedilla
041	!	exclam	...			235	ˆ	hungarumlaut
042	"	quotedbl	132	Z	Z	236	˘	ogonek
043	#	numbersign	133	[bracketleft	237	ˇ	caron
044	\$	dollar	134	\	backslash	240		space
045	%	percent	135]	bracketright	241	¡	exclamdown
046	&	ampersand	136	^	asciicircum	242	¢	cent
047	'	quoteright	137	_	underscore	243	£	sterling
050	(parenleft	140	`	quoteleft	244	¤	currency
051)	parenright	141	a	a	245	¥	yen
052	*	asterisk	...			246	¦	brokenbar
053	+	plus	172	z	z	247	§	section
054	,	comma	173	{	braceleft	250	¨	dieresis
055	-	minus	174		bar	251	©	copyright
056	.	period	175	}	braceright	252	ª	ordfeminine
057	/	slash	176	~	asciitilde	253	«	guillemotleft
060	0	zero	220	ı	dotlessi	254	¬	logicalnot
...			221	`	grave	255	-	hyphen
071	9	nine	222	´	acute	256	®	registered
072	:	colon	223	^	circumflex	257	ˉ	macron
073	;	semicolon	224	~	tilde	260	°	degree
074	<	less	225	˘	macron	261	±	plusminus
075	=	equal	226	˘	breve	262	²	twosuperior
076	>	greater	227	˙	dotaccent	263	³	threesuperior
077	?	question	230	¨	dieresis	264	´	acute
100	@	at	232	•	ring			

Table B-3 (Cont.): Characters Listed in the Order of ISO Latin1 Encoding

<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>
265	µ	mu	316	î	Icircumflex	347	ç	cedilla
266	¶	paragraph	317	ï	Idieresis	350	è	egrave
267	·	periodcentered	320	Ð	Eth	351	é	eacute
270	.	cedilla	321	Ñ	Ntilde	352	ê	ecircumflex
271	ı	onesuperior	322	Ò	Ograve	353	ë	edieresis
272	º	ordmasculine	323	Ó	Oacute	354	ì	igrave
273	»	guillemotright	324	Ô	Ocircumflex	355	í	iacute
274	¼	onequarter	325	Õ	Otilde	356	î	icircumflex
275	½	onehalf	326	Ö	Odieresis	357	ï	idieresis
276	¾	threequarters	327	×	multiply	360	ð	eth
277	¿	questiondown	330	Ø	Oslash	361	ñ	ntilde
300	À	Agrave	331	Ù	Ugrave	362	ò	ograve
301	Á	Aacute	332	Ú	Uacute	363	ó	oacute
302	Â	Acircumflex	333	Û	Ucircumflex	364	ô	ocircumflex
303	Ã	Atilde	334	Ü	Udieresis	365	õ	otilde
304	Ä	Adieresis	335	Ý	Yacute	366	ö	odieresis
305	Å	Aring	336	Þ	Thorn	367	÷	divide
306	Æ	AE	337	ß	germandbls	370	ø	oslash
307	Ç	Ccedilla	340	à	agrave	371	ù	ugrave
310	È	Egrave	341	á	aacute	372	ú	uacute
311	É	Eacute	342	â	acircumflex	373	û	ucircumflex
312	Ê	Ecircumflex	343	ã	atilde	374	ü	udieresis
313	Ë	Edieresis	344	ä	adieresis	375	ý	yacute
314	Ì	Igrave	345	å	aring	376	þ	thorn
315	Í	Iacute	346	æ	ae	377	ÿ	ydieresis

Table B-4: Characters Listed in the Order of DEC MCS Encoding

<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>
040		space	076	>	greater	224	~	tilde
041	!	exclam	077	?	question	225	˘	macron
042	"	quotedbl	100	@	at	226	˘	breve
043	#	numbersign	101	A	A	227	˙	dotaccent
044	\$	dollar	...			230	¨	dieresis
045	%	percent	132	Z	Z	232	˚	ring
046	&	ampersand	133	[bracketleft	233	¸	cedilla
047	'	quoteright	134	\	backslash	235	˝	hungarumlaut
050	(parenleft	135]	bracketright	236	ˇ	ogonek
051)	parenright	136	^	asciicircum	237	˘	caron
052	*	asterisk	137	_	underscore	241	¡	exclamdown
053	+	plus	140	‘	quoteleft	242	¢	cent
054	,	comma	141	a	a	243	£	sterling
055	-	minus	...			245	¥	yen
056	.	period	172	z	z	247	§	section
057	/	slash	173	{	braceleft	250	¤	currency
060	0	zero	174		bar	251	©	copyright
...			175	}	braceright	252	ª	ordfeminine
071	9	nine	176	~	asciitilde	253	«	guillemotleft
072	:	colon	220	ı	dotlessi	260	°	degree
073	;	semicolon	221	`	grave	261	±	plusminus
074	<	less	222	´	acute	262	²	twosuperior
075	=	equal	223	ˆ	circumflex	263	³	threesuperior

Table B-4 (Cont.): Characters Listed in the Order of DEC MCS Encoding

<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>	<i>octal</i>	<i>char</i>	<i>name</i>
265	µ	mu	316	î	Icircumflex	346	æ	ae
266	¶	paragraph	317	ÿ	Idieresis	347	ç	ccedilla
267	·	periodcentered	321	Ñ	Ntilde	350	è	egrave
271	¹	onesuperior	322	Ò	Ograve	351	é	eacute
272	²	ordmasculine	323	Ó	Oacute	352	ê	ecircumflex
273	»	guillemotright	324	Ô	Ocircumflex	353	ë	edieresis
274	¼	onequarter	325	Õ	Otilde	354	ì	igrave
275	½	onehalf	326	Ö	Odieresis	355	í	iacute
277	¿	questiondown	327	Œ	OE	356	î	icircumflex
300	À	Agrave	330	Ø	Oslash	357	ï	idieresis
301	Á	Aacute	331	Ù	Ugrave	361	ñ	ntilde
302	Â	Acircumflex	332	Ú	Uacute	362	ò	ograve
303	Ã	Atilde	333	Û	Ucircumflex	363	ó	oacute
304	Ä	Adieresis	334	Ü	Udieresis	364	ô	ocircumflex
305	Å	Aring	335	Ý	Ydieresis	365	õ	otilde
306	Æ	AE	337	ß	germandbls	366	ö	odieresis
307	Ç	Coedilla	340	à	agrave	367	œ	oe
310	È	Egrave	341	á	aacute	370	ø	oslash
311	É	Eacute	342	â	acircumflex	371	ù	ugrave
312	Ê	Ecircumflex	343	ã	atilde	372	ú	uacute
313	Ë	Edieresis	344	ä	adieresis	373	û	ucircumflex
314	Ì	Igrave	345	å	aring	374	ü	udieresis
315	Í	Iacute	346	æ	ae	375	ÿ	ydieresis

Table B-5: Characters Listed in Alphabetical Order

name	character	encoding			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
A	A	101	—	101	101
Æ	Æ	341	—	306	306
Aacute	Á	—	—	301	301
Acircumflex	Â	—	—	302	302
Adieresis	Ä	—	—	304	304
Agrave	À	—	—	300	300
Alpha	Α	—	101	—	—
Aring	Å	—	—	305	305
Atilde	Ã	—	—	303	303
B	B	102	—	102	102
Beta	Β	—	102	—	—
C	C	103	—	103	103
Ccedilla	Ç	—	—	307	307
Chi	Χ	—	103	—	—
D	D	104	—	104	104
Delta	Δ	—	104	—	—
E	E	105	—	105	105
Eacute	É	—	—	311	311
Ecircumflex	Ê	—	—	312	312
Edieresis	Ë	—	—	313	313
Egrave	È	—	—	310	310
Epsilon	Ε	—	105	—	—
Eta	Η	—	110	—	—
Eth	Ð	—	—	320	—
F	F	106	—	106	106
G	G	107	—	107	107
Gamma	Γ	—	107	—	—

Table B-5 (Cont.): Characters Listed in Alphabetical Order

<i>name</i>	<i>character</i>	<i>encoding</i>			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
H	H	110	—	110	110
I	I	111	—	111	111
Iacute	Í	—	—	315	315
Icircumflex	Î	—	—	316	316
Idieresis	Ï	—	—	317	317
Ifraktur	Œ	—	301	—	—
Igrave	Ì	—	—	314	314
Iota	Ι	—	111	—	—
J	J	112	—	112	112
K	K	113	—	113	113
Kappa	Κ	—	113	—	—
L	L	114	—	114	114
Lambda	Λ	—	114	—	—
Lslash	Ł	350	—	—	—
M	M	115	—	115	115
Mu	Μ	—	115	—	—
N	N	116	—	116	116
Ntilde	Ñ	—	—	321	321
Nu	Ν	—	116	—	—
O	O	117	—	117	117
OE	Œ	352	—	—	327
Oacute	Ó	—	—	323	323
Ocircumflex	Ô	—	—	324	324
Odieresis	Û	—	—	326	326
Ograve	Ò	—	—	322	322
Omega	Ω	—	127	—	—
Omicron	Ο	—	117	—	—

Table B-5 (Cont.): Characters Listed in Alphabetical Order

<i>name</i>	<i>character</i>	<i>encoding</i>			
		Adobe Standard	Symbol	ISOLatin1	DEC MCS
Oslash	Ø	351	—	330	330
Otilde	Õ	—	—	325	325
P	Ɔ	120	—	120	120
Phi	Φ	—	106	—	—
Pi	Π	—	120	—	—
Psi	Ψ	—	131	—	—
Q	Ɔ	121	—	121	121
R	Ɔ	122	—	122	122
Rfraktur	Œ	—	302	—	—
Rho	Ɔ	—	122	—	—
S	Ɔ	123	—	123	123
Scaron	Š	—	—	—	—
Sigma	Σ	—	123	—	—
T	Ɔ	124	—	124	124
Tau	Ɔ	—	124	—	—
Theta	Θ	—	121	—	—
Thorn	Ɔ	—	—	336	—
U	Ɔ	125	—	125	125
Uacute	Ú	—	—	332	332
Ucircumflex	Û	—	—	333	333
Udieresis	Û	—	—	334	334
Ugrave	Ù	—	—	331	331
Upsilon	Υ	—	125	—	—
Upsilon1	Υ	—	241	—	—
V	Ɔ	126	—	126	126
W	Ɔ	127	—	127	127
X	Ɔ	130	—	130	130

Table B-5 (Cont.): Characters Listed in Alphabetical Order

name	character	encoding			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
Xi	Ξ	—	130	—	—
Y	Y	131	—	131	131
Yacute	Ý	—	—	335	—
Ydieresis	ÿ	—	—	—	335
Z	Z	132	—	132	132
Zcaron	Ž	—	—	—	—
Zeta	Ζ	—	132	—	—
a	a	141	—	141	141
aacute	á	—	—	341	341
acircumflex	â	—	—	342	342
acute	´	302	—	222	222
adieresis	ä	—	—	344	344
ae	æ	361	—	346	346
agrave	à	—	—	340	340
aleph	ℵ	—	300	—	—
alpha	α	—	141	—	—
ampersand	&	046	046	046	046
angle	∠	—	320	—	—
angleleft	∠	—	341	—	—
angleright	∠	—	361	—	—
approxequal	≈	—	273	—	—
aring	å	—	—	345	345
arrowboth	↔	—	253	—	—
arrowdblboth	↔	—	333	—	—
arrowdbldown	⇓	—	337	—	—
arrowdblleft	⇐	—	334	—	—
arrowdblright	⇒	—	336	—	—

Table B-5 (Cont.): Characters Listed in Alphabetical Order

<i>name</i>	<i>character</i>	<i>encoding</i>			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
arrowdblup	↑↑	—	335	—	—
arrowdown	↓	—	257	—	—
arrowhorizex	—	—	276	—	—
arrowleft	←	—	254	—	—
arrowright	→	—	256	—	—
arrowup	↑	—	255	—	—
arrowvertex		—	275	—	—
asciicircum	^	136	—	136	136
asciitilde	~	176	—	176	176
asterisk	*	052	—	052	052
asteriskmath	*	—	052	—	—
at	@	100	—	100	100
atilde	ã	—	—	343	343
b	b	142	—	142	142
backslash	\	134	—	134	134
bar		174	174	174	174
beta	β	—	142	—	—
braceex		—	357	—	—
braceleft	{	173	173	173	173
braceleftbt	{	—	356	—	—
braceleftmid	{	—	355	—	—
bracelefttp	{	—	354	—	—
braceright	}	175	175	175	175
bracerightbt	}	—	376	—	—
bracerightmid	}	—	375	—	—
bracerighttp	}	—	374	—	—
bracketleft	[133	133	133	133

Table B-5 (Cont.): Characters Listed in Alphabetical Order

<i>name</i>	<i>character</i>	<i>encoding</i>			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
bracketleftbt	⌊	—	353	—	—
bracketleftex	⌋	—	352	—	—
bracketlefttp	⌈	—	351	—	—
bracketright	⌋	135	135	135	135
bracketrightbt	⌌	—	373	—	—
bracketrightex	⌍	—	372	—	—
bracketrighttp	⌎	—	371	—	—
breve	˘	306	—	226	226
brokenbar	⌚	—	—	246	—
bullet	•	267	267	—	—
c	ç	143	—	143	143
caron	ˇ	317	—	237	237
carriagereturn	␣	—	277	—	—
cedilla	ç	—	—	347	347
cedilla	¸	313	—	233	233
cent	¢	242	—	242	242
chi	χ	—	143	—	—
circlemultiply	⊗	—	304	—	—
circleplus	⊕	—	305	—	—
circumflex	ˆ	303	—	223	223
club	♣	—	247	—	—
colon	:	072	072	072	072
comma	,	054	054	054	054
congruent	≡	—	100	—	—
copyright	©	—	—	251	251
copyrightsans	©	—	343	—	—
copyrightserif	©	—	323	—	—

Table B-5 (Cont.): Characters Listed in Alphabetical Order

<i>name</i>	<i>character</i>	<i>encoding</i>			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
currency	¤	250	—	244	250
d	d	144	—	144	144
dagger	†	262	—	—	—
daggerdbl	‡	263	—	—	—
degree	°	—	260	260	260
delta	δ	—	144	—	—
diamond	◆	—	250	—	—
dieresis	¨	310	—	230	230
divide	÷	—	270	367	—
dollar	\$	044	—	044	044
dotaccent	·	307	—	227	227
dotlessi	ı	365	—	220	220
dotmath	·	—	327	—	—
e	e	145	—	145	145
eacute	é	—	—	351	351
ecircumflex	ê	—	—	352	352
edieresis	ë	—	—	353	353
egrave	è	—	—	350	350
eight	8	070	070	070	070
element	€	—	316	—	—
ellipsis	...	274	274	—	—
emdash	—	320	—	—	—
emptyset	∅	—	306	—	—
endash	–	261	—	—	—
epsilon	ε	—	145	—	—
equal	=	075	075	075	075
equivalence	≡	—	272	—	—

Table B-5 (Cont.): Characters Listed in Alphabetical Order

<i>name</i>	<i>character</i>	<i>encoding</i>			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
eta	η	—	150	—	—
eth	ð	—	—	360	—
exclam	!	041	041	041	041
exclamdown	¡	241	—	241	241
existential	∃	—	044	—	—
f	f	146	—	146	146
fi	fi	256	—	—	—
five	5	065	065	065	065
fl	fl	257	—	—	—
florin	f	246	246	—	—
four	4	064	064	064	064
fraction	/	244	244	—	—
g	g	147	—	147	147
gamma	γ	—	147	—	—
germandbls	ß	373	—	337	337
gradient	∇	—	321	—	—
grave	`	301	—	221	221
greater	>	076	076	076	076
greaterequal	≥	—	263	—	—
guillemotleft	«	253	—	253	253
guillemotright	»	273	—	273	273
guilsingleft	<	254	—	—	—
guilsingright	>	255	—	—	—
h	h	150	—	150	150
heart	♥	—	251	—	—
hungarumlaut	˘	315	—	235	235
hyphen	-	055	—	255	—

Table B-5 (Cont.): Characters Listed in Alphabetical Order

name	character	encoding			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
i	i	151	—	151	151
iacute	í	—	—	355	355
icircumflex	î	—	—	356	356
idieresis	ÿ	—	—	357	357
igrave	ì	—	—	354	354
infinity	∞	—	245	—	—
integral	∫	—	362	—	—
integralbt	∫	—	365	—	—
integralex		—	364	—	—
integraltp	∫	—	363	—	—
intersection	∩	—	307	—	—
iota	ι	—	151	—	—
j	j	152	—	152	152
k	k	153	—	153	153
kappa	κ	—	153	—	—
l	l	154	—	154	154
lambda	λ	—	154	—	—
less	<	074	074	074	074
lessequal	≤	—	243	—	—
logicaland	∧	—	331	—	—
logicalnot	¬	—	330	254	—
logicalor	∨	—	332	—	—
lozenge	◊	—	340	—	—
lslash	ł	370	—	—	—
m	m	155	—	155	155
macron	-	305	—	225	225
minus	-	—	055	055	055

Table B-5 (Cont.): Characters Listed in Alphabetical Order

<i>name</i>	<i>character</i>	<i>encoding</i>			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
minute	'	—	242	—	—
mu	μ	—	155	265	265
multiply	×	—	264	327	—
n	n	156	—	156	156
nine	9	071	071	071	071
notelement	€	—	317	—	—
notequal	≠	—	271	—	—
notsubset	⊄	—	313	—	—
ntilde	ñ	—	—	361	361
nu	ν	—	156	—	—
numbersign	#	043	043	043	043
o	o	157	—	157	157
oacute	ó	—	—	363	363
ocircumflex	ô	—	—	364	364
odieresis	ö	—	—	366	366
oe	œ	372	—	—	367
ogonek	.	316	—	236	236
ograve	ò	—	—	362	362
omega	ω	—	167	—	—
omegal	Ω	—	166	—	—
omicron	ο	—	157	—	—
one	1	061	061	061	061
onehalf	½	—	—	275	275
onequarter	¼	—	—	274	274
onesuperior	¹	—	—	271	271
ordfeminine	ª	343	—	252	252
ordmasculine	º	353	—	272	272

Table B-5 (Cont.): Characters Listed in Alphabetical Order

name	character	encoding			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
oslash	ø	371	—	370	370
otilde	õ	—	—	365	365
p	p	160	—	160	160
paragraph	¶	266	—	266	266
parenleft	(050	050	050	050
parenleftbt	(—	350	—	—
parenleftex		—	347	—	—
parenleftfp	(—	346	—	—
parenright)	051	051	051	051
parenrightbt)	—	370	—	—
parenrightex		—	367	—	—
parenrightfp)	—	366	—	—
partialdiff	∂	—	266	—	—
percent	%	045	045	045	045
period	.	056	056	056	056
periodcentered	·	264	—	267	267
perpendicular	⊥	—	136	—	—
perthousand	‰	275	—	—	—
phi	φ	—	146	—	—
phi1	φ	—	152	—	—
pi	π	—	160	—	—
plus	+	053	053	053	053
plusminus	±	—	261	261	261
product	∏	—	325	—	—
propersubset	⊂	—	314	—	—
propersuperset	⊃	—	311	—	—
proportional	∝	—	265	—	—

Table B-5 (Cont.): Characters Listed in Alphabetical Order

name	character	encoding			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
psi	Ψ	—	171	—	—
q	q	161	—	161	161
question	?	077	077	077	077
questiondown	¿	277	—	277	277
quotedbl	"	042	—	042	042
quotedblbase	„	271	—	—	—
quotedblleft	“	252	—	—	—
quotedblright	”	272	—	—	—
quoteleft	‘	140	—	140	140
quoteright	’	047	—	047	047
quotesingbase	,	270	—	—	—
quotesingle	’	251	—	—	—
r	r	162	—	162	162
radical	√	—	326	—	—
radicalex	—	—	140	—	—
reflexsubset	⊆	—	315	—	—
reflexsuperset	⊇	—	312	—	—
registered	®	—	—	256	—
registersans	®	—	342	—	—
registerserif	®	—	322	—	—
rho	ρ	—	162	—	—
ring	◊	312	—	232	232
s	s	163	—	163	163
scaron	š	—	—	—	—
second	”	—	262	—	—
section	§	247	—	247	247
semicolon	;	073	073	073	073

Table B-5 (Cont.): Characters Listed in Alphabetical Order

<i>name</i>	<i>character</i>	<i>encoding</i>			
		Adobe Standard	Symbol	ISOLatin1	DEC MCS
seven	7	067	067	067	067
sigma	σ	—	163	—	—
sigma1	ς	—	126	—	—
similar	~	—	176	—	—
six	6	066	066	066	066
slash	/	057	057	057	057
space		040	040	040	040
spade	♠	—	252	—	—
sterling	£	243	—	243	243
suchthat	∃	—	047	—	—
summation	Σ	—	345	—	—
t	t	164	—	164	164
tau	τ	—	164	—	—
therefore	∴	—	134	—	—
theta	θ	—	161	—	—
theta1	ϑ	—	112	—	—
thorn	þ	—	—	376	—
three	3	063	063	063	063
threequarters	¾	—	—	276	—
threesuperior	³	—	—	263	263
tilde	~	304	—	224	224
trademark	™	—	—	—	—
trademarksans	™	—	344	—	—
trademarkserif	™	—	324	—	—
two	2	062	062	062	062
twosuperior	²	—	—	262	262
u	u	165	—	165	165

Table B-5 (Cont.): Characters Listed in Alphabetical Order

<i>name</i>	<i>character</i>	<i>encoding</i>			
		Adobe Standard	Symbol	ISOLatin1	DECMCS
uacute	ú	—	—	372	372
ucircumflex	û	—	—	373	373
udieresis	ü	—	—	374	374
ugrave	ù	—	—	371	371
underscore	_	137	137	137	137
union	∪	—	310	—	—
universal	∀	—	042	—	—
upsilon	υ	—	165	—	—
v	v	166	—	166	166
w	w	167	—	167	167
weierstrass	∅	—	303	—	—
x	x	170	—	170	170
xi	ξ	—	170	—	—
y	y	171	—	171	171
yacute	ý	—	—	375	—
yudieresis	ÿ	—	—	377	375
yen	¥	245	—	245	245
z	z	172	—	172	172
zcaron	ž	—	—	—	—
zero	0	060	060	060	060
zeta	ζ	—	172	—	—

Compatibilities and Differences: LN03R and PRINTSERVER 40

The LN03R is compatible with the high-speed PRINTSERVER 40 Laser Print Server and supports some of the PRINTSERVER 40 features. This appendix describes the differences and compatibilities between the LN03R and the PRINTSERVER 40.

C.1 Virtual Memory

The LN03R contains 470K bytes of PostSCRIPT virtual memory. The PRINTSERVER 40 contains 1 megabyte of PostSCRIPT virtual memory. This means that a PostSCRIPT program that runs on the PRINTSERVER 40 may not run successfully on the LN03R if the program requires more virtual memory than is available on the LN03R.

C.2 Operators

Most PostSCRIPT operators are device-independent. However, to provide the user with direct access to device-dependent features of print engines, the base reference portion of the PostSCRIPT language have been extended. PostSCRIPT extensions are device-specific, but have been made as general as possible to minimize differences between the product-specific Extended PostSCRIPT language sets.

Some PRINTSERVER 40 operators have no meaning on the LN03R. The following PRINTSERVER 40 operators return "undefined" when they are issued to the LN03R implementation:

Table C-1: Undefined PRINTSERVER 40 Operators

operand	operator	result
-	A3TRAY	-
-	A4TRAY	-
-	A5TRAY	-
-	B4TRAY	-
-	B5TRAY	-
-	DEFAULTOUTPUTTRAY	integer
-	LEDGERTRAY	-
-	LEGALTRAY	-
-	LETTERTRAY	-
integer	SETDEFAULTOUTPUTTRAY	-
integer	SETDEFAULTPAPERTRAY	-
integer	SETOUTPUTTRAY	-
integer	SETPAPERTRAY	-
-	11X17TRAY	-

C.3 Communications Interface

The PRINTSERVER 40 and LN03R use different interfaces to communicate between the host and the print engine. As a result, different communications protocols are used.

The PRINTSERVER 40 is a network-oriented printer that communicates with one or more clients. The PRINTSERVER 40 uses the LAPS protocol for messages that are layered on DECNET, since the client and PRINTSERVER 40 are geographically separated.

The LN03R is connected to the host with an RS-232 interface and uses the ASCII character set for commands and data. This communications channel is more limited than the PRINTSERVER 40 and thus is not compatible with it.

MS-DOS and ULTRIX Installations

This appendix describes the procedures for installing and using the LN03R in MS-DOS and ULTRIX environments.

D.1 Using the LN03R with MS-DOS

Connect the LN03R to a serial printer port on the back of the computer. If the computer is a VAXmate, the printer should be set up for XON/XOFF protocol. If the computer is an IBM PC, the printer should be set up for polarity for DTR (Polarity Low is ready). Since the MS-DOS spooler does not recognize messages sent by the printer, error messages returned to the host are lost. Thus, if your PostSCRIPT output has an error, you will not receive the error message. If you select a font that is not available on the LN03R, an *invalidfont* error results.

D.1.1 Using Applications with MS-DOS

Some MS-DOS applications do not support the LN03R. However, the LN03R is similar to the LaserWriter and LaserWriter Plus,[®] allowing you to use applications that support those products. When loading your software application and configuring the LN03R, select the Laserwriter or LaserWriter Plus as the printer connected to your VAXmate or PC/AT clone. Make the same selection for MS-Windows.

If you are using MS-Windows, display the control window on the screen and add the PSCRPT.DRV driver to the system. Then return device control to the PostSCRIPT driver. For applications that do not run under MS-Windows, use the set-up feature and assign control to a LaserWriter Plus.

[®] LaserWriter and LaserWriter Plus are trademarks of the Apple Computer Corporation.

The difference between the LN03R and non-DIGITAL POSTSCRIPT printers is the range of available typefaces. The typefaces normally available with all POSTSCRIPT printers are the following:

- TIMES with Roman, Italic, Bold, and Bold Italic
- COURIER: with Normal, Oblique, Bold, and Bold Oblique
- HELVETICA: with Normal, Oblique, Bold, and Bold Oblique
- SYMBOL

In addition to the above, the LN03R also supplies the following faces:

- NEW CENTURY SCHOOLBOOK: with Roman, Italic, Bold, and Bold Italic
- SOUVENIR LIGHT: with Light Italic, Demi, and Demi Italic
- AVANT GARDE BOOK: with Book Oblique, Demi, and Demi Oblique
- LUBALIN GRAPH BOOK: with Book Oblique, Demi, and Demi Oblique

The Laserwriter applications do not support any of the additional faces. The LaserWriter Plus applications support New Century Schoolbook Roman and Avant Garde Book (plus some other typefaces not supported by the LN03R).

D.1.2 Transmitting Print Requests

Since transmitting a print request can be slow in the MS-DOS environment, the LN03R displays the transmission status on the front panel. A flashing controller error with a "6" displayed indicates that the computer and printer controller are communicating and generating an image. The printer is active as long as these symbols are displayed. Another indication that the printer is active is that the print spooler icon appears on your windows screen.

Transmitting print requests that contain graphics produced by a "paint" program take more time, since a bitmap (rather than image-drawing directions) is sent to the printer. Bitmaps are up to 900K bytes in size, and the communications link is 9600-19.2K baud, which accounts for the delay.

D.2 Using the LN03R with VMS VAXmate Server

The LN03R can be used on a VAX or MicroVax II system running VMS and acting as a server for a network of VAXmates or PC clones. To set up a VMS server for VAXmates, follow these steps:

1. Install the LN03R software by the standard procedure (see the *VAX/VMS Software Installation Guide: LN03R SCRIPTPRINTER*). Create generic print queues for POSTSCRIPT and ANSI output with recognizable names, such as LN03R\$POST and LN03R\$ANSI.
2. Install the VMS server software by standard procedure. Using PC ADMIN, add the generic print queues, using the ADD PRINTER QUEUE menu item.
3. Create a key diskette for each VAXmate or PC Clone and direct the printer ports to REMOTE printers on the host system. For instance, direct LPT1 to LN03R\$POST and LPT2 to LN03R\$ANSI.

When you start up the VAXmate, the initialization process shows each printer port and the queue on the host system that each is directed to. Using MS-Windows, bring up the CONTROL panel and, with the INSTALLATION menu, select ADD NEW PRINTER. Add the POSTSCRIPT driver and LA50 driver to the known drivers in the system. Using the SETUP menu, select the printer ports. This associates the drivers with the correct printer port. For instance, associate LPT1 (LN03R\$POST) with the POSTSCRIPT driver and LPT2 (LN03R\$ANSI) with the LA50 driver. The LA50 driver outputs ASCII text, which can be printed by the LN03R's ANSI translator.

To print using MS-Windows, use the control panel to select the default printer port. Then when you print files, they are directed to the default printer port and thus to the correct queue. Many applications that run under MS-Windows allow you to select the printer, so the default may be overridden.

For applications not running under MS-Windows, you must select the printer port using the application's setup routine.

To print directly from MS-DOS, use the NET PRINT command and direct the output to the appropriate printer port.

D.3 Using the LN03R with ULTRIX

The LN03R is connected to your system via a serial communication line. Follow the directions in your ULTRIX installation documentation for setting up the communications line for a printer. Configure the LN03R for XON/XOFF protocol.

D.3.1 The TRANSCRIPT Application

To use the LN03R in the ULTRIX environment, you need to use the TRANSCRIPT application package from Adobe Systems. This package includes instructions for installing the software, which performs the following functions:

- Provides the communications path between the printer daemon (LPD) and the LN03R.
- Creates banner pages for files sent to the printer and provides a front end (enscript) to the LPR command.
- Provides filters for converting various formatted output into POSTSCRIPT, such as DITROFF and ASCII.
- Returns error messages from the printer and puts them in a log file.

Character Bitmap Distribution Format

This document* describes Adobe System's character bitmap distribution format, version 2.1. The format is intended to be easily understood by both humans and computers. The format described is subject to change without prior notification.

NOTE

This appendix describes the format of tapes obtained from Adobe Systems, Inc. Tapes obtained from other sources may have different formats.

E.1 Tape Format

Each tape is 1600 bpi, nine track, unlabeled, and contains two or more files. Each file is followed by an EOF mark. The last file on the tape is followed by two EOF marks. Physical records contain 512 bytes. The last physical record in a file (preceding an EOF mark) may contain fewer than 512 bytes.

Each file is encoded in the printable characters (octal 40 through 176) of USASCII plus carriage return and linefeed. Each file consists of a sequence of variable-length lines. Each line is terminated by a carriage-return (octal 015) and line-feed (octal 012). The first file on the tape is the Adobe Systems Copyright notice, followed by font files. The format of font files is described in the following sections.

NOTE

Font tapes may also be obtained in UNIX* tar format. Be sure to specify tar format if desired. No other tape formats are currently supported by Adobe Systems.

* Copyright Adobe Systems, Inc. Reprinted by permission.

* UNIX is a registered trademark of American Telephone & Telegraph Company.

E.2 File Format

Character bitmap information will be distributed in an USASCII encoded, human readable form. The information about a particular family and face at one size and orientation will be contained in one file. The file begins with information pertaining to the face as a whole, followed by the information and bitmaps for the individual characters.

A font bitmap description file has the following general form, where each item is contained on a separate line of text in the file. Items on a line are separated by spaces.

1. The word `STARTFONT` followed by a version number, indicating the exact file format used. The version described here is number 2.1.
2. One or more lines beginning with the word `COMMENT`. These lines may be ignored by any program reading the file.
3. The word `FONT` followed by the family name and the face name, separated by a hyphen.
4. The word `SIZE` followed by the point size of the characters, the x resolution and y resolution of the device for which these characters were intended.
5. The word `FONTBOUNDINGBOX` followed by the width in x , height in y , and the x and y displacement of the lower left corner from the origin. (See the examples in Section E.4.)
6. Optionally the word `STARTPROPERTIES` followed by the number of properties (p) that follow.
7. Then come p lines consisting of a word for the property name followed by either an integer or string surrounded by `quotedbl` (042). Internal `quotedbl` characters are indicated by using two in a row.
8. The property section, if it exists, is terminated by `ENDPROPERTIES`.
9. The word `CHARS` followed by the number of character segments c that follow.
10. Then come c character segments in the following format:
 - a. The word `STARTCHAR` followed by up to 14 characters (no blanks) of descriptive name of the glyph.
 - b. The word `ENCODING` followed by a positive integer representing the Adobe Standard Encoding value. If the character is not a member of the Adobe Standard Encoding, `ENCODING` is followed by -1 and an optional integer specifying the glyph index.

- c. The word SWIDTH followed by the scalable width in x and y of character. Scalable widths are in units of 1/1000th the size of the character. If the size of the character is p points, the width information must be scaled by $p/1000$ to get the width of the character in printer's points. This width information should be considered as a vector indicating the position of the next character's origin relative to the origin of this character. To convert the scalable width to the width in device pixels, multiply SWIDTH times $p/1000$ times $r/72$ where r is the device resolution in pixels per inch. The result is a real number giving the ideal print width in device pixels. The actual device width must of course be an integral number of device pixels and is given in the next entry.
 - d. The word DWIDTH followed by the width in x and y of the character in device units. Like the SWIDTH, this width information is a vector indicating the position of the next character's origin relative to the origin of this character.
 - e. The word BBX followed by the width in x (BBw), height in y (BBh) and x and y displacement ($BBox$, $BBoy$) of the lower left corner from the origin of the character.
 - f. The optional word ATTRIBUTES followed by the attributes as 4 hex-encoded characters. The interpretation of these attributes is undefined in this document.
 - g. The word BITMAP.
 - h. h lines of hex-encoded bitmap, padded on the right with zeroes to the nearest byte (i.e., multiple of 8).
 - i. The word ENDCHAR.
11. The file is terminated with the word ENDFONT.

E.3 Metric Information

Figures E-1 and E-2 best illustrate the bitmap format and character metric information.

Figure E-1: Example of a Descender

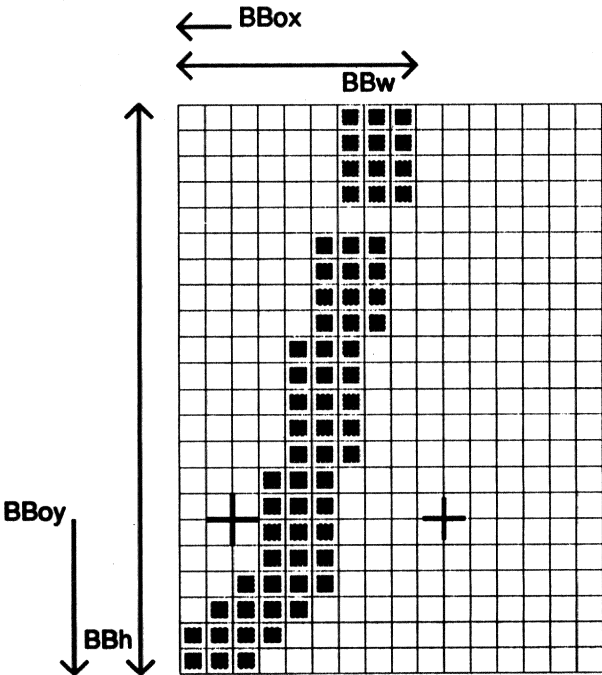
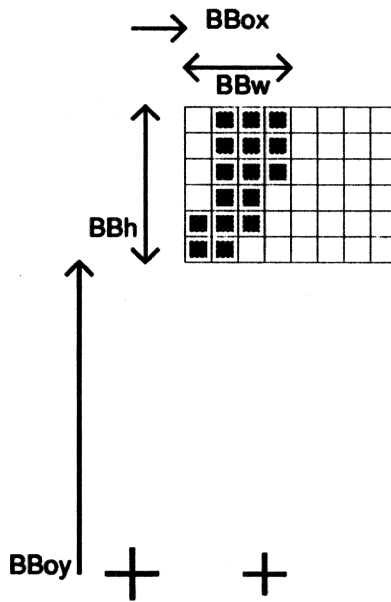


Figure E-2: Example with the Origin Outside the Bounding Box



E.4 An Example File

The following is an example of a bitmap file containing the specification of two characters (j and quoteright).*

```
STARTFONT 2.1
COMMENT This is a sample font in 2.1 format.
FONT Helvetica-Bold
SIZE 8 200 200
FONTBOUNDINGBOX 9 24 -2 -6
STARTPROPERTIES 2
MinSpace 4
Copyright "Copyright (c) 1987 Adobe Systems, Inc."
ENDPROPERTIES
CHARS 2
STARTCHAR j
ENCODING 106
SWIDTH 355 0
DWIDTH 8 0
BBX 9 22 -2 -6
BITMAP
0380
0380
0380
0380
0000

0700
0700
0700
0700
0E00
0E00
0E00
0E00
0E00

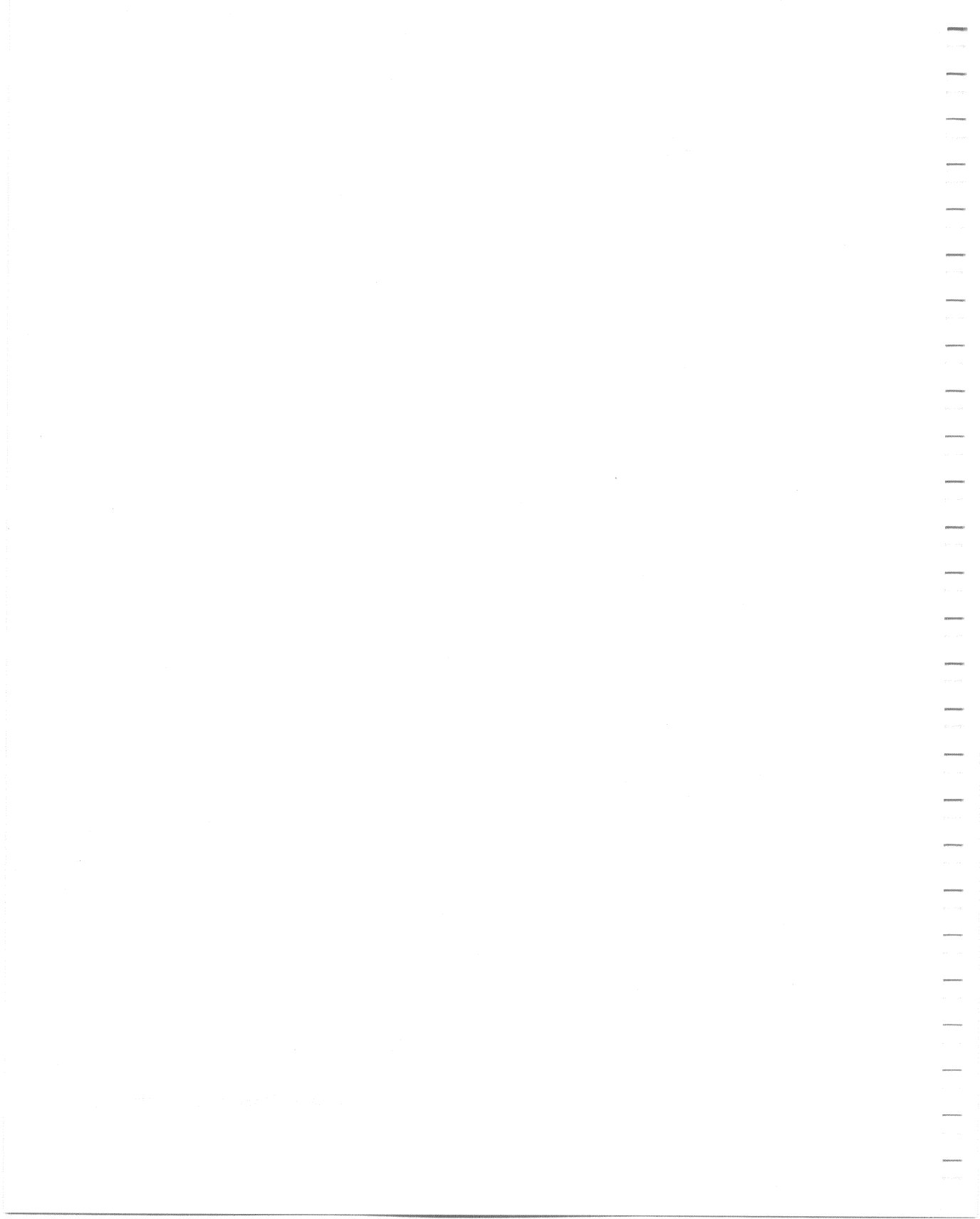
1C00
1C00
1C00
1C00

2C00
7800
F000
E000
```

* Helvetica is a registered trademark of Allied Corporation.

ENDCHAR
STARTCHAR quoteright
ENCODING 39
SWIDTH 223 0
DWIDTH 5 0

BBX 4 5 2 12
ATTRIBUTES 01C0
BITMAP
70
70
60
EO
CO
ENDCHAR
ENDFONT



Glossary

bitmap A digitized image that can be stored, transmitted, and reproduced.

composite character A character that is made up of two or more other characters. For example, an accented letter is a composite character.

Ethernet A local area network that uses coaxial cable as a passive communication medium to connect different types of computers, printer products, and office equipment at a site.

encoding vector The association between character codes and character descriptions. An encoding vector on the LN03R is a 256-element array, indexed by character code. The elements of the array are character names.

font The artistic representation of a typeface that describes a set of characters by point size, weight, and style.

font file A data file that contains information used to reproduce a font.

host The computer that provides services and enables start up and management of the peripheral devices, such as printers.

initialize To set counters, switches, addresses, or contents of memory to 0 or other starting values at the beginning of, or at prescribed points in, a computer routine.

interpreter A program in the printer that converts data for imaging data syntax, such as text or graphics, into a bitmap.

kerning Subtracting the space between characters.

language extensions POSTSCRIPT operators and objects added to the standard POSTSCRIPT operators and objects in a specific implementation of the POSTSCRIPT interpreter. These extensions control the system parameters for that type of POSTSCRIPT printer.

line A physical communications path.

name A descriptive identifier (ASCII string) that is associated with a subject or object in the system.

network A group of computers that are connected by communications lines to share information and resources.

object type of extensions POSTSCRIPT language extensions can be of type **operator**, **integer** or **string**. See the *POSTSCRIPT Language Reference Manual* for more information.

operator A built-in POSTSCRIPT language object that performs operation. An operator, when invoked, can result in data being manipulated, system parameters being modified, and/or sheets being printed.

persistent parameters System parameters that remain unchanged across POSTSCRIPT print jobs. These parameters are stored in nonvolatile memory and therefore are maintained even when power is removed from the printer.

printer controller firmware The software that interprets the data in a print request according to a specified data syntax, builds bitmaps of each page to be printed, and forwards the bitmaps to the print engine that produces the hard-copy output.

printer software The software that handles the communications among the process that makes a print request (terminal), the process that provides resources (a host), and the process that performs the printing service (a print queue).

print spooler A feature that enables a computer to maintain a print queue and print documents while performing other computer tasks.

privilege The level of system access allowed to a user.

system parameter Values maintained within the POSTSCRIPT interpreter and used by the POSTSCRIPT interpreter to control various system-specific features. These features are related to the print engine model and the type of communication interface between the printer and the host. Also see **persistent parameters** and **volatile parameters**.

translator A program that changes data into a form that can be used by the printer.

user The person who initiates requests for services. These requests are handled by the host, which forwards them to the appropriate queue.

volatile parameters System parameters that stay in effect for one POSTSCRIPT print job. At the end of the print job, volatile parameters revert to their default values.



INDEX

A

Adobe bitmap
 example file • E-6
 file format • E-2
 metric information • E-3
 tape format • E-1
Ancillary interfaces • 1-5
Application programs • 1-5
 TRANSCRIPT • D-4
 with MS-DOS • D-1
 with ULTRIX • D-4
Automatic Spacing • 5-2

B

banddevice operator
 restriction • 6-4
bind operator • 5-2

C

Character bitmap distribution format • E-1
Character encoding
 encoding vectors • B-1
Character metrics section • 4-7
Character set
 control characters • 6-12
 printable • 6-13
checkpassword operator • 3-3
clear operator
 restriction • 6-4
Clipping values • 2-9

closefile operator
 restriction • 6-4
Communications Interface • C-2
Composite characters • 4-10
Context layer • 6-2
Context state
 current • 6-2
 preserving • 6-3
Coordinate system positioning • 2-9
Coordinate system unit size • 2-9
coppage operator • 5-3
Current context • 6-2

D

defaultjobtimeout operator • 3-4
Default timeouts • 2-11
defaulttimeouts operator • 3-5
Device controller • 1-4
Device control modules • 6-1
 POSTSCRIPT operators • 6-2
 PRINT command parameters • 6-1
 redefining POSTSCRIPT operators • 6-2
Device-dependent programs • 6-3
Dictionary stack
 limit to entries • 6-9
dostartpage operator • 3-6

E

eescratch operator • 3-7
eexec operator • 3-8
Encoding vectors • 2-12, B-1
 Adobe Standard Encoding • B-1

Encoding vectors (cont'd.)

- DEC MCS • B-6
- examples • 2-12
- ISO Latin1 • B-4
- Symbol • B-9

- erasepage operator
 - restriction • 6-4
 - executive operator • 3-9
 - exitserver operator • 3-10
-

F

- findfont operator • 6-9
 - Font metrics • 4-1
 - character metrics • 4-7
 - composite character section • 4-10
 - file format • 4-3
 - sample • 4-3
 - file names • 4-1
 - global font keys • 4-5
 - kerning data section • 4-8
 - Fonts
 - global font keys • 4-5
 - numbers for idlescan conversion • 2-7
 - framedevice operator
 - restriction • 6-4
-

G

- Global section • 4-5
 - gsave
 - limit to use • 6-9
-

H

- Host system hardware requirements • 1-4
-

I

- idlefonts operator • 3-11
- initclip operator
 - restriction • 6-4
- initgraphics operator
 - restriction • 6-5
- initmatrix operator
 - restriction • 6-5

- Interactive mode • 2-5
-

J

- jobname operator • 3-13
 - jobtimeout operator • 3-14
-

K

- Kerning data section • 4-8
-

L

- Layering • 6-2
 - Limit to entries
 - dictionary stack • 6-9
 - Limit to use
 - gsave • 6-9
 - Polygon fill • 6-9
 - quit operator • 6-10
 - start • 6-10
 - vmstatus • 6-10
 - LN03R
 - application programs • 1-5
 - compatibility with LPS40 • C-1
 - hardware • 1-2
 - LN03R software • 1-5
 - LPS40
 - compatibilities with LN03R • C-1
 - differences with LN03R • C-1
-

M

- Margins
 - paper • 2-9
- margins operator • 3-15
- MS-DOS
 - connecting the LN03R • D-1
 - installations • D-1
 - LN03R applications • D-1
 - Transmitting Print Requests • D-2
- MS-Windows • D-1
- multiple copies • 5-4

N

Name of job parameter • 2-12

O

Operators

compatibility with LPS40 • C-1

P

POSTSCRIPT

clipping values • 2-9
coordinate system positioning • 2-9
coordinate system unit size • 2-9
extensions
 checkpassword • 3-3
 defaultjobtimeout • 3-4
 defaulttimeouts • 3-5
 dostartpage • 3-6
 eescratch • 3-7
 eexec • 3-8
 executive • 3-9
 exitserver • 3-10
 format • 3-1
 idlefonts • 3-11
 jobname • 3-13
 jobtimeout • 3-14
 margins • 3-15
 object types • 3-1
 pagecount • 3-16
 pagestackorder • 3-17
 papersize • 3-18
 printername • 3-19
 product • 3-20
 restrictions • 3-2
 revision • 3-21
 setdefaultjobtimeout • 3-22
 setdefaulttimeouts • 3-23
 setdostartpage • 3-24
 seteescratch • 3-25
 setidlefonts • 3-26
 optimizing performance with • 5-4
 setjobtimeout • 3-28
 setmargins • 3-29
 setpassword • 3-31

POSTSCRIPT

extensions (cont'd.)

 setprintername • 3-32
 setvmlimit • 3-33
 using • 3-2
 waittimeout • 3-34

operators

 findfont • 6-9
 guidelines • 6-3
 quit • 6-10
 restrictions • 6-9

POSTSCRIPT software driver

 techniques to avoid • 6-12
 useful techniques • 6-11
 writing • 6-11

Page counter

 parameter • 2-10

pagecount operator • 3-16

pagestackorder operator • 3-17

Paper feed • 2-9

Paper handling

 margins • 2-9
 parameters • 2-8

Paper orientation • 2-9

papersize operator • 3-18

Password

 parameter • 2-11

Performance

 optimizing • 5-1

Polygon fill • 6-9

 limit to use • 6-9

Printer • 1-2

 device controller • 1-4
 paper trays • 1-4
 print engine • 1-3
 printer name parameter • 2-10

Printer name

 parameter • 2-10

printername operator • 3-19

Procedure calls

 eliminating • 5-2
 product operator • 3-20

Q

quit operator • 6-10

 limit to use • 6-10

quit operator (cont'd.)
restriction • 6-6

R

renderbands operator
restriction • 6-5

Restrictions

banddevice operator • 6-4
clear operator • 6-4
closefile operator • 6-4
control characters • 6-12
dictionary stack entries • 6-9
erasepage operator • 6-4
findfont • 6-9
framedevice operator • 6-4
gsave • 6-9
initclip operator • 6-4
initgraphics operator • 6-5
initmatrix operator • 6-5
POSTSCRIPT operators • 6-4
polygon fill • 6-9
quit • 6-10
quit operator • 6-6
renderbands operator • 6-5
setmatrix operator • 6-5
start • 6-10
systemdict operator • 6-5
transform operator • 6-5
userdict operator • 6-5
vmstatus • 6-10

revision operator • 3-21

S

Scaling the Coordinate System • 5-3

Scan conversion
fonts • 2-7

Scan conversion parameter • 2-6

Scratch memory
parameter • 2-11

setdefaultjobtimeout operator • 3-22

setdefaulttimeouts operator • 3-23

setdostartpage operator • 3-24

seteescratch operator • 3-25

setidlefonts extension
optimizing performance with • 5-4

setidlefonts operator • 3-26
setjobtimeout operator • 3-28

setmargins operator • 3-29
adjustment ranges • 3-29

setmatrix operator
restriction • 6-5

setpassword operator • 3-31

setprintername operator • 3-32

setvmlimit operator • 3-33

show operator • 5-2

showpage operator • 5-3

Smaller operator names • 5-2

Software revision parameter • 2-12

start

limit to use • 6-10

systemdict operator
restriction • 6-5

System parameters • 2-1

default timeout • 2-11

location • 2-2

name of current job • 2-12

page counter • 2-10

paper handling • 2-8

password • 2-11

persistent • 2-2

printer name • 2-10

scan conversion • 2-6

scratch memory • 2-11

software revision • 2-12

test page status • 2-10

volatile • 2-5

T

Test page

parameter • 2-10

transform operator

restriction • 6-5

Translators • 1-5

U

ULTRIX • D-4

Installing the LNO3R • D-4

userdict operator

restriction • 6-5

V

Virtual memory

- LPS40 and LN03R • C-1
- operators that consume • 6-7
- persistent parameters • 2-2
- preventing limitchecks • 6-6
- setting limit • 3-33
- status • 6-10
- volatile parameters • 2-5

VM

See Virtual memory

vmstatus

- limit to use • 6-10

VMS VAXmate Server

- using the LN03R with • D-3
-

W

waittimeout operator • 3-34

widthshow operator • 5-2



HOW TO ORDER
ADDITIONAL DOCUMENTATION

From	Call	Write
Alaska, Hawaii, or New Hampshire	603-884-6660	Digital Equipment Corporation P.O. Box CS2008 Nashua, NH 03061
Rest of U.S.A. and Puerto Rico*	800-258-1710	
* Prepaid orders from Puerto Rico must be placed with DIGITAL's local subsidiary (809-754-7575)		
Canada	800-267-6219 (for software documentation)	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6 Attn: Direct Order desk
	613-592-5111 (for hardware documentation)	
Internal orders (for software documentation)	—	Software Distribution Center (SDC) Digital Equipment Corporation Westminster, MA 01473
Internal orders (for hardware documentation)	617-234-4323	Publishing & Circulation Serv. (P&CS) NR03-1/W3 Digital Equipment Corporation Northboro, MA 01532

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent:

- Assembly language programmer
- Higher-level language programmer
- Occasional programmer (experienced)
- User with little programming experience
- Student programmer
- Other (please specify) _____

Name _____ Date _____

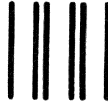
Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear — Fold Here and Tape

digital

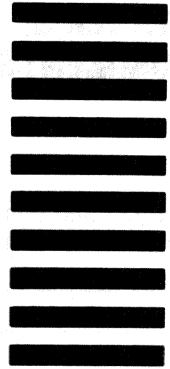


NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

**DIGITAL EQUIPMENT CORPORATION
CORPORATE USER PUBLICATIONS
MLO5-5/E45
146 MAIN STREET
MAYNARD, MA 01754-2571**



Do Not Tear — Fold Here

Cut Along Dotted Line