



HPL Operating Manual

*for the HP 9000 Series 200
Model 216/226/236 Computers*

Manual Part No. 98614-90010

© Copyright 1983, Hewlett-Packard Company.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this document is subject to change without notice.

Use of this manual and flexible disc(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs can be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(B) of the Rights in Technical Data and Software clause in DAR 7-104.9(a).



Hewlett-Packard Company
3404 East Harmony Road, Fort Collins, Colorado 80525

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

January 1984...First Edition. This manual replaces the HPL Operating Manual & Programming Update, 09826-90040. All versions of Series 200 HPL are covered in this manual.

Warranty Statement

Hewlett-Packard products are warranted against defects in materials and workmanship. For Hewlett-Packard Fort Collins Systems Division products sold in the U.S.A. and Canada, this warranty applies for ninety (90) days from the date of delivery. * Hewlett-Packard will, at its option, repair or replace equipment which proves to be defective during the warranty period. This warranty includes labor, parts, and surface travel costs, if any. Equipment returned to Hewlett-Packard for repair must be shipped freight prepaid. Repairs necessitated by misuse of the equipment, or by hardware, software, or interfacing not provided by Hewlett-Packard are not covered by this warranty.

HP warrants that its software and firmware designated by HP for use with a CPU will execute its programming instructions when properly installed on that CPU. HP does not warrant that the operation of the CPU, software, or firmware will be uninterrupted or error free.

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

* For other countries, contact your local Sales and Support Office to determine warranty terms.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Table of Contents

Chapter 1: Getting Started

Introduction	1
System Loading	2
Reserved HPL Select Codes	2
Flexible Discs	3
Disc Handling Precautions	3
Inserting Discs	4
Write Protection	4
Data File Compatability	5
Using Discs	5
Copying Discs	6
Operating Features	8
Which Computer	8
Program Autostart	8
Computer Operating Features	9
Display Organization	10
Graphics Mode	11
Keyboard Operations	12
Character Entry Keys	12
Numeric Pad	12
Cursor Controls	13
Editing Lines	13
System Control Keys	16
Special Function Keys	18
Labelled Special Function Keys	18
Immediate Execute Special Function Keys	20
Immediate Continue Special Function Keys	20
Keys with Multiple Statements	21
Exended Control Special Function Keys	21
Default Special Function Keys	22
Program Control Keys	23
Arithmetic Operations	24

Chapter 2: Program Transfer

Introduction	27
Disc Transfer	27
Interface Transfer	28
Programs	28
Data	28

Chapter 3: HPL Programming

Introduction	31
Mainframe Programming	32
The Read, Data and Restore Statements	32
The Programmable Beep Statement	33
The Machine Function	33
CRT Display Control	34
The Clear Alpha Statement	34
Highlighting on Model 216 and 236	34
The Alpha On, Alpha Off Statements	35
The Key Labels On/Off Statements	35
The Dump Alpha Statements	35
The Tab X-Y Statements	35
The Read CRT Statements	36
The System Printer Select Code Statement	36
Math Functions	37
Flags	37
The Cross Reference Statement	37
The Find Statement	37
Tape Cartridge Operations	38
Record Binary Statement	40
String Variables	40
Value Function	40
String Function	40
Read Statement	40
Systems Programming	41
Intelligent Terminal Instructions	41
The On Knob Statement	41
The Knob Function	41
The Knob Status Function	42
The Key Buffer Empty Function	42
Serial Interface Control	42
The Remote Keyboard Statement	43
The Press Keyboard Statement	43
The Define SFK Statement	44
Systems Programming Instructions	45
The System Boot Statement	45
Matrix Programming	45
Array Output	45
Disc Programming	46
The Mass Storage Unit Specifier	46
Device Format Table	46
The Mass Storage Is Statement (msi)	48
The Assign Statement	48
The Drive Statement	49
The Initialize Statement	49
The Disc Type Function	51
The Catalog Statement	52

The Killall Statement	53
The Open Statement	53
The Disc Copy Statement	54
New Alternate Syntax for Disc Copy	55
The File Copy Statement	55
The Save Binary Statement	56
The Disc Read and Disc Write Statements	56
Unimplemented Disc Programming Statements	57
Unimplemented Binary Statements	57
More I/O	58
Interface Control Operations	58
HP-IB Operations	60
98626A Serial I/O Operations	62
Internal Clock Access	65
The Set Clock-Time Statement	65
The Read Clock-Time Function	65
Interrupt Control	66
The On Interrupt Statement	66
Interrupt Lockouts	66
The On Clock-Cycle Statement	66
The Cycle Function	66
The On Clock-Delay Statement	67
The On Clock-Match Statement	67
Buffered I/O	68
Terminating I/O Transfers	68
I/O Buffer Extended Status	68
I/O Buffer Pointer Control	68
Saving String Buffers to Disc	69
Plotter Control	70
Internal Graphics Control	70
The Plotter Select Code Statement	70
The Pen Select Statement	71
Graphics Screen Control Statements	71
The Dump Graphics Statement	72
The Graphics Pointer (Cursor) Statement	72
Binary Graphics	72
The Binary Plot Statement	73
The Graphics Load and Graphics Store Statements	74

Chapter 4: Powerfail Programming

Introduction	77
What Is It?	77
Detecting a Power Fail	78
The Power Function	78
The Power Shutdown Statement	79
The Power Time Statement	79

Chapter 5: Color Interface Programming

The Graphics Instructions	81
Using the Statements in Programs	82
Character Animation	83
Alpha Text Programming	84
Setting the Alpha Print Position	84
Writing to the Holding Register and CRT	86
The Interface Set-up Instruction	86
Select Text Color and Inverse Video	87
Summary of Alpha Instructions	88

Appendix A: Disc Programming Technical Appendix

Supported Device/Format Combinations	A-1
HPL Implementation of LIF	A-2
LIF Disc Structure	A-2
Rules for Legal File Names	A-2
LIF Directory Entries Support Larger File Sizes	A-3
LIF Discs May Require More Frequent Repacking	A-3
LIF ASCII Files	A-3
Creating ASCII Files	A-3
Assigning ASCII Files	A-4
Accessing ASCII Files	A-4
Using the Type Function with ASCII Files	A-5
Using On End with ASCII Files	A-5
Binary Data File Support	A-5
Creating Binary Data Files	A-5
Assigning Binary Data Files	A-6
Accessing Binary Data Files	A-6
Assigned File Pointers on 8290x-Series Discs	A-7
9885 Disc Access Requires a DMA Card	A-7
HP-IB Programming Consideration	A-7
HPL Disc Programming Error Messages	A-7

Appendix B: Code Charts

ASCII Control Codes	B-2
Keycode Conversion Tables	B-3

Appendix C: Series 200 HPL Differences

Mass Storage Operations	C-1
Graphics Changes	C-1
Color Interface Support	C-2
Color Alpha	C-3
Powerfail Support	C-4
Display Control	C-4
Data Transfers	C-5
Error Trapping	C-5
String Search	C-6
On-event Changes	C-6
HPL 1.0 Bugs Fixed in HPL 2.0	C-6
Human Interface Changes	C-7
HPL 2.0 Bugs Fixed in HPL 2.1	C-8
HPL 2.0 Bugs Not Fixed	C-8

Index



Chapter 1

Getting Started

Introduction

This manual helps an experienced HPL programmer get started using HPL on a Series 200 computer. Series 200 HPL is supported on these computers:

- Model 216 (HP 9816)
- Model 226 (HP 9826)
- Model 236 (HP 9836 with monochrome display)

HPL is not supported on the Model 220 (HP 9920), Model 236 with color display (9836C), or computers with the HP 98143U Memory Management upgrade.

This manual assumes you are familiar with 9825 HPL. A set of current 9825 manuals is supplied with Series 200 HPL for your reference. The differences between 9825 HPL and Series 200 HPL are mentioned throughout this manual. The differences between revisions of Series 200 HPL are listed at the back of this manual.

System Loading

This section shows you how to load Series 200 HPL on your computer. You should already have unpacked, installed and powered-up your computer, as explained in its installation or operating manual.

If you have additional memory or interface cards, be sure to install them as explained in the computer's manual before loading HPL. When installing interface cards, remember that these select codes are reserved by the HPL system:

Reserved HPL Select Codes

0	Keyboard and CRT display line
7	HP-IB interface (built-in)
16	CRT print area

That leaves select codes 1 thru 6 and 8 thru 15 for plug-in interface cards. The manual supplied with each card shows how to set its select code switches.

Series 200 HPL 2.1 is supplied on one system disc. Earlier versions of HPL, HPL 1.0 and HPL 2.0, were available on both disc and ROM cards. The following pages show you how to load HPL 2.1 from disc. The same procedure can be used to load earlier disc versions.

To Load HPL 2.1 from Disc:

1. Insert the HPL System Disc in the default disc drive. This is the Model 226 internal drive, the Model 236 right-hand internal drive, or the unit 0 drive connected to a Model 216.
2. Switch the computer on. With a Model 216 computer, switch the disc drive on first.

The computer first runs its self-tests and then looks for an operating system to load. If the HPL disc is in a drive, the computer will find it and load HPL. The message:

```
(RAM) HPL 2.1 Ready
```

indicates you are ready to use the system.

If the computer doesn't display the ready message after about 15 seconds or if the message `UNABLE TO FIND SYSTEM` is displayed, first try to load the system again. Switch the computer off, check to be sure the system disc is inserted correctly in a disc drive, close the drive door and switch the computer on again. With the Model 216 computer, verify that the HP-IB cable is connected between the drive and the computer and the drive is switched on. The drive should be set to bus address 00 and the disc should be inserted in the drive indicated as unit 0.

If the system does not load, either the disc is defective or the computer requires service. Call HP for service. A list of office locations was supplied with the system.

The next section explains how to handle, use and back up your discs. Be sure to make at least one back-up copy of your HPL system disc after loading the system. See the following pages for details.

Flexible Discs

This section introduces you to the flexible disc media and explains how to copy (back up) the contents of one disc to another. HPL operating commands are available for initializing discs, cataloging disc files and purging disc files. These and other commands are explained in the HPL Programming chapter.

The built-in disc drive handles standard 5¼ inch flexible discs. The flexible disc, also called a mini-disc and a diskette, is a thin piece of plastic enclosed in a special plastic jacket. The disc is covered with a thin oxide coating on which your program and data information are stored.

When you insert the disc in the drive and close the door, the drive is ready to read information from or write information onto the disc. When the computer requests a read or write, the disc spins at a constant rate, (like a phonograph record). The yellow light on the disc drive indicates that reading or writing is taking place. Do not attempt to remove the disc when the yellow light is on.

The built-in disc drive reads and writes on both sides of the disc and requires discs labeled for “double-sided” and “double density” use. Be sure to use only media supplied or approved by HP. Boxes of ten discs are available by ordering HP part number 92190A. Other discs may not be of adequate quality or may damage the drive.

Disc Handling Precautions

Be sure to follow these guidelines to ensure trouble-free operation:

- Handle discs only by the labeled area. Never touch the disc surface which shows through the protective jacket.
- Always return the disc to its storage envelope after each use. The envelope not only protects the disc from physical damage, it's made of an anti-static material to prevent dust from accumulating.
- Write only on the disc label using only a felt-tip pen. Don't write on the disc jacket. Don't use a lead pencil or a ball-point pen.
- Although the disc is flexible, don't bend or fold it.
- Avoid using or storing discs in temperature extremes, or in areas with excessive smoke or dust. Even cigarette ash can damage the disc surface. Close the disc drive door when it's not in use.
- Do not place discs near sources of strong magnetism, such as an electric motor or toy magnet. This will destroy data on the disc and may prevent further use of the disc.
- Do not attempt to clean the disc or remove it from its protective jacket.
- Use only discs approved by HP. Others may impair data integrity or damage the disc drive.

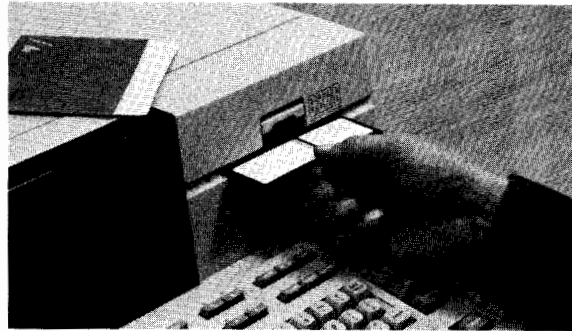
Note

Do not use more than two layers of adhesive labels on a disc. Additional labels could cause the disc to jam in the drive or prevent reliable disc operation.

Inserting and Removing Discs

Open the drive door by lifting the door handle up. Check to make sure there is not another disc in the drive already. Insert the disc as shown on the right. Close the door.

Be sure to return the disc to its storage envelope when not in use. This keeps dust from getting on the oxide surface. Also close the drive door when not in use.



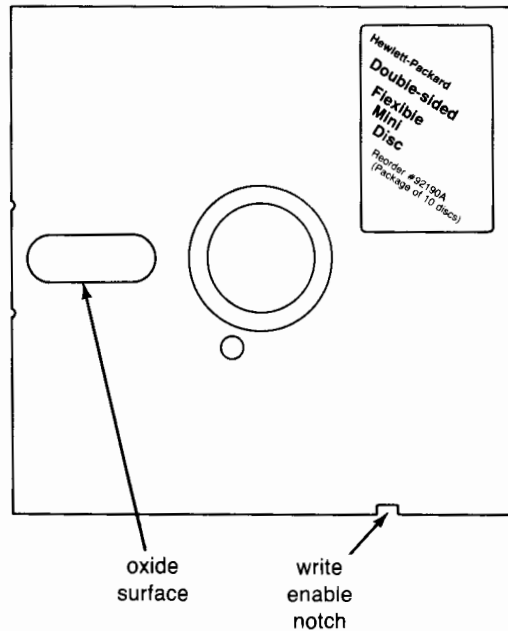
CAUTION

IF YOU ACCIDENTALLY INSERT ANOTHER DISC WHEN ONE IS ALREADY IN THE DRIVE, REMOVE THE BOTTOM DISC FIRST. OTHERWISE, THE READ/WRITE HEADS COULD BE DAMAGED.

Write Protection

Covering or uncovering a notch in the disc jacket determines whether the disc drive can write information on the disc. When the notch is covered, it's impossible for the drive to write on the disc; thus information already on the disc is protected from being written over or erased. This is useful when a disc contains source information which should only be read.

Labels are supplied with discs to allow you to cover the write-enable notch.



The disc catalog (or directory) is listed on the display.

If you wish to use 9825-type tape commands, you must still “mark” files just as if there were a tape cartridge inserted in the computer. Then commands such as rcf and ldf can be used to store and load programs and data. Refer to the Tape Cartridge Operations section of the HPL Programming chapter in this manual for additional topics of concern when using tape commands.

Copying Discs

Although flexible discs are an extremely reliable storage media, like phonograph records, they do wear out. Since discs can also be damaged due to accidents or careless handling, you should keep a duplicate or back-up copy of each important disc. HPL programs are provided on a utilities disc to copy the files from one disc to another on the Model 226. On the Model 236, use the “copy” statement described below. The “cbackup” (complete backup) program automatically copies all files from one disc to another. The “ibackup” (individual backup) program allows copying selected files to the same disc or a second disc. The following instructions show you how to run the cbackup program.

The “cbackup” program can be used to copy all files from a disc originated by a Model 226 Computer. Although the program runs on the HPL language system, it copies disc files containing BASIC. HPL and other programs originated on a Model 226.. “cbackup” will also copy files from mini discs recorded on other HP equipment which conforms to LIF (Logical Interchange Format) standards. Your HP sales office can furnish a list of LIF-compatible equipment.

The “cbackup” program copies files from one disc to another by reading portions of the first (master) disc into computer memory and writing each portion onto the second (backup) disc. Since only one disc drive is available, the program asks you to exchange the master disc for the backup disc one or more times. After you insert the master disc, the program determines how many disc exchanges will be needed to copy all files to the backup disc.

The “cbackup” program overwrites any files on the destination disc. The program also automatically initializes the destination disc if requested.

On a Model 226 Computer:

1. Load the HPL system. If HPL is already loaded, clear the memory by executing:

```
erase 
```

2. Insert the HPL Utilities Disc in the drive and close the door.
3. Load and run the cbackup program:

```
set "cbackup"  
```

Follow the displayed instructions.

Note

Be sure to press when instructed, not the key...otherwise, you must stop and rerun the program.

On a Model 216 or 236 Computer:

1. Load the HPL system.
2. Insert the source disc into drive 0, the right-hand Model 236 drive.
3. Insert the initialized destination disc into drive 1, the left-hand Model 236 drive.
4. On a Model 236:

```
COPY ":I,0","to",":I,1" EXECUTE
```

On a Model 216:

```
COPY ":M700,0","to",":I,1" EXEC
```

The last statement assumes unit 0 in a HP 8290x Disc Drive set to bus address 00.

More information on specifying external drives is in the Disc Programming section of the HPL Programming chapter.

Note

This manual shows keys available on both current Series 200 keyboards. The key symbol **EXECUTE**, for example, indicates a key on the large Model 226/236/optional 98203B keyboard. The equivalent key on the small, standard Model 216 keyboard is EXEC and is shown with the equivalent key like this: **EXECUTE** (EXEC).

Operating Features

The rest of this chapter introduces many of the computer's operating features, including the keyboard functions, display-control keys, arithmetic operations and printer controls. Whether you plan to run prerecorded (canned) programs or develop your own, first take a few moments to get acquainted with the computer by reading the next few pages.

Durability is a built-in feature of this easy-to-operate computer, so don't be afraid to test it. After reading each section and trying the examples shown, try your own examples. Experiment. You cannot damage the computer by pressing the wrong keys. The worst that can happen is an error message will appear.

Which Computer

There are several obvious differences between the Models 216, 226 and 236 computers as you can see in the following illustration. Since the two machines do have several differences, such as dual disc drives and an 80-column wide CRT on the 236, you may be writing a program which you want to run on either machine. The **machine** function is provided to determine which computer you are using. You may want to take advantage of this function to decide on how you write to the CRT, whether you are going to access the extra disc drive in the 236 and other similar operations. For details on the machine function, see Chapter 3.

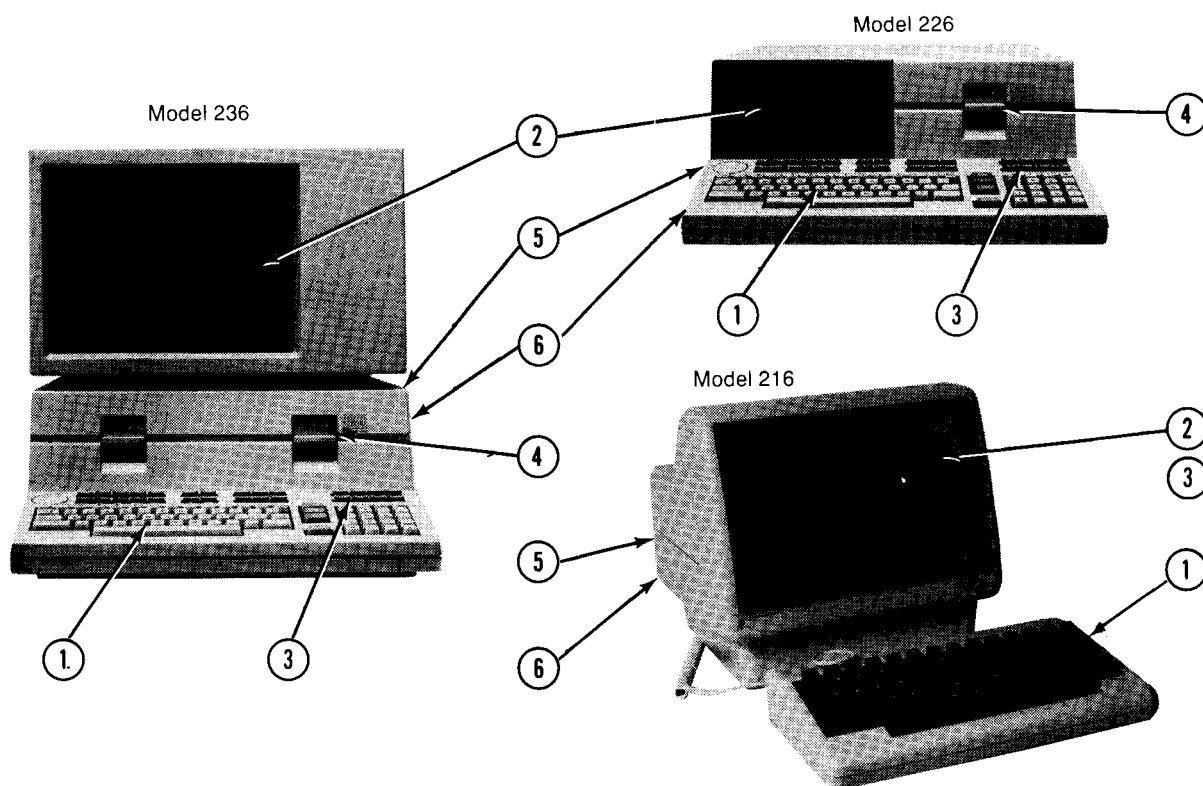
Program Autostart

You can have the computer automatically load and start running a program named AUTOSTH by inserting the disc in the disc drive before switching the computer on.

At power-on, the computer always checks for a disc in its drive. If a disc is inserted, the computer looks for a program file named AUTOSTH. If the AUTOSTH file is not found, the system searches for file named TOF000, and loads and runs it if present. This emulates tape cartridge autostarting. If the right file isn't on the disc, the computer simply displays the HPL READY message and awaits your command.

If your system is soft-loaded at power-up, the autostart disc must have both the SYSTEM-type file and an autostart file to automatically load and run both the language system and a program.

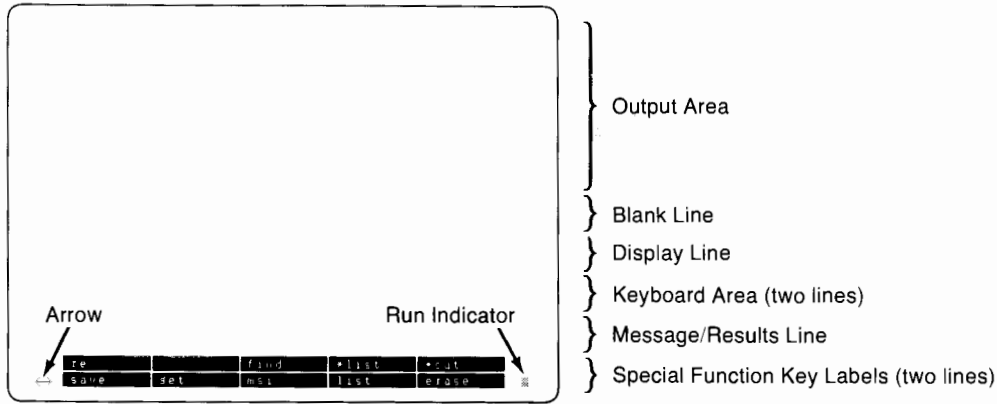
Computer Operating Features



- ① Easy-to-use Keyboard. The keyboard is arranged into logical groups for your convenience: character-entry keyboard, number-entry pad, display controls, system command keys, and program-defined keys called special function keys.
- ② An Organized Display. The display is partitioned into defined areas for maximum useability. One area, for example, is reserved for entering and executing keyboard commands, as shown later.
- ③ Graphics Display. The display can be set to either of two modes, normal alpha or graphics. The computer automatically sets the graphics mode under program control to display bar charts, x-y plots, etc.
- ④ Disc Mass Storage. The built-in disc drive uses standard 130mm (5-1/4 inch) discs for storing data, programs and other computer information. Each disc can hold about 1/4 million bytes (characters) of information.
- ⑤ Standard HP-IB Interface. A Hewlett-Packard Interface Bus (HP-IB) is built into the computer, allowing direct connection of up to 14 compatible instruments (printers, voltmeters, etc.). The programming language for controlling devices via the HP-IB is also built-in, allowing a program to control instrumentation systems and allows you to easily direct printouts, program listings, and displayed graphics to a printer or plotter via the bus.
- ⑥ Expandable Memory and Interfacing. In addition to the built-in HP-IB connector, a Model 226/236 has eight accessory slots. Each slot can hold a memory card. Each alternate slot can hold an interface card. The Model 216 has built-in HP-IB and RS-232 (serial) connectors, and has two additional accessory slots. Both slots can hold memory cards, or one slot can hold a memory card while the other slot holds an interface card.

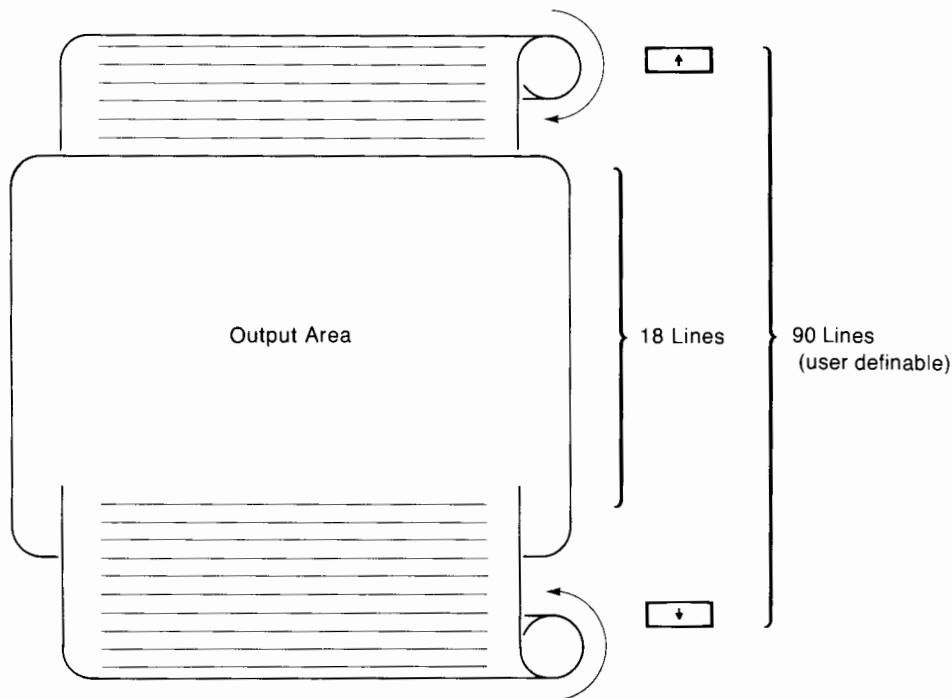
Display Organization

The Series 200 HPL system defines a 25-line work area on each computer display. The Model 216 and 236 computers have 80-character-wide display lines, while the Model 226 computer has 50-character-wide lines. HPL partitions the display the same on each computer:



The Run Screen Format

The **output area** can hold 90 or more lines of information, although only 18 lines appear on the display. Results of some keyboard and program output appear in this area. When the 18-line area is filled, the top lines scroll off into a buffer (holding) area of memory. To view these lines, use the cursor-control keys and the cursor wheel to scroll through the page. When the entire output area is filled, each new line entered causes a line to be lost off the top of the buffer.

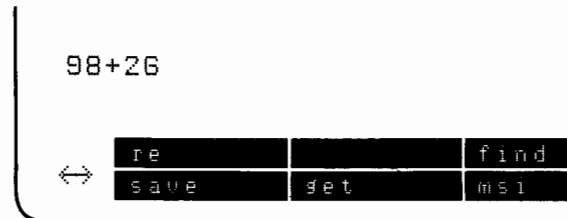


The Output Display Area and Buffer

The **display line** is reserved for instructions (prompts) from a program to the operator.

The **keyboard area** is where you enter responses to program prompts or type in commands. Press **CLR LN** (CLR L) to clear the keyboard area. Now do a simple arithmetic problem:

Type: 98+26
Press: **EXECUTE** (EXEC)



The operation is first entered in the keyboard area. When executed, the result replaces the operation in the message/results line.



Now try repeating the operation. Press **RECALL** (RCL) and **EXECUTE** (EXEC).

The results of keyboard operations always appear in the **message/ results line**. The results of some keyboard commands, however, like cat (cataloging a disc), appear in the display's output area.

The **special function key labels area** is reserved for labels which appear when one or more of the special function keys (**k0** thru **k9**) are defined. A program displays these labels when the special function keys are defined, as explained later.

The **run indicator** tells you what state the computer is currently in. When the indicator is blank, the computer is free and awaiting your command.

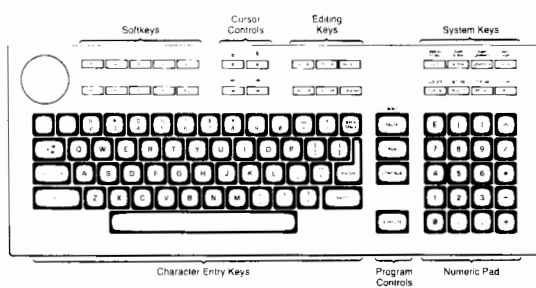
The **arrow** in the left-hand corner indicates the currently set direction of the cursor wheel, either up-and-down or left-and-right. Use the wheel with the cursor-control keys to rapidly position the display cursor.

Graphics Mode

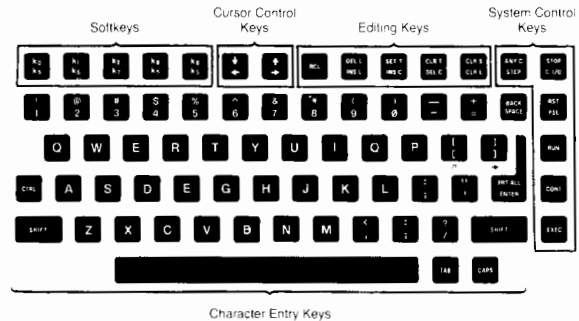
In addition to the normal 50-character by 24-line alpha mode, the display has a graphics mode for presenting charts, drawings and other pictorial representations. The graphicsmode can be automatically set when a program outputs graphic data on the display. You can switch back and forth, between graphics and alpha modes, byusing the Model 226/236 **GRAPHICS** and **ALPHA** keys. With the Model 216 standard keyboard, use the gon, goff, aon and aoff statements.

Keyboard Operations

The computer keyboard is arranged into functional groups for your convenience:



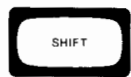
Model 226, 236 and 98203B Keyboard



Model 216 Standard Keyboard

Character Entry Keys

The character-entry keys are arranged like a typewriter, but have some added features.



You can enter the standard upper-case and lower-case letters using the **SHIFT** key to access the alternate case.



The **CAPS LOCK** (**CAPS**) key sets the unshifted keyboard to either upper case or lower case (for normal typewriter operation). The computer displays the mode set when you press the key.



The **ENTER** key has two functions: When a program is running, press **ENTER** to input data requested by the program. When a program isn't running, the programmer uses **ENTER** to store each line of program code.

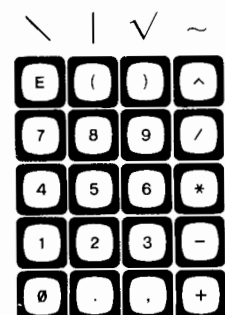


The **TAB** key is the HPL assignment operator key, "**→**". You can also use **SHIFT** - **[]**.



The **CTRL** (control) key works like **SHIFT** to access a set of standard computer-control characters, such as line feed (**LF**) and form feed (**FF**). These characters are useful to the programmer for controlling some devices and when communicating with other computers.

Numeric Pad¹

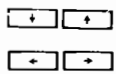


The numeric pad provides a convenient way to quickly enter numbers and perform arithmetic operations. Once each arithmetic problem (expression) is typed in, press **EXECUTE** to calculate and display the result.

For more details and example arithmetic problems, see Arithmetic Operations later in the chapter.

¹ These keys are not available with the standard Model 216 keyboard.

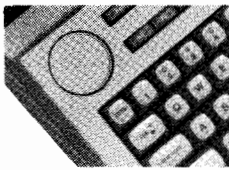
Cursor Controls



The cursor-control keys move the display cursor one space at a time. Press **←** and **→** to move the cursor to the end of the line. The **↑** and **↓** keys allow you to scroll information in the displayed output area up and down.



The **BACK SPACE** key works identically to the **←** cursor control key and is used to move the cursor 1 character back in the line.



The cursor wheel allows you to rapidly move the cursor up and down or back and forth, depending on the position of the little arrow in the lower-left of the display. You can change direction by pressing the an appropriate cursor-control key, **←**, **→** or **↑**, **↓**. Another way to change direction is by pressing **SHIFT** while rotating the knob.

Take a few moments to move the cursor around using the keys and the wheel. Notice that the computer automatically sets the direction of the cursor-wheel arrow after some operations.

Editing Lines



The editing keys put easy character and line editing at your fingertips.



Sets the insert-line mode, similar to the insert-character mode. Press **INS LN** (**INS LN**) again to cancel insert-line mode (edit mode only).



Deletes the line containing the cursor (edit mode only). If a line is deleted, it is stored in the recall buffer and can be recalled by pressing **RECALL** (**RCL**).



Recalls the last line entered or executed. Pressing repeatedly recalls up to 10 previous entries. Pressing **SHIFT** - **RECALL** (**RCL**) moves forward through this recall buffer.



Sets the insert mode, allowing you to insert characters to the left of the cursor. Press **INS CHR** (**INS C**) again to cancel the insert mode.



Deletes the character under the cursor.



Clears the end of the line, starting from the cursor position. Press **SHIFT** - **CLR+END** to clear the beginning of the line, starting with the cursor position.



Clears the entire line. Press **SHIFT** - **CLR LN** (**CLR L**) to clear the entire display screen.



Enters the `edit` command, allowing the programmer to use an editing mode for entering and editing program lines. (This is the 9825 **FETCH** key). To enter the edit mode with a standard Model 16 keyboard, execute: `edit` **EXECUTE**.

When a program is in computer memory, the program editing mode displays the program lines and waits for the programmer to scroll through, line by line, using the cursor wheel and cursor-control keys. Program changes are made by editing each line and pressing **ENTER**.

¹ This key is not available with the standard Model 216 Keyboard.

To exit the program edit mode, press the **PAUSE** (PSE) key. Edit mode is automatically exited when RUN, CLR SCR, or any other operation accessing the run screen printing area is executed.

```

0: aclr ;yclr
1: psc 16; scl 1,400,1,300; pen# 1
2: csiz 6
3: for I=90 to 91
4: for J=250 to 251
5: plt I,J; lbl "Display Program"
6: next J; next I

7: csiz 6 _

8: for I=10 to 11
9: for J=80 to 81
10: plt I,J; lbl "Alternate editing line"
11: next J; next I
12: l→P
13: pen# -P→P

```

re	find	*list	*cat
save	get	msl	list
			erase

Example Display During Program Edit Mode
(Replace-Line Edit Screen)


```

0: aclr ;sclr
1: psc 16;scl 1,400,1,300;pen# 1
2: csiz 6
3: for I=90 to 91
4: for J=250 to 251
5: plt I,J;lbl "Display Program"
6: next J;next I

—

7: csiz 6
8: for I=10 to 11
9: for J=80 to 81
10: plt I,J;lbl "Alternate editing line"
11: next J;next I
12: l>P
13: pen# -P>P

```



re		find	*list	*cat
save	get	msl	list	erase

Example Display During Program Edit Mode
(Insert-Line Edit Screen)

An Editing Exercise

Now let's quickly type in a few lines and go back to make some corrections. To first clear the screen, press **(SHIFT)** - **(CLR LN)** (**CLR L**). Here's our first try:

```

Using the editing keys makes correctins lines of
text as esy as pie.

```

re		find	*list	*cat
save	get	msl	list	erase

Original Line
(Model 236 shown)

Not bad, we only misspelled one word in the first line "editting". Move the cursor to the extra "t" in "editting" using the **(←)** and **(→)** and the knob; then press **(DEL CHR)** (**DEL C**) once.

The second line needs a character added to “esy”. First move the cursor to the “s” in “esy”, press **INS CHR** (**INS C**) and insert the missing “a”. To cancel the insert mode, press **INS CHR** (**INS C**) once again.

```

Using the editing keys makes correcting lines of
text as easy as pie.
re          find  *list  *cat
save       det   mai   list  erase
  
```

Corrected Line

Please notice that this is only a practice exercise. You can't **EXECUTE** or **ENTER** the lines; the computer wouldn't recognize them as either a command or a program line. It would, however, beep and display an error message. Go ahead and try it: it won't be the last time you'll hear that beep!

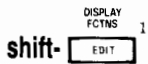
System Control Keys



The keys in the upper-right corner control various system functions related to the display, printer and editing operations. Most of these keys execute their functions immediately, as the key is pressed.



Enters the `edit` command. The programmer uses the edit mode when entering and editing programs.



Sets a display-functions mode, allowing you to see special control characters such as line feed (lf) and carriage return (cr) in the display output area. Press **SHIFT-EDIT** again to cancel the display-functions mode. A complete set of display-functions characters is shown in the ASCII table at the back of this manual.



These keys allow you to view either one or both of the display modes, normal alpha or graphics. For example, if a program sets the graphics mode and outputs a graphics display, you can return to the alpha mode by pressing **ALPHA**. You can later return to the graphics mode by pressing **GRAPHICS**.

Pressing **ALPHA** or **GRAPHICS** once sets that mode but doesn't reset the other mode. Pressing the key a second time resets the other mode. So you can view both display modes simultaneously if you wish.



Enters the `adump` command. Pressing **EXECUTE** causes the complete alpha display to be output to the current prtsc device. See the HPL Programming chapter for details.



Enters the `gdump` command. Pressing **EXECUTE** causes the graphics display to be output to the current prtsc device. See the HPL Programming chapter for details.



Allows the programmer to step through a program, one line at a time. Using **STEP** to debug programs is covered in the 9825 Operating and Programming Reference.

¹ These keys are not available on the standard Model 216 keyboard.

shift- ANY CHAR
STEP (ANY C)

Shows each available display character. First press **SHIFT** - **STEP** (ANY C). Then enter any three-digit number from 000 thru 255. The computer automatically displays the equivalent character. The programmer uses this function when developing programs. A table of characters and their decimal values is in the ASCII table in Appendix B.

CLR LN (CLR L)

Clears the display input line and the message/result line.

shift- CLR SCR
CLR LN (CLR S)

Clears the entire alpha display, including the input line and the scrolling buffer, and turns on key labels.

RESULT¹

Enters `res`, for the result function. When executed, returns the result of the last expression executed. For example, press **CLR LN**. Then enter:

23 + 45 **EXECUTE**

68.00

↕	re		find
	save	get	msl

Now return the result to the keyboard line and add 123 to it:

RESULT + 123 **EXECUTE**

191.00

↕	re		find
	save	get	msl

PRT ALL

Toggles the printall mode on or off, allowing keyboard operations and displayed error messages to be copied to the system printer. Press **PRT ALL** once to set printall ON and again to set printall OFF. Since the display is automatically set as the system printer at power up, the printall mode can be used to log all keyboard operations in the display's output area. Setting the system printer is explained in the HPL Programming chapter.

shift- STOP
CLR I/O

Halts program execution after the current line is executed, as when pressing **PAUSE** (PSE). To re-start the program, press **RUN**.

¹ This key is not available on the standard 216 keyboard.

Special Function Keys

The ten keys labeled **k0** through **k9** are Special Function Keys. These may be defined and labeled by the operator as typing aids, immediate execute keys, or immediate continue keys. They may also be defined from a program.

Another 22 SFKs can be defined at the same time, but without displayed labels. As explained later in this section, the `sfk` statement can define **k0** thru **k31**. Then **k0** thru **k9** can each be accessed from the keyboard. With all but the standard Model 216 keyboard, you can also access **k10** thru **k19** using and **k20** thru **k29** with . **k30** and **k31** can only be accessed using the `pkbd` (press keyboard) statement. With the standard Model 216 keyboard, however, **k10** thru **k31** must be accessed using `pkbd`.

Labelled Special Function Keys

When a special function key is defined, a label can be assigned to that key as a part of the key's definition. Key labels are displayed on the bottom two lines of the CRT, in positions corresponding to the associated special function key.

To include a label as part of a special function key's definition, type the desired text (up to 10 characters on the Model 226 and 16 on the Model 216 or 236) within quotes, followed by a colon, followed by the remainder of the key definition. Some examples will help to clarify this:

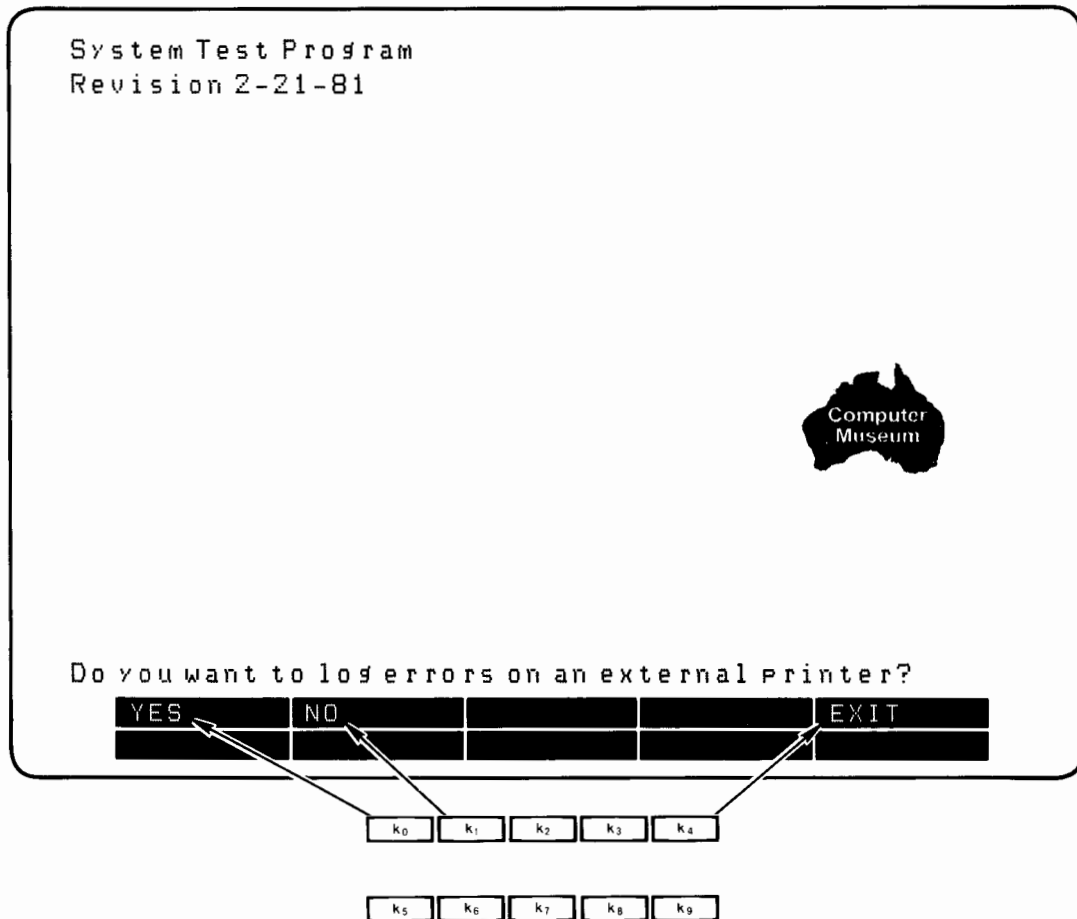
Press: or type EDIT
 Type: "Label":definition
 Press:

Notice that `Label` is now the "soft" key label associated with .

Other labelled special function key examples:

```
"Recover" :*cont "Restart"
"Abort" :*sf$ 30;% "Set shutdown flag"
"LogE base":/2,71828182846
```

Since the computer can offer a wide selection of operations with each set of defined special function keys, the set of key labels is often called a menu. Here's one of the menus available with the System Test program:



With the standard Model 216 keyboard, k0 thru k4 are accessed using **SHIFT**.

Immediate Execute Special Function Keys

If a line to be stored under a special function key is preceded by an asterisk (*), it is an immediate execute key. This means that when the key is pressed, the contents of the key are appended to the display and the line in the display is executed automatically.

For example:

Press: **EDIT** **k3**

Accesses **k3**.

Type: *P r t " " , π

The asterisk makes this an immediate execute key.

Press: **ENTER**

This stores the line entered in the display under **k3**.

Whenever **k3** is pressed and the display is clear, the following is printed:

3.14

Immediate execute keys are useful for executing selected segments of a program. Using the continue command followed by a line number, you can make several entry points in your programs. For example:

k2 : *cont 5

k3 : *cont 10

Each time **k2** is pressed, the program continues at line 5, or at line 10 if **k3** is pressed.

Immediate Continue Special Function Keys

If a line to be stored as a special function key is preceded by a slash (/), it is an immediate continue key for use with the enter statement. "Immediate continue" means that when the key is pressed, the contents of the key are appended to the display and continue is executed automatically. Immediate continue keys are used to enter often used values in enter statements. For example:

Press: **EDIT** **k3**

Fetches special function key **k3**.

Type: /2.71828182846

This enters the value of e , the base of the natural logarithms, into the display.

Press: **ENTER**

This stores the line in the display under **k3**.

Whenever an enter statement is waiting for a value and the **k3** key is pressed, the approximate value for e (i.e., 2.71828182846) is entered and the program continues.

Keys with Multiple Statements

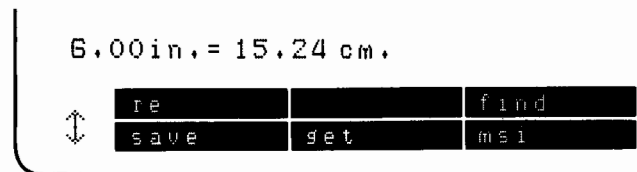
By separating statements with semicolons, several statements can be stored under one special function key. As an example, suppose you want to convert inches to centimetres. The following line is stored under special function key **k2**.

Press: **EDIT** **k2**

Type: *↵R;dSPR,"in,=",2.54R,"cm, "

Press: **ENTER**

Then key in a number, such as 6, and press **k2**. The display will show:



Extended Control Special Function Keys

If a special function key definition is preceded by an EOL character (decimal 127: use the **ANY CHAR** (**ANY C**) key), it is an extended control key. This means that when the key is depressed, the contents of the key are effectively “pushed” on the keyboard. If the key definition is standard alphanumeric characters, these are “typed” into the display. However, the key definitions can also be control keys such as **←**, **INS CHR** (**INS C**), and **ENTER**. These keys are accessed by pressing **CTRL** and the command key simultaneously. This allows you to define some very powerful special function keys. For example, the following key is a labeled “Comment” key, useful when editing programs:

Press: **EDIT** **k1**

Access SFK 1

Type: "Comment":

Label it as a “comment” key

ANY CHAR

Press: **SHIFT** - **STEP**

Any char of 127 is the EOL char

Type: 127

Press: **CTRL** **SHIFT** - **→**

Home right

Press: **CTRL** **INS CHR** (**INS C**)

Take out of insert-character mode.

Press: **CTRL** - **SHIFT** - **←**

Home left key

Press: **CTRL** - **INS CHR** (**INS C**)

Insert character mode

Type: %"

Percent symbol, first quote

Press: **CTRL** - **SHIFT** - **→** Home right key

Type: " Second quote

Press: **CTRL** - **ENTER** Store key

Press: **ENTER** Key Defined!

Now, any text typed on the keyboard can be entered as a comment line by pressing special function key **k1**, or "C o m m e n t".

Default Special Function Keys

There are several special function keys predefined by Series 200 HPL. The meanings and use of these keys are discussed in this section. Note that although several keys are predefined at power-up and whenever **sfk** (with no parameters) is executed, they may be defined by the user at any time with **EDIT** and the **sfk** statement.

re **k0** This is an immediate execute key that is appended to the FRONT of the keyboard line which is then executed. Typically, this key would be used to resave a program by first pressing **k5** (the save key), typing in the program name between the quotes, then pressing **k1** to resave the program.

Press: **k5** Displays: `save ""` with insert cursor

Type: NAME Now looks like: `save "NAME"`

Press: **k0** Now looks like: `resave "NAME"`

Automatically executes `resave "NAME"`





find **k2** This is a typing-aid key that displays `find " "` on the keyboard line, with an insert cursor between the quotes. The characters that you want to search the program for go inside the quotes.

***list** **k3** This is an immediate execute key that lists the program in memory to the current system printer.





***cat** **k4** This is an immediate execute key that catalogs the system disc (set by `msi` or drive) to the current system printer.

save **k5** This is a typing-aid key that displays `save ""` on the keyboard line, with an insert cursor between the quotes. The file name of the program to be saved is then typed (and goes between the quotes). When **EXECUTE** (**EXEC**) is pressed, the program in memory is saved to the current system disc drive (set by `msi` or drive).

set **k6** This is a typing-aid key that displays `set ""` on the keyboard line, with an insert cursor between the quotes. The file name of the program to be loaded is then typed (and goes between the quotes). When **EXECUTE** (**EXEC**) is pressed, the specified program is loaded into memory (if one exists on the disc).

- m s i**  This is a typing-aid key that displays `m s i " : "` on the keyboard line, with an insert cursor between the colon and the second quote. The mass storage unit specifier is then typed (and goes between the colon and the quote). When  is pressed, the disc specified becomes the system mass storage device.
- l i s t**  This is a typing-aid key that displays `l i s t` on the keyboard line. Its use is essentially the same as the **LIST** key was on the 9825.
- e r a s e**  This is a typing-aid key that displays `e r a s e` on the keyboard line. Its use is essentially the same as the **ERASE** key was on the 9825.

Program Control Keys

The , ,  and  keys allow you to control execution of the program stored in the computer's memory.



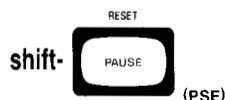
Starts program execution from the beginning.



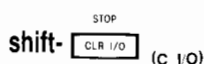
Pauses program execution after the current line, returning computer control to the keyboard.




Resumes program execution from where it was paused, or enters data for an active `ent` or `enp` statement.



Stops program execution immediately without erasing the program or data memory. The HPL READY message indicates that the computer is ready for your command. This is the equivalent of the 9825 **RESET** key.



Stops program execution after the current line, as by pressing .



Arithmetic Operations

The arithmetic operators are located within the numeric keypad. Their operations are listed in the arithmetic hierarchy of HPL.

- sqr or $\sqrt{\quad}$ Square root (shift of \square)
- \wedge Exponeniation
- no operator Implied multiplication
- $*/$ Multiplication and division
- $+ -$ Addition and subtraction

If you prefer, you can also use the same characters found within the typewriter keyboard.

To perform arithmetic operations, first clear the display's "keyboard" area by pressing \square (CLR LN) (CLR L). Then simply type-in the problem and press \square (EXECUTE) (EXEC). Try these examples:

First, how many characters can be entered into the 9826 computer's complete 50-character line by 18-line by 5-page alpha-display area?

Enter: 50*18*5

Press: \square (EXECUTE) (EXEC)

```

50*18*5
4500.00      (characters)
↑↓
┌───┬───┬───┐
│ r e │   │ f i n d │
├───┴───┴───┘
│ s a v e │ g e t │ m s l │
└───┬───┬───┘

```

If you spend 3% of your time today reading this manual, how much of your eight-hour workday is left for work?

Enter: 8-8*.03

Press: \square (EXECUTE) (EXEC)

```

8-8*.03
7.76      (hours left)
↑↓
┌───┬───┬───┐
│ r e │   │ f i n d │
├───┴───┴───┘
│ s a v e │ g e t │ m s l │
└───┬───┬───┘

```

If the floor in your office is square, with each side measuring 6.2 metres, how many square metres of carpeting are needed to cover it? You can either multiply $6.2 * 6.2$, or you can find the square of 6.2 by raising it to the second power ($6.2 \uparrow 2$):

Enter: 6.2^2 \square (EXECUTE) (EXEC)

```

6.2^2
38.44      (square metres)
↑↓
┌───┬───┬───┐
│ r e │   │ f i n d │
├───┴───┴───┘
│ s a v e │ g e t │ m s l │
└───┬───┬───┘

```

Notice, in each case, that the computer displays the result in the line below where you entered the problem. This allows you to either recall the problem (press **RECALL**) and compare it with the result, or recall the result (press **RESULT**) and use it as part of another problem. For example:

Enter: $98+26$ **EXECUTE**

124.00

Enter: $25*$ **RESULT**

25*res

Press: **EXECUTE**

3100.00

Enter: **RESULT** /37

res/37

Press: **EXECUTE**

83.78

↕	re	find
	save	get mem

Note

The **RESULT** key is not available on the standard Model 216 Keyboard.

Notes

Chapter 2

Program Transfer

Introduction

This section addresses a major concern of 9825 owners and programmers: how to get 9825 programs and data into the new computer. There are two approaches possible: one, simply save the 9825 programs and data on a 9885 or 9895 disc, then reconnect the disc drive to the Model 216, 226 or 236 computer and read the programs and data in. This approach is a natural for those users already using the disc extensively. The second approach uses I/O to transfer programs and data, and requires a bit more coordination between the two computers (and work on your part). The best interface to use to avoid complications from losing data is the HP-IB interface, which is built into the new computer and is the 98034 interface card for the 9825.

Disc Transfer

The 9885 and 9895 discs offer the simplest means of transferring programs and data from the 9825 to the Model 216/226/236 (and vice-versa, if desired). In addition, disc file transfers offer the only method of transferring key files from the 9825 to the new computer. Once the disc drive is installed on the new computer, the same commands that were used on the 9825 can be used. For example, to load a 9825 program "Prog1" into the new computer (from a 9885 disc, unit number 0 at select code 8), the following statements could be used:

```
drive 0,8  
set "Prog1"
```

The above sequence can be shortened by using the new mass storage unit specifier (msus) available on the new computer:

```
set "Prog1:FB,0"
```

Disc data files can be accessed in the same manner as from the 9825, using the same program statements. One additional statement may need to be added to your programs if they assume default drive parameters of unit number 0, select code 8. To access an external disc drive, a drive or msi statement must be executed stating the unit number and select code of the disc drive. (The default drive on the Model 226 is the internal minifloppy and the internal drive on the right side on the Model 236.)

There are additional disc programming capabilities offered by the Series 200 computers. You may wish to refer to the HPL Programming chapter of this manual for these disc programming enhancements.

Interface Transfer

Interface transfers offer a means of getting programs and data from a tape-based 9825 system into the Series 200 computers. The concept is fairly straightforward, and there are just a few minor details to take care of to implement the transfer. To keep matters simple, the example presented assumes an HP-IB interface for the transfer operation.

Programs

The actual operation is a simple `list # 731` of the program from the 9825, and a short program on the Series 200 computer that reads in the program and stores it into memory. You then save the program on a disc.

The first step of the process is to connect the two desktops together and power them up. The 9825 requires the 98034 interface, which is connected to the built-in HP-IB interface of the Series 200 computer. The 98034 is assumed here to be set at factory default switch settings: most importantly System Controller and bus address 21.

From the 9825, execute `Pct 731` to pass controller functions to a non-existent bus device. This makes disassembly and reassembly of the 98034 unnecessary.

Now from the Series 200 computer execute `wtc 7,30` to change its bus address so there won't be a conflict with the 9825's bus address. (Both have bus addresses of 21, unless the hardware switch settings from the factory have been changed.)

On the 9825, load in the desired program (for example, `trk1;ldfB`). Then RUN the following program on the Series 200 computer:

```
0: "Loader":dim A#[85];inl→N;721→S
1: "input":red S,A#;if len(A#)≤2;sto "input"
2: if A#[1,1]="*";sto "done"
3: "store":on err "error";store A#,nal
4: sto "input"
5: "error":"%"&A#→A#;sto "store"
6: "done":red S,A#,A#,A#,nal→X
7: Prt "Record the program on the desired track and file"
8: Prt "(or file name), from line N through line X."
9: Prt "For example, 'trk1;rcf B,N,X'. Then delete the"
10: Prt "old program and re-run this program for the"
11: Prt "next program to be transferred.";ifxd 0
12: Prt "To delete the saved program, use 'del",N,",",X,"/"
13: end
*17613
```

On the 9825, execute `list #731`. The 9825 program is now transferred.

Data

Now that you have seen the program transfer process, you must consider the data transfer process. The machine set-up for data transfers is identical to that used to transfer programs, but now you need a program in each machine. The programs must be specifically tailored for the particular data files to be transferred. A simple example here will illustrate the concept. The task is to transfer ten 3000 character string files from the 9825 tape to the Series 200 computer's disc. The string files will be saved in tape-file format for maximum simplicity.

9825 Program

```
0: dim A$[3000];% "This line changes according to your own data structure"
1: trk 1;for I=0 to 0;% "Again, modify according to your data structure"
2: ldf I,A$;wrt 731,A$
3: next I;prt "Transfer complete.";end
```

Series 200 HPL Program

```
0: dim A$[3000];% "This line must be identical to the 9825 program line 0"
1: trk 1;for I=0 to 0;% "Again, this must agree with 9825 program above"
2: red 721,A$;rcf I,A$;% "This line is the only one different"
3: next I;prt "Transfer complete.";end
```

Both programs are essentially the same except that line 2 of each is suited to the particular function that is being performed: the 9825 is taking data from the tape and sending it to the new computer; the new computer is reading in the data from the 9825 and saving it on the disc.

Notes

Chapter 3

HPL Programming



Introduction

This chapter summarizes the enhancements and differences between 9825 HPL and Series 200 HPL. The entire Series 200 HPL language is listed in the *HPL Condensed Reference*, 98614-90020.

To help you learn the differences, this chapter is organized by 9825 ROM function. The information in parentheses accompanying most section titles refer to a specific 9825 manual:

O&P	<i>Operating and Programming Reference</i> , 09825-90200.
I/O	<i>I/O Control Reference</i> , 09825-90210.
Disc	<i>Disc Programming Manual</i> , 09825-90220.
Matrix	<i>Matrix Programming Manual</i> , 09825-90022.

Mainframe Programming

The Read, Data, and Restore Statements

To assign constant values to string and numeric program variables, the `data` and `read` statements can be used.

```

data string or numeric constant [ ,string or numeric constant]...
read variable name [ ,variable name]...
rstr [label]

```

The `data` statement provides string and numeric constant values to be assigned to program variables. String constants may be quoted or unquoted. Each constant is separated from the next by a comma. The `read` statement assigns the constants provided by data statements to variables in the read statement variable list, one constant per variable. As each constant is read from the data list, the data pointer is positioned to the next constant in the data list. This data pointer can be reset either to the first data statement occurring in the program or to the specified line label by using the restore statement, `rstr`. A short example will help clarify this concept:

```

0: dim A$[20]
1: data 100,"item 1",200,"item 2",300,"item 3"
2: "line 2":data 400,"item 4",500,"item 5",600,"item 6"
3: data 700,"item 7",800,"item 8",900,"item 9"
4: for I=1 to 9:read X,A$:prt X,A$
5: if I=6:rstr "line 2"
6: next I:end

```

Press: RUN

```

100.00 item 1
200.00 item 2
300.00 item 3
400.00 item 4
500.00 item 5
600.00 item 6
400.00 item 4
500.00 item 5
600.00 item 6

```

Note that after the sixth data item pair was read, the `rstr` statement was used to reset the data pointer to "line 2", data item 4. If line 5 of the program is changed to

```
5: if I=6:rstr
```

the printout looks like this:

```

100.00 item 1
200.00 item 2
300.00 item 3
400.00 item 4
500.00 item 5
600.00 item 6
100.00 item 1
200.00 item 2
300.00 item 3

```

The data pointer was reset all the way back to the first line of the program, which meant that the next data item read was data item 1 on line 1.

The Programmable Beep Statement (O&P p.3-16)

```
P B E E P [frequency[ ,duration]]
```

This statement drives the internal beeper with a programmable frequency and duration. Allowable frequencies range from 0 Hz through 5167 Hz. The frequency is taken as a modulus of 81.23 Hz, which means it is “rounded” to the nearest multiple of 81.23 Hz. Allowable durations range from 0 through 2.56 seconds. The number of seconds duration is rounded to the nearest hundredth of a second, (.01 sec).

The Machine Function

This `machine` function returns a value that characterizes the internal configuration of the computer.

bit (0, machine)	0 = 80 column alpha (Model 216 or 236) 1 = 50 column alpha (Model 226)
bit (1, machine)	0 = 400x300 pixel graphics (Model 226) 1 = 512x390 pixel graphics (Model 216 or 236)
bit (2, machine)	0 = CRT has no highlights (Model 226) 1 = CRT has highlights (Model 216 or 236)
bit (3, machine)	0 = machine has a keyboard 1 = machine has no keyboard

The first three of these examples can be used to determine which Series 200 computer is currently in use.

CRT Display Control (O&P p. 3-16)

The Clear Alpha Statement

```
a c l r [number of scrolling pages]
```

The `aclr` statement clears the alpha screen and optionally assigns the number of scrolling pages for the scrolling buffer. If not specified, the number of scrolling pages selected is left unchanged from the previous value (four at power-up).

On the Model 226, 900 bytes (18 lines of 50 characters) of read-write memory is used for each scrolling page. On the Model 216 and 236, 2880 bytes are used per scrolling page. The reason that so much more memory is used is because two bytes are reserved for each character cell; 1 byte for character and 1 byte for the highlight (e.g. blinking, see `crt` statement). This means that 18 lines of 80 characters per line times 2 bytes per displayed character is used — or 2880 bytes of memory.

The following program line sizes the scrolling buffer according to the amount of memory remaining, leaving enough room (1800 bytes or more) for most stack operations. Remember, large string operations require that enough memory be available for the operation!

```
avm/(900bit(0,machine)+2880(1-bit(0,machine)))
```

When `aclr` is executed:

- the graphics screen is unaffected
- key labels are displayed
- the RUN screen format of the CRT is selected as opposed to the Edit screen
- Display Functions is turned off

Highlighting on Model 216 and 236

```
c r t value
```

The `crt` statement is used to control the video highlight scrolling capabilities on the Model 216 and 236 CRT only.

The following values are used:

- | | |
|--------------------------------------|---|
| 0 – normal | 8 – halfbright |
| 1 – inverse video | 9 – inverse and halfbright |
| 2 – blinking | 10 – blinking and halfbright |
| 3 – inverse and blinking | 11 – inverse, blinking and halfbright |
| 4 – underlining | 12 – underlined and halfbright |
| 5 – inverse and underlined | 13 – inverse, underlined and halfbright |
| 6 – blinking and underlined | 14 – blinking, underlined and halfbright |
| 7 – inverse, blinking and underlined | 15 – inverse, blinking, underlined and halfbright |

The Alpha On, Alpha Off Statements

```
aoff
aon
```

The `aoff` statement turns off the alpha display without affecting the contents of alpha memory. When an `aon` statement is executed, the alpha screen is displayed. All screen operations (`prt`, `dsp`) function normally whether the display is turned on or off. Key labels (SFk labels) are not affected - see the `kloff` and `klon` statements.

The Key Labels On/Off Statements

```
kloff
klon
```

The `kloff` statement turns off the display of special function key labels. The contents of the labels are not affected, so that executing a `klon` statement causes the original labels to be displayed. (Note that the `sfk` statement - Systems Programming - can be used to define SFK's labels from a program.)

The Dump Alpha Statement

```
adump[select code or buffer[ ,number of lines]]
```

The `adump` statement sends the specified number of alpha display lines to the desired select code or buffer. If no parameters are specified, the first 18 lines of the alpha screen are sent to the system printer (`PRtSc` device). Any defined key labels are not dumped.

The Tab X-Y Statement

```
tabxy column , row
```

The `tabxy` statement directs CRT printing (and reading!) operations to the character and line position of the screen as specified by the "column" (character) and "row" (line) parameters. The column parameter must be within the range of 0 through 49 on the Model 226 and 0 through 79 on the Model 216 and 236. The row parameter should be within the range of 0 through 17. Specifying a row greater than 17 causes the row parameter to be truncated to 17 (you cannot tab below the bottom of the screen). Column 0, row 0 is the top left-hand corner of the CRT.

The Read CRT Statements

```
read 16[ ,format number] , variable list
rdb ( 16 )→variable
```

Data can be read from the CRT in much the same manner as from an external device. Data is taken from the CRT at the present cursor position and assigned to variables in the variable list according to any formatting in effect. Read binary (rdb) operations from the CRT operate in the same manner as for external devices.

Note

There are no “hidden” control characters on the screen. Unless “Display Functions” is on, control characters such as carriage-return and line-feed are not part of the CRT data. Also, reading past the end of scrolling memory simply returns blanks (decimal 32).

The System Printer Select Code Statement

```
prtsc select code or buffer[ ,width]
```

This statement directs all print operations to the specified select code or buffer. This affects statements such as `prt`, `tlist`, `aprt`, `spc`, `listk`, `cat`, `list`, and also directs PRINT ALL messages. The optional “width” parameter determines the number of characters to be printed before a carriage-return/line-feed is sent by the `aprt` and `listk` statements. Powerup default for `prtsc` is to the CRT, width = 50 on the Model 226 and width = 80 on the Model 226 and 236. (No, execution `prtsc 16,80` will not cause 80 character lines to be displayed on the Model 226 CRT, and `prtsc 16,16` will not emulate the 9825 strip printer!)

Math Functions (O&P p. 3-22)

`sqr` expression

Returns the square root of a non-negative expression. Identical to the $\sqrt{\quad}$ operator.

`pi`

Returns the value of pi (π). Identical to the π function.



Flags (O&P p. 3-28)

There are 32 program flags available, numbered 0 through 31. Flags 0 through 15 are identical to 9825 program flags, with special meanings for flags 13, 14, and 15. Flags 16 through 31 are strictly user program flags, not affected by the system.

The Cross Reference Statement (O&P p. 4-32)

The `xref` statement now lists references to p-numbers.

The Find Statement

`find` string expression [,line number₁ [,line number₂]]

To list all of the lines that contain the specified character string, the `find` statement is used. The list of these program lines is output to the current `prtsc` device.

If no line numbers are given, the entire program is searched. If a single line number is given, the program beginning from the specified line to the end of the program is searched. If both line numbers are given, the range of program lines is searched.

The search is not done on the line number nor the colon that follows the line number but does include the space following the colon.

To search for quotes, be sure to follow the double quote convention. For example to find all lines that contain a label you could execute:

```
find "Δ""
```

where the Δ character represents a blank. The doubled quotes will be considered a single quote.

At execution time, this must be the last statement in the line.

Tape Cartridge Operations (O&P chapter 5)

A program utilizing tape cartridge mass storage operations will in general be able to run unmodified on an HPL 2.0/2.1 operating system. However, do not try to insert your tape cartridge into the disc drive!

The actual method used to implement tape operations on a disc is to create files on the disc that correspond to 9825 marked tape files. This is done automatically by the operating system: you don't have to worry about how to do it. However, this has some implications that you should be aware of.

- You can do tape operations to any disc drive supported by HPL 2.0/2.1: 9885, 9895, 8290x, and the internal drives. Tape operations are directed to the current disc drive, set by drive or msi (see Disc Programming).
- The files accessed by the tape statements are more or less ordinary disc files which have been named according to a special convention. This convention was established so that given a track number and file number, the file name is uniquely defined. The convention is:

TxFyyy

where x is a single-digit track number, 0 or 1 and y is a three-digit file number, 000 to 999. An example tape file name, track 0 file 15 is:

T0F015

- Three new disc file types have been created to allow tape statement emulation. These new types are:
 - NULL (tape file-type 0): file with current size of 0
 - NBDATA (tape file-type 2): file with numeric data
 - SBDATA (tape file-type 3): file with string or mixed data
- The mrk and ert statements do not initialize the disc media, rather they simply create and purge disc files on an initialized disc. Therefore, an uninitialized disc must first be initialized (with an `i n i t` statement) before the tape files can be marked.
- The format for a tlist printout has been modified to take advantage of the new wider CRT. A tlist now indicates the secured/unsecured status (scd) of the program, and the file type is now listed as a mnemonic rather than a number.

Track#	Secured Program Indicator	Current msi Device		
trk	scd	msi	(#)	cur_size abs_size
file	type	":I,O"		
=====				
#0	* PROGRAM		(6)	432 512
#1	KEYS		(5)	46 256
#2	NBDATA		(2)	304 512
#3	NULL		(0)	0 0

File# → #0, #1, #2, #3

File Type Mnemonic → PROGRAM, KEYS, NBDATA, NULL

File Type Number (same as value returned by idf) → (6), (5), (2), (0)

Current File Size (bytes) → 432, 46, 304, 0

Absolute File Size (bytes) → 512, 256, 512, 0

- Disc files are always sized as a multiple of 256 bytes per record. Thus, any tape file will necessarily have to be rounded up to a multiple of 256 bytes, even if you specify it to be less. (For instance, although you executed “mrk 1,50”, you actually got one file of 256 bytes.)
- Due to the increased memory size of the Series 200 computer, recording strings (rcf N,A#) requires two extra bytes overhead per string on a file. Usually, this will cause no problem for an old 9825 program being run on a new computer, because all files are rounded up to a multiple of 256 bytes.
- A `t l i s t` will list only tape files of the tape file-name convention, and they will be listed in order by file number, regardless of their order in the disc directory.
- A `c a t` (catalog) of the disc will list both tape files and disc files. The order of the file names in the directory is unimportant. Both tape and disc files are allowed on the same disc.
- Auto-verify (avd, ave) affects only tape statements, as expected. Verify (von, voff) affects only disc statements. Defaults at power-on, reset, and erase a, are ave and voff. These are identical to the 9825 defaults.
- Record and load memory (rcm, ldm) are not implemented in HPL 2.0 and 2.1. (Imagine rcm on a 4 megabyte machine!)
- Disc statements can access files created with tape statements (mrk) and tape statements can access files created with disc statements provided they conform to the tape file-name conventions listed above.

Note

It is highly recommended that only tape statements be used to access tape files and only disc statements be used to access disc files! There are a number of subtle points affecting file access that must be considered, as described above. If you wish to mix statement and file types between tape and disc operations, it is suggested that you do so on non-critical data at first, until you feel confident that you understand the interactions! Refer to the discussion on “Binary Data File Support” in the “Disc Programming Technical Appendix” at the back of this manual for more details on tape files on disc.

Record Binary Statement

`r c b` file number

The record binary statement stores the binary program in memory onto the specified tape file. The file number is any numeric expression, and if omitted file 0 is assumed.

String Variables

Value Function (O&P p. 6-17)

`v a l` (string expression[,base])

By specifying an optional second parameter “base”, string expressions representing numbers of any base (2 thru 36 inclusive) can be converted to base 10.

String Function (O&P p. 6-19)

`s t r` (numeric expression[,base])

By specifying an optional second parameter, “base”, numeric expressions can be converted from base 10 to a string expression representing a number in any other base (2 thru 36 inclusive).

Examples: `v a l ("FF",16) ↪ A` `A = 255`
 `s t r (10,2) ↪ A$` `A$ = "1010"`

Read Statement (O&P p. 6-32)

The general I/O `r e d` statement now allows substrings as data destination variables.

Example: `r e d 3 , B $ [15 , 250]`

Systems Programming

Intelligent Terminal Instructions (O&P p.7-7)

Additional capabilities for program emulation of intelligent terminals are provided beginning with HPL 2.0. In addition to program response to keypresses, the program can also respond to rotation of the Rotary Control Knob.

The On Knob Statement

```
on knob [label],[rate]]
.
.
.
kret
```

This statement zeroes the knob-count, then enables program interrupts from the Rotary Control Knob. Whenever the knob is rotated, the knob-count accumulates at a rate of 120 units per revolution and an end-of-line branch to the on-knob service routine is taken. The accumulated knob-count can be accessed via the `knob` function, which indicates both direction and amount of rotation. The on-knob service routine can also access the status of the **CTRL** and **SHIFT** keys (for shifted-knob and control-knob functions) via the `kstat` function. The on-knob service routine exits with the `kret` statement, which returns control back to the main program.

The rate is an optional parameter that specifies the number of seconds between knob interrupts. The legal range is from 0.01 (default) to 2.56 seconds. The default value of 0.01 seconds is also set when a program is not running. When a program is resumed, the previous set rate is again used.

The Knob Function

```
knob
```

The `knob` function returns the accumulated count of the RPG (Rotary Pulse Generator), or knob count, indicating both direction and degree of rotation. The knob count is approximately 120 units per revolution, or three degrees per unit count. This is an approximate count, suitable for relative positioning information only. (Counts may be missed when the knob is spun very rapidly.)

Negative values for `knob` indicate counter-clockwise rotation. Positive values for `knob` indicate clockwise rotation.

The Knob Status Function

`kstat`

The `kstat` function returns an 8-bit knob status. This is a self-zeroing function that zeroes when read and is set when an on-knob end-of-line branch is taken.

The meanings assigned to `kstat` bits are as follows:

Most Significant Bit				Least Significant Bit			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	1	$\overline{\text{Control}}$	$\overline{\text{Shift}}$	0	0	0	0
Value = 128	Value = 64	Value = 32	Value = 16	Value = 8	Value = 4	Value = 2	Value = 1

Bits 0-3: Always zero

Bit 4: Zero if **SHIFT** is pressed

Bit 5: Zero if **CTRL** is pressed

Bit 6: Always one.

Bit 7: Always one

The `kstat` function is useful in determining whether the user is pressing the **SHIFT** or the **CTRL** key while rotating the knob. Your program could then perform an alternate function for the knob's rotation.

The Key Buffer Empty Function

`key`

The `key` function is essentially unchanged from the 9825 `key` function, in that it returns unprocessed keycodes. The `key` function now returns hardware key codes, but the `asc` function can be used to obtain the ASCII codes for keys. These ASCII keycodes are identical for all HPL computers. For example, the hardware keycode for "5" on the 9825 numeric keypad is 83 while on the Model 236 the same key has a code of 2885. On both computers, however, the ASCII code is 53.

If your programs depend on the 9825 hardware keycodes, they will have to be converted to function correctly with the new Serial 200 keycodes. Refer to the Keycode Conversion Table for the corresponding values of 9825 and Series 200 keys, or you can use the keycode conversion utility function "9825 key" supplied with the HPL Utility Programs to achieve program compatibility.

Serial Interface Control (O&P p. 7-14)

The Remote Keyboard Statement

```
r k b d select code [ ,type]
```

The `r k b d` statement operates in an identical fashion to the 9825 `r k b d` statement except that the type 0 remote keyboard now requires Series 200 keycodes. Note that the type 1 remote keyboard (ASCII) operates the same for both machines.

For power-up remote keyboard operations, there is a jumper on the 98626A Interface that must be cut (as with the 98036A). Refer to the 98626A interface diagram in the 98626A Installation manual for the location of the remote keyboard jumper.

The Press Keyboard Statement

```
p k b d[string expression]
```

The press-keyboard statement effectively “pushes” keys on the keyboard, exactly as if the human operator were pressing those keys. System control keys are pressed by putting their ASCII value in the press-key string. For example, a decimal 10 character (line-feed) is the character for **EXECUTE**, and has the effect of executing previous characters in the string. The statement

```
p k b d "5*2^F"           results in a display of 10.00.
p k b d "5*2" & char(10) results in the same display.
```

The `EDIT A$` statement in BASIC can be simulated in HPL using `p k b d A$; ent A$`. The string in `A$` will be placed in the keyboard line. Any modifications that need to be made can be done using the editing keys. Then when all corrections have been made, press **CONTINUE** (**CONT**).

The `p k b d` statement has several potential uses. One use is to give the user “live keyboard” capability even though a running program has an “on key” service in effect. Now, instead of having to write a complete interpreter within the program to implement “live keyboard,” the program merely has to `p k b d` those ASCII keycodes that need to be executed by the system. For example, to emulate a full live keyboard with a running program:

<pre>0: on key "svc" 1: sto +0 2: "svc":if not (asc key>K);kret 3: if K#1;p k b d char(K);kret 4: end *3629</pre>	<pre>Set up service routine Loop Test for and exit if empty buffer Else "push" key and exit unless PAUSE key was pressed: stop.</pre>
--	--

Another example of `pkbd` is to access special function keys `k30` and `k31` after having defined them using the `sfk` statement. As an interesting exercise, check the default definitions of these keys.

Type: `pkbd "` Enter `pkbd` with opening quote.
 Press: `shift-STEP` Define the ASCII key code for `sfk 30`.
 Type: `158`
 Type: `"` Close the quotes.

Now press `EXECUTE` (`EXEC`)

The display shows the version of HPL:

```
(RAM) HPL 2.0 Ready
```

The Define SFK Statement

```
sfk[key number[ ,definition string[ ,label string]]]
```

The `sfk` statement allows special function keys to be defined from a program. Keys can be labelled with the optional label string, and defined to be any valid key sequence with the optional definition string.

```
sfk 9          erases the definition for SFK#9
sfk           erases all key definitions; replaces them with their de-
              fault settings
sfk 8, "wrt 705," defines key #8 to be the string "wrt 705,"
sfk 3, "HELLO JIM", "LOGON" defines key #3 to be labelled a log-on key sequence
                          for a terminal emulator
```

The length of the definition string plus the label string must not exceed 77 characters. Specifying a definition string only (with no label string) allows 80 characters of definition.

The ASCII value of special function keys on the keyboard range from 128 (`k0`) through 159 (`k31`), and can be defined by the program using `sfk` and "pressed" by the program using `pkbd`. The following example extends the live keyboard emulator to allow you to type control characters on the keyboard (`CTRL-A` for example) and not have them be executed as control keys. With the previous example, `CTRL-A` effectively executed a `PAUSE` (`PSE`). With this example, an ASCII "S_H" character is displayed, just as if there were no program running.

```
0: on key "svc"
1: goto 1
2: "svc":asc (key→A)→K
3: if bit(9,A)=0 and K<32:sfk 31,char(K):pkbd char(159):kret
4: pkbd char(K):kret
5: end
```

Systems Programming Instructions (O&P p. 7-21)

The System Boot Statement

```
sysboot ["file name"]
```

The `sysboot` statement allows you to “boot” or start up any ROM or soft operating system, including a re-booting of the current system. Omitting the file name parameter causes the current operating system to be reloaded and initialized.

Specifying a file name of one character followed by a null character (ANY CHR of 000) will cause the soft system file name "SYSTEM_x" to be booted from the internal minifloppy, if the system file is present. (The character “x” of the system file name is the same character specified in the `sysboot` statement.) If no disc is present with the "SYSTEM_" file on it, the ROM operating system “x” will be booted and initialized.

Example: `sysboot "B^U"` Boot BASIC ROM language system (if BASIC soft system is not present)

↙ (null)

`sysboot "SYSTEM_B"` Boot BASIC soft system



Matrix Programming

Array Output (Matrix p.8)

```
apr array variable[ ,array variable]...
```

The array print statement prints the elements of an array to the system printer. The rightmost subscript now increments most rapidly. This is different from array prints for the 9825, which incremented the leftmost subscript most rapidly. In addition, `apr` now utilizes the full width of the printer to print the array: the printed array is no longer formatted as if it were being printed to the 9825's strip printer.

Disc Programming

In general, disc operations in Series 200 HPL operate identically to 9825 disc operations. However, there are some optional extensions to the Disc Programming Syntax that allow you to access some additional disc drive types and some new capabilities (such as having different file pointers on different disc controllers, and accessing tape files on disc).

If you are familiar with disc programming, reading this section should give you enough information to utilize the new disc programming capabilities. If you should need more information about performance or implementation, refer to the Disc Programming Technical Appendix at the back of this manual.

If you are not familiar with disc programming, you should first read the 9825 Disc Programming Manual, then refer to this section for information regarding the extensions made to HPL 2.0 disc programming.

The Mass Storage Unit Specifier

A mass storage unit specifier, or msus, is a string expression giving a formal description of the mass storage device being accessed:

1. The device and format being used;
2. The device select code (and bus address, if applicable);
3. The device unit number.

The following table summarizes mass storage unit specifier parameters.

Device Format Table

Device Format Specifier	Disc Type	Disc Format	Select Code Range	HP-IB Address Range	Default Select Code Value	Unit Number Range
I	Internal	LIF ¹	–	–	–	0*
M	8290x	LIF ¹	1-15	0-7	700	0-3
M	9133V micro disc	LIF ¹	1-15	0-7	700	0
F	9885	9825 Compatible	1-15(not 7)	–	8	0-3
G	9885	LIF ¹	1-15(not 7)	–	8	0-3
H	9895	9825 Compatible	1-15	0-7	707	0-3
H	9133V Winchester	9825 Compatible	1-15	0-7	700	0-3
J	9895	LIF ¹	1-15	0-7	707	0-3
J	9133V Winchester	LIF ¹	1-15	0-7	700	0-3

* Unit numbers 0 and 1 on a Model 236.

¹ LIF: Hewlett-Packard Logical Interchange Format: File type ASCII is compatible with Series 200 BASIC language ASCII files, Pascal .ASC files, and HP 2642A Terminal files.

The format for a mass storage unit specifier (msus) is:

```
: [device format [select code]] [ ,unit number]
```

Normally, only the device format need be given. Each device format specifier has a default select code and unit number. However, certain critical statements require you to specify all the msus parameters due to the potentially destructive effect of those statements (`init`, `killall` and `disc copy`).

The select code parameter must consist of 1 to 4 digits only. The unit number parameter must consist of a single digit only. Plus or minus signs, periods, or “e” are not allowed. This means that if functions such as the `str` function are used to generate the desired string expression, `fxd0` or equivalent must be in effect.

Note that when specifying the Model 226/236 internal drive as the mass storage device, there is no associated select code, so that parameter is omitted. Some examples help clarify the msus concept:

```
:FB,0      Specifies 9885 disc, 9825-compatible format, select code 8, unit number 0.
:I         Specifies Model 226/236 internal disc drive.
:J707,2    Specifies 9895 disc, LIF format, select code 707, unit number 2.
:H,3       Specifies 9895 disc, 9825-compatible format, default select code for “H”,
           unit number 3.
:,2        Specifies current device format and select code, unit number 2.
```

The msus is allowed as an extension to a file name parameter as shown in the following examples:

```
assign "Data1:H707,3",1
assign "Data2:I",2
kill "Oldprog:FB,1"
```

All disc programming statements except the `files` statement that use the “file name” parameter can now use the “msus” parameter appended to the file name.

The Mass Storage Is Statement

There is a new statement, `msi`, added to the Disc Programming Syntax to allow you to specify the current mass storage device (as was accomplished with the `drive` statement on the 9825) with a mass storage unit specifier (`msus`). This allows access not only to 9825-compatible discs, but also to LIF discs. The syntax is:

```
msi[msus]
```

where `msus` can be any suitable string expression that conforms to the format already defined. Executing `msi` with a specified `msus` will set the value of the current default `msus` (current drive and format). That value will remain in effect until it is explicitly changed with another `msi` or `drive` statement, or until a power-up, reset, or erase occurs.

Executing `msi` with no parameter will restore the power-up default `msus` of `" : I "` for the internal minifloppy.

Some examples of the `msi` statement follow:

<code>msi</code>	Restore power-up default (" : I " for Model 226/ 236 internal drive)
<code>fxd0;" : J " → M\$; 707 → S; 2 → U</code> <code>msi M\$&str(S)&" , "&str(U)</code>	Set mass storage device to 9895 disc select code 707, unit number two, LIF format.

The Assign Statement (Disc Programming p. 3-5)

```
assign file name , file number [ , unit number [ , return variable ] ]
```

The assign statement has been extended to allow you to include the `msus` with the file name in order to completely define the characteristics of the file and drive. When the `msus` extension is being used and the return variable is given in order to obtain file status, you can specify a "unit number" of -1 which prevents overriding the `msus` unit number. This becomes, in effect, a "place holder" in the parameter list. An example illustrates this:

```
assign "Datafile:H701,3" , 6 , -1 , X
```

This assigns file number 6 to file "Datafile" on a 9895 disc, 9825-compatible format, controller address 701, unit number 3. The -1 unit number is a place holder, and X is the return variable for the file status.

```
assign "Datafile:H701,3" , 6 , 0 , X
```

This statement has all the same effects as the one above, except that now unit number 0 is selected, overriding the `msus` unit number of 3.

The Drive Statement (Disc Programming p.1-14)

The drive statement is essentially unchanged from its 9825 definition except that file pointers are no longer cleared if the select code is specified. However, the select code range for a 9885 disc is now 1-6 and 8-15. (Select code 7 is reserved for the internal HP-IB.) Note that you cannot direct disc operations to an LIF format disc by using the drive statement. The drive statement **can** be used to change just the current unit number leaving other specifications unchanged. This can be done regardless of the current device format in effect, even if it is an LIF disc.

The Initialize Statement (Disc Programming p.4-3)

```
init unit number , select code [ ,interleave factor]
or
init complete msus [ ,interleave factor[ ,number of directory records]
```

The initialize statement can be used as a strictly 9825-compatible statement as shown in the first syntax. Only 9885 and 9895 discs of 9825-compatible formats (F and H formats) can be initialized when the first syntax version of `init` is used.

The second syntax shown for the `init` statement allows the complete range of HPL supported discs and formats to be specified, as per the “msus” specifier.

With either version of the `init` statement, the complete disc drive msus specifications **MUST** be given. This information is always printed on the first line of a catalog listing of a disc for future reference.

Allowable interleave factors are 1 thru 15 for 5¼-inch discs, and 1 thru 29 for 8-inch discs. The default interleave factor for a given disc drive yields the best performance possible under a wide range of conditions. However, in certain cases, you can increase the performance of the disc system by specifying a different interleave factor (but if you go below the recommended minimum, extremely poor performance may result!). Some of the cases where better performance may be obtained by specifying a smaller interleave factor are:

- Using a DMA card with a 9895 disc drive to obtain better HP-IB transfer rates.
- Using a disc primarily for backup purposes, and autoverification is disabled (voff).
- Using a disc primarily for tape operations, and autoverification is disabled (avd).
- NOT using a disc for large numeric array or large string transfers to TDATA or ASCII file types.
- NOT using a disc for large numbers of random accesses to consecutive records of a TDATA type file.

Disc Interleave Factors

Disc Drive	Default Interleave Factor	Overall Optimum Interleave Factor	Suggested Minimum Interleave Factor ^{1,2}
Internal	2	2	1
8290x	5	5	3 ³
9885	2	2	1
9895 no DMA	4	4	3 ³
9895 with DMA	4	3	2 ³

You can now specify on a LIF disc the number of records used to hold the directory. Each directory record can hold up to 8 directory entries. The default number of directory records for 5¼-inch discs is 14, or 112 entries. The default for 8-inch discs is 28 records, or 224 entries.

Those records not used for directory entries are available for programs and data. Depending upon your application, you can trade off directory records for data records or vice-versa. If you have a great number of small data files, for instance, you may need more than the default number of directory records to hold a greater number of directory entries. You can specify from 1 to 500 records for the directory depending upon your needs. Normally, the default number of directory records will suffice.

The following table suggests some practical guidelines for selecting a maximum number of directory records per disc type to allow one directory entry for each sector (256 bytes). (This configuration would be necessary if a data disc needed a maximum number of files of 256 bytes or less.)

Maximum Practical Number of Directory Records for LIF Discs

Media	# of directory records	# of directory entries	# of user records
Minifloppy	118	944	936
9885	210	1680	1678
9895 sgl sided	244	1952	1944
9895 dbl sided	500	4000	3998

Both the default interleave factor and default number of directory records can be selected by specifying a parameter of -1 in the parameter position. For example,

```
init ":J707,0",-1,-1
```

- 1** See considerations listed above. Also, if an internal disc is used primarily as a system disc, the suggested minimum interleave factor may be used.
- 2** The suggested minimum interleave factor yields the optimum performance obtainable for a given disc, under optimum conditions. This must usually be arrived at experimentally, as different programs and access requirements may need larger interleave factors to attain maximum performance.
- 3** This interleave is efficient only for backup without auto verification. All other operations require an interleave factor at least one higher than recommended.

The Disc Type Function (Disc Programming p. 1-15)

`dtype`

The `dtype` function now returns a value of 9 for either the internal minifloppy or external 8290x minifloppy drives. However, the open drive door status returned by `dtype` is different for the two drives.

Internal drive: status check indicates “disc changed” but not “door open” or “disc present”. A value of 1 is never returned, and values of 2 and 9 do not guarantee that the door is now closed and disc is present.

8290x drive: status check does not indicate a condition of “door opened”, so a value of 2 is never returned.



The Catalog Statement

HPL 2.0/2.1 supports 3 new file types on 9825-compatible discs. All three are related to 9825 tape statement emulation. The same file types are also supported on LIF discs. The file type mnemonics used in 9825-format disc catalog listings for these new files are:

Z (NULL) null file - 9825 tape file type 0
 N (NBDATA) numeric binary data - 9825 tape file type 2
 S (SBDATA) string/mixed binary data - 9825 tape file type 3

The catalog listing format for 9825-compatible discs is identical to that of the 9825/98228 ROM combination. Old 9825 programs which do cat's to buffers to access the catalog information should have no problem doing the same with 9825-compatible discs on Series 200 HPL.

The catalog listing format for LIF discs, however, is different from the 9825-compatible disc catalog listing due to the fact that the 9825-compatible disc catalog listing only has space for 6-character file names, whereas LIF file names can be up to 10 characters in length. Thus, the 9825-compatible and LIF catalog listing formats are somewhat different. An example of a LIF catalog listing from the Model 226 internal disc is shown below:

```
MSI ":I,0"
AVAILABLE RECORDS 1002
FILE NAME    SCD TYPE                    #BYTES #RCRDS    ADDRESS
=====
grades            TDATA                                    10        16
Class            PROGRM                                  96        26
Count            PROGRM                                  62        27
Name             PROGRM                                 242       28
give             TDATA                                    10        29
Keys             KEYS                                    200       39
null_file        NULL                                     0        50
secured_Pr *     PROGRM                                 242       40
ascii_file       ASCII                                     1        51
numeric_bd       NBDATA                                  24        52
strings_bd       SBDATA                                 134       53
```

Notes:

1. The "complete msus" is given on the msi line. Users may find this useful for reference, since the new alternate syntaxes for init, killall, and disc copy require the "complete msus".
2. The "SCD" field, following the file name, is used to indicate secured programs. If a "*" is present in this field, the program is secured.
3. The file address is given as a single logical record address, as opposed to breaking it up into separate logical track and sector addresses.

The Killall Statement (Disc p. 1-18)

```
killall unit number , select code
or
killall complete msus
```

The `killall` statement can also be used with the full range of HPL 2.1 supported discs and formats by specifying the second syntax, which allows the `msus` specifier.

The Open Statement (Disc p. 3-2)

The `open` statement now allows you to specify the desired type of file when opening a data file. The new syntax is shown below.

```
open file name , number of records [ ,file type]
```

where `file type` is a string expression having the value "ASCII", "NULL", or "TDATA".

The `open` statement can be used to create two new file types besides typed-data (TDATA) files. ("TDATA" corresponds to the old 9825 typed-data files, designated by a "D" in the catalog listing.) The two new file types supported by HPL 2.0 are "ASCII" and "NULL". The default file type is "TDATA". As with "TDATA" files, when an "ASCII" file is created, each record of the file is initialized with logical end-of-file marks. The records of "NULL" files, however, require no initialization; thus, none is performed.

"ASCII" files provide compatibility between HPL and other Series 200 programming languages. "ASCII" data files are also used to provide transportability to and from other computers or terminals with LIF ASCII file capability. Their use is identical to that of regular data files ("TDATA"), with some exceptions. Technical considerations in the use of ASCII files are discussed in the Disc Programming Technical Appendix at the back of this manual.

"NULL" files are essentially unused "tape" emulation data files. Their use from a programming standpoint is like accessing data files on the 9825 tape cartridge. "NULL" files become either "NBDATA" (Numeric Binary Data) files or "SBDATA" (String or Mixed Binary Data Files) files when data is stored to the file. These files can be accessed either with tape cartridge operations or disc programming operations. Their use with tape cartridge operations (`rcf`, `ldf`) is discussed under the Tape Cartridge Operations section of this manual and in the Disc Programming Technical Appendix.

The Disc Copy Statement (Disc p. 4-7)

```
copy [source drive number [ , select code] ,]"to"  
[ , destination drive number [ , select code]]
```

The original syntax of the (disc) copy statement with select codes works exactly as it did on the 9825; it allows access to 9825-compatible 9885 and 9895 drives. However, if the source or destination select code is omitted, the current default device format and select code are used in each case, whatever they may be.

The original syntax of the (disc) copy statement without select codes in HPL 2.0 works like the new drive statement: the unit number, if given, modifies only the current default drive number, and uses the current default device format and select code, whatever they may be.

Examples:

```
copy 0,8,"to",1,707
```

Copy source disc, 9825-compatible 9885, select code 8, unit number 0, to destination disc, 9825-compatible 9895, select code 707, unit number 1.

```
msi ":G";copy "to",1
```

Copy source disc, LIF-type 9885, select code 8, unit number 0, to destination disc, LIF-type 9885, select code 8, unit number 1.

Important

Even though the syntax doesn't require it, it is highly recommended that the complete information be explicitly given, for the user's own protection, due to the "serious" nature of a disc copy.

NOTE

With the 9825/98228 ROM, you were not prevented from doing disc copy from a single-sided disc to a double-sided disc, in which case the double-sided disc would end up with the same amount of user space as the single-sided disc! Due to this unfortunate possibility as well as the even worse possibility of copying a minifloppy to a double-sided full-sized floppy (resulting in a 75% loss in capacity!), a test is performed by the operating system to prohibit disc copies between "significantly" different-sized media. If the destination disc is 50% larger than the source disc, the copy will not be allowed.

New Alternate Syntax for Disc Copy

`COPY` complete source msus , "to" , complete destination msus

With the new alternate syntax, all supported device/format specifiers can be accessed. Defaults are not allowed in the msus; select code and unit number must be specified EXPLICITLY.

Examples:

```
COPY ":M700,0","to",":I,0"
COPY ":FB,0","to",":H707,0"
```

Regardless of the syntax used, disc copies are not allowed from a LIF disc to a 9825-compatible disc or vice-versa.

The File Copy Statement (Disc p.4-7)

`COPY` source file name [, drive number [, select code]] ,
destination file name [, drive number [, select code]]

In general, almost all HPL files can be copied back and forth between LIF and 9825-compatible discs without any problems. There are, however, some complications associated with differences between LIF and 9825-compatible discs, which might prevent a file copy from taking place.

1. Invalid file name (error D3): LIF allows up to 10 characters in a file name, while 9825-compatible discs only allow up to 6.
2. Wrong file type (error D6): Some file types supported with LIF are not supported on 9825-compatible discs. ASCII files are the only HPL-created files with this characteristic.
3. Directory entry field overflow (error f3): Certain fields in a 9825-type directory entry are smaller than their counterparts in a LIF directory entry. This makes a directory entry field overflow possible when copying the file from a LIF disc to a 9825-compatible disc. The field most likely not to fit is the file size field. 9825-compatible discs cannot support files whose size in bytes is greater than 65536. Note that this does not apply to "TDATA" files, where the file size field is not used. Also, one field in a LIF directory is smaller than its counterpart in a 9825-type directory. This field is guaranteed not to overflow when copying an HPL-created 9825-compatible disc to a LIF disc, but with 9825-compatible files created by other mainframes, it might.

As with all other HPL statements having the “file name” parameter, file copy’s “file name” parameter has been extended to allow appending an optional msus. However, since the file copy syntax supports an optional unit number and select code in addition to the “file name” parameter, there may be confusion as to which select code and unit number will take precedence. The following rules apply:

1. If the “file name” does not contain an msus, the current default device format, select code, and unit number will be used, unless explicitly overridden by the optional unit number and select code. If the optional select code parameter is given, 9825-compatible 9885/9895 format is implied, the same as for the drive statement.
2. If the “file name” parameter contains an msus, the optional select code parameter (outside the msus) is not allowed.
3. If the “file name” contains an msus, and the optional unit number parameter (outside the msus) is given, it will override the unit number specified or implied by the msus.
4. The source and destination file names with their optional unit numbers and select codes are completely independent, that is, one can contain an msus while the other has the optional select code.

Examples:

```
copy "file1","file1b"
copy "file",0,"file",1
copy "file",0,B,"file:I"
```

The Save Binary Statement

```
saveb file name
```

The save binary statement stores the current binary in memory to the specified disc file. 9825 binaries and Series 200 binaries are not compatible. Binaries written for the 9825 are processor-dependent and as such would have to be completely re-written to run on a Series 200 computer. Because the file types for 9825 and Series 200 binaries have the same name, a user cannot merely look at a catalog listing and determine whether an unknown binary is for a 9825 or for a Model 236; both show up as type “B” on 9825-compatible discs and “BINARY” on LIF discs. However, HPL sets a flag in the directory to enable it to distinguish between binaries, so a Model 236 will not attempt to load a 9825 binary. To do so would have completely unpredictable results. The 9825, on the other hand, does not look at the flag, so it will attempt to load a Model 236 binary if directed to do so by the user. This will certainly generate unpredictable operation of the 9825. Don’t do it!

The Disc Read and Disc Write Statements

Important

Use of the dwrt statement can destroy the data on a disc, including the catalog. Use with caution!

```

d r e d starting logical record # , destination string
d w r t starting logical record # , source string

```

The `d r e d` and `d w r t` statements allow reading and writing to logical records on a disc. This is an extremely dangerous operation since one accidental write to the directory can destroy the disc. The disc must have been previously initialized. All types of strings, string arrays, and substrings are allowed with the following rules:

- The string (if a substring) must start on an even address.
- The string must specify an even number of bytes.
- If doing a `d r e d` to a string array, the entire array will be filled.
- If doing a `d w r t`, and the source string is a string array, every element of the array must be full.
- If you write a number of bytes which is not an even multiple of the record size (256 bytes), the unused portion of the last record will contain garbage.
- `d r e d` and `d w r t` will work on all supported mass storage devices (internal, 9895, 9885, 8290X, etc.).

Unimplemented Disc Programming Statements

Four disc statements have not been implemented in Series 200 HPL. They can be syntaxed, stored, listed, recorded with programs, and loaded with programs. However, any attempt to execute one of them will result in error f9 (Statement not implemented). The unimplemented statements are:

```

d u m p
l o a d
s a v e m
s e t m

```

Unimplemented Binary Statements

HPL 2.1, like the 9825's 98228 ROM, does not implement the statements contained in the binaries on the 98217 ROM's Disc System Cartridge (`init` and `killall` are implemented with a different syntax). These statements are:

```

d t r k
t i n i t
l t r k
d i r c
b o o t
v f y b
P t r n t s t
c k r d
i n i t (implemented with different syntax)
k i l l a l l (implemented with different syntax)

```

More I/O

Interface Control Operations

In general, Series 200 HPL register and interface programming operations operate identically to their 9825 counterparts. This is due to an emulation of the 9825 interface registers. You should be aware, however, that Series 200 interface hardware is indeed different from that of the 9825, and that this prevents a 100% accurate emulation. The following tables list the primary differences between 9825 interface operations and the Series 200 HPL emulation of those operations.

Select Code Summary Table

Select Code	9825 Source/Destination	Series 200 Source/Dest.
0 READ WRITE	Keyboard (KDP) Single line display	Keyboard Display line of CRT
1	Reserved for tape cartridge	External
2	External	.
.	.	.
.	.	.
7	.	Internal HP-IB
8	.	External
.	.	.
.	.	.
16 READ WRITE	Keyboard (KDP) Internal printer	Read from CRT Write to CRT

Note:

- Select code 1 on Series 200 is available for external devices.
- Select code 7 on Series 200 is reserved for internal HP-IB.
- Read from select code 16 does not return keycodes on the Series 200, but instead reads data from the CRT print area.
- Read from select code 0 on Series 200 returns Series 200 hardware keycodes, completely unlike 9825 hardware keycodes.

98622A GPIO Operations

The 98622A GPIO Interface operates identically to its 9825 counterpart, the 98032A, except that now when the invert-data jumper is installed, all data is inverted regardless of the transfer type selected. (Recall that the 9825 and 98032A did not invert data for DMA or fast read-write transfers, or for register I/O.) Also, a software RESET of the interface now will terminate any active transfer to or from the interface.



Series 200 Select Codes 0 and 16

Operation	Select code 0	Select code 16
<code>rdb</code>	Wait for keypress, then return keycode	Read a byte from CRT at current print position
<code>red</code>	Not allowed	Formatted read from CRT at current print position
<code>wtb</code>	Write bytes to display line	Write bytes to CRT at current print position
<code>wrt</code>	Formatted write to display line	Formatted write to CRT at current print position
<code>rds</code>	Always returns value of 8	Always returns value of 8
<code>wtc</code>	Select language jumper (wtc 0,5 selects Katakana)	Select language jumper (wtc 16,5 selects Katakana)
<code>wti4</code>	No operation	No operation
<code>wti5</code>	No operation	No operation
<code>wti6</code>	No operation	No operation
<code>wti7</code>	Select language jumper	Select language jumper
<code>rdi(4)</code>	Always returns value of 0	Always returns value of 0
<code>rdi(5)</code>	Always returns value of 8	Always returns value of 8
<code>rdi(6)</code>	Always returns value of 0	Always returns value of 0
<code>rdi(7)</code>	Always returns value of 0	Always returns value of 0

NOTE:

- wti5 on the 9825 could control features such as insert/replace cursor type, run light on/off, trigger beeper, trigger printer, and trigger display. These are not accessible on Series 200.
- Read and write operations have different effects on Series 200 select codes 0 and 16. On the 9825 `rdb(0) = rdb(16)`.
- `rdi(4)` on the 9825 reads the hardware keyboard scanner. There is no equivalent on the Series 200.
- `wti4` and `wti6` on the 9825 wrote bytes to the display and printer buffers. There is no equivalent operation on the Series 200.

98623A BCD I/O Operations

For BCD input, operation of the 98623A BCD Interface is identical to its 9825 counterpart, the 98033A. However, there is also an 8-bit output latch available to the user. This is to be considered simply an output latch, with no dedicated handshake lines. Access to the latch is via binary operations. For example to latch a byte "N" to a BCD interface on select code 3:

```
wti0,3; wti4,N
```

or simply

```
wtb 3,N
```

Executing a write-control (`wtc`) to a BCD interface with the RESET bit set (bit 5=1) will reset the interface and terminate any active `tfr` for the interface.

HP-IB Operations

There are some slight differences between register operations with the internal 98624A HP-IB interface and its 9825 counterpart, the 98034A. Please note that all high-level operations (wrt, red, tfr, pct, etc.) are compatible with 9825 operations! The following table serves to summarize HP-IB operations for the 9825 and Series 200.

HP-IB Operations

Operation	9825 Result	Series 200 Result																																																																
<p>rds(S,A,B,C) → D</p> <p>A</p> <p>B</p> <p>C</p> <p>D</p>	<p>HP-IB Extended Status</p> <table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>DCL</td><td>0</td><td>Error</td> </tr> </table> <p>1 1 0 - HPIB Address -</p> <table border="1"> <tr> <td>EOI</td><td>REN</td><td>SRQ</td><td>ATN</td><td>IFC</td><td>ND</td><td>NR</td><td>DAV</td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td>AC</td><td>FD</td><td></td> </tr> </table> <p>SRQ CA TA LA SC 1 SPL EOI</p>	0	0	0	0	0	DCL	0	Error	EOI	REN	SRQ	ATN	IFC	ND	NR	DAV						AC	FD		<p>HP-IB Extended Status</p> <table border="1"> <tr> <td>0</td><td>0</td><td>REM</td><td>LLO</td><td>GET</td><td>DCL</td><td>IFC</td><td>Error</td> </tr> <tr> <td></td><td></td><td>LOC</td><td></td><td></td><td></td><td></td><td></td> </tr> </table> <p>1 1 0 - HPIB Address -</p> <table border="1"> <tr> <td>EOI</td><td>REN</td><td>SRQ</td><td>ATN</td><td>IFC</td><td>ND</td><td>NR</td><td>DAV</td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td>AC</td><td>FD</td><td></td> </tr> </table> <p>- REN, IFC may be incorrect¹ if System Controller - ATN indicates attempted drive status, not actual state of line if Active Controller - SRQ may be incorrect¹ if Active Controller - EOI may be incorrect¹ if Active Talker - NDAC, NRFD may be incorrect¹ if not Active Talker or Active Listener</p> <table border="1"> <tr> <td>SRQ</td><td>CA</td><td>TA</td><td>LA</td><td>SC</td><td>1</td><td>0</td><td>EOI¹</td> </tr> </table> <p>- SRQ reflects current state of SRQ line: this is not latched as it was on the 98034 Interface - The 98034 SPL bit is now always false on the 98624A</p>	0	0	REM	LLO	GET	DCL	IFC	Error			LOC						EOI	REN	SRQ	ATN	IFC	ND	NR	DAV						AC	FD		SRQ	CA	TA	LA	SC	1	0	EOI ¹
0	0	0	0	0	DCL	0	Error																																																											
EOI	REN	SRQ	ATN	IFC	ND	NR	DAV																																																											
					AC	FD																																																												
0	0	REM	LLO	GET	DCL	IFC	Error																																																											
		LOC																																																																
EOI	REN	SRQ	ATN	IFC	ND	NR	DAV																																																											
					AC	FD																																																												
SRQ	CA	TA	LA	SC	1	0	EOI ¹																																																											
<p>rds(s)</p> <p>wtcS,X</p>	<p>Return fourth HP-IB status byte (D, above)</p> <p>Not Allowed.</p>	<p>Return fourth HP-IB status byte (D, above)</p> <p>If X ≤ 31, then software reset and terminate transfer. If X < 31 then HP-IB address is set to X. If X = 31 then no further action. If X > 31 then configure parallel poll response: Bit 4 = 1: disable ppoll response Bit 4 = 0: enable ppoll response Bits 0 - 2: define ppoll response line 0 thru 7 Bit 3 = 0: "0" true ppoll response Bit 3 = 1: "1" true ppoll response</p>																																																																
<p>rdi4</p> <p>rdi5</p> <p>rdi6</p> <p>rdi7</p> <p>wti4 or 6</p> <p>wti5</p> <p>wti7,X</p>	<p>Immediate read of R4</p> <p>Immediate read of R5</p> <p>Immediate read of R6</p> <p>Immediate read of R7</p> <p>Immediate write to R4/R6</p> <p>Immediate write to R5 (Not recommended)</p> <p>Immediate write to R7</p>	<p>Always 0</p> <p>Always 0</p> <p>Always 0</p> <p>Always 0</p> <p>Wait for FLG then write to R4/R6</p> <p>Same as EIR</p> <p>If bit 7 of X = 0 then bits 0-7 define serial poll response byte. If bit 7 of X = 1 then Bit 2 (ATN): 1 = TRUE Bit 4 (EOI): 1 = enable EOI with next data byte</p>																																																																
<p>ios</p> <p>iof</p>	<p>Return 98034 STS line.</p> <p>Return 98034 FLG line</p>	<p>Identical operation to 9825.</p> <p>If the 9826 HP-IB is addressing or talking to the bus then iof = 1 when ready for next data item, else iof = 0.</p>																																																																

¹ These lines are driven from Series 200 in the specified state. However, the actual status of these lines may be different from that indicated by simply reading status. (Consider the case of an HP-IB analyzer or a defective instrument forcing a line to a state different from the attempted drive state of the 98624A HP-IB interface card.)

HP-IB Operations (Cont'd)

Operation	9825 Result	Series 200 Result														
<p><code>e i r</code></p> <p><code>t f r</code> (buffer type 5)</p> <p><code>t f r 701,1,"buf"</code></p> <p><code>t f r 701,2,"buf"</code></p> <p><code>t f r "buf",701,1</code></p> <p><code>t f r "buf",701,2</code></p> <p>Output to non-existent device</p>	<table border="1"> <tr> <td>SRQ</td> <td>CA</td> <td>TA</td> <td>LA</td> <td>Internal Use Only</td> <td>Err DCL SDC</td> <td>EOI</td> </tr> </table> <p>Not allowed</p> <p>Format ignored</p> <p>Format ignored</p> <p>Format ignored</p> <p>Format ignored</p> <p>Completes.</p>	SRQ	CA	TA	LA	Internal Use Only	Err DCL SDC	EOI	<table border="1"> <tr> <td>SRQ</td> <td>CA</td> <td>TA</td> <td>LA</td> <td>Internal Use Only</td> <td>IFC GET DCL SDC</td> <td>EOI</td> </tr> </table> <p>NOTE: CA, TA, LA interrupts do NOT wait for ATN to go false before triggering! (The 98034A triggered these interrupts when ATN went false.) The user service routine should wait for ATN false before accessing the bus.</p> <p>Byte DMA transfer if DMA card is present.</p> <p>Terminate on EOI.</p> <p>Do not terminate on EOI.</p> <p>Send EOI with last byte.</p> <p>Do not send EOI.</p> <p>Hangs. Card reset is required (w/c7,31 or SHIFT PAUSE)</p>	SRQ	CA	TA	LA	Internal Use Only	IFC GET DCL SDC	EOI
SRQ	CA	TA	LA	Internal Use Only	Err DCL SDC	EOI										
SRQ	CA	TA	LA	Internal Use Only	IFC GET DCL SDC	EOI										

HP-IB Capability Comparison

98034	Series 200 HPL
SH1 (Complete)	same
AH1 (Complete)	same
T6 (No Talk-only)	same
TE0 (No Extended Talker)	same
L4 (No Listen-only)	same
LE0 (No Extended Listener)	same
SR1 (Complete)	same
RL0 (none)	RL1 (Complete)
PP2 (Locally configurable only)	PP1 (Complete)
DC1 (Complete)	same
DT0 (none)	DT1 (Complete)
C1,2,3,4,5	same
E1 (Open collector driver)	E2 (Three-state driver)

98626A Serial I/O Operations

The 98626A RS-232C Interface is nearly identical in operation to the 98036A RS-232 Interface with the following exceptions:

- There is a new register, R3OUT, that must be set before successful emulation of the 98036A Interface is possible. If a female RS-232C cable (DCE connector) is present, bit 0 of R3 must be set (= 1). For male cable (DTE connector), R3 bit 0 must be clear (=0). To enable CTS/DTR line transmitter control (this emulates the 98036A: any device connected to CTS/ DTR could control the USART transmitter with this line), R3 bit 1 must be clear (=0). To disable transmitter control, set R3 bit 1 (=1).

R3 OUT

Most Significant Bit						Least Significant Bit	
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
–	–	–	–	–	–	Handshake	Cable Type
Value = 128	Value = 64	Value = 32	Value = 16	Value = 8	Value = 4	Value = 2	Value = 1

Bit 0: 1 = Female connector (DCE: Std cable)

0 = Male connector (DTE: Opt. 001 cable)

Bit 1: 0 = Enable transmitter handshake (CTS/DTR)

1 = Disable transmitter handshake

USART Mode Word (R4C)

Most Significant Bit					Least Significant Bit		
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Number of Stop Bits 00 = not valid 01 = 1 bit 10 = 1.5 bits 11 = 2 bits		Parity Type 0 = Odd 1 = Even	Parity Enable 0 = Disable 1 = Enable	Character Length 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits		98036A Bit Rate Factor not used for 98626A	

Bits 0 and 1 (Bit Rate Factor) are ignored.

USART Control Word (R4D)

Most Significant Bit				Least Significant Bit			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Always 0	USART Reset	No Connect Request To Send Pin 4 (Option 001)	Reset Status Bits of USART Status Word	Send Bread Character	Enable Data Receiver	Data Set Ready Pin 6(Std.) Data Terminal Ready Pin 20 (Option 001)	Enable Data Transmitter

Same as 98036A, except that bit 5 cannot be used to control the Clear-to-Send line on the standard (DCE) cable. Instead, Clear-to-Send is controlled by Request-to-Send of the DTE device.

USART Status Word (R4E)

Most Significant Bit				Least Significant Bit			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RTS-4 (Std.) DSR-6 (Option 001)	Always 0	Framing Error	Overrun Error	Parity Error	Transmitter Empty	Receiver Ready	Transmitter Ready

Same as 98036A.

R6 Registers**R6 OUT (Opt. 001)**

Most Significant Bit				Least Significant Bit			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	Half/Full Speed	No Connect	No Connect	DSRS pin 23	SRTS pin 19
Value = 128	Value = 64	Value = 32	Value = 16	Value = 8	Value = 4	Value = 2	Value = 1

- R6OUT, Opt. 001 (DTE) cable has two no-connects. There is no U.K. Data Signal Rate Select (pin 11): bit 2. There is no Special Purpose line (pin 25): bit 3.

R6 IN (Opt. 001)

Most Significant Bit				Least Significant Bit			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	1	1	1	1	Secondary Carrier Detect pin 12	Ring Indicator pin 22	Data Carrier Detect pin 6
Value = 128	Value = 64	Value = 32	Value = 16	Value = 8	Value = 4	Value = 2	Value = 1

- R6IN, Opt. 001 (DTE) cable is unchanged.

R6 OUT (Standard)

Most Significant Bit				Least Significant Bit			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	Half/Full Speed Control	Ring Indicator pin 22	No Connect	Secondary Carrier Detect pin 12	Data Carrier Detect pin 8
Value = 128	Value = 64	Value = 32	Value = 16	Value = 8	Value = 4	Value = 2	Value = 1

- R6OUT, Standard (DCE) cable has one no-connect. There is no Secondary Carrier Detect (pin 12): bit 2. Note that Ring Indicator and Data Set Ready (pins 22 and 6) are tied together. The last one set will determine the actual line state of both.

R6 IN (Standard)

Most Significant Bit				Least Significant Bit			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	1	1	1	1	0	No Connect	Secondary Request To Send pin 19
Value = 128	Value = 64	Value = 32	Value = 16	Value = 8	Value = 4	Value = 2	Value = 1

- R6IN, Standard (DCE) cable has one no-connect. There is no Data Signal Rate Select (pin 23): bit 1.

Internal Clock Access

Series 200 computers have built-in real-time clocks that are accessible from the HPL for the purposes of keeping time, reading time, and executing program routines at a set time, periodically, or after a programmed delay. The following statements provide the capability to set and read the real time. There are several new interrupt control statements provided to enable interrupts from the real-time clock. Refer to the following section for the syntaxes of these statements.

The Set Clock-Time Statement (I/O p. 5-15)

```
stime [seconds]
```

This statement sets the internal clock to the specified time (in seconds). The resolution of the internal clock is .01 seconds, and the range is from zero seconds to 180 years. As an example, the following routine sets the internal clock to a value relative to March 1, 1900, taking into account leap-years. This format is compatible with BASIC and Pascal language systems.

```
0: "Timeexp1":
1: dim T#[80],XE[12],M#[12,3]
2: aclr
3: qsb "Set-time"
4: qsb "Read-time"
5: dsp T#;gto 4
:
24: "Set-time":
25: prt "Type in 24-hour time and date as:"
26: prt "Hr (H),Min(N),Sec(S)"
27: prt "Month(M),Day(D),Year(Y)"
28: ent H,N,S,M,D,Y;if Y>1899;Y-1900}Y
29: if M>2;M-3}M;gto 31
30: M+9}M;Y-1}Y
31: stime (int(1461*Y/4)+int((153*M+2)/5)*D)*86400+H*3600+N*60+S;ret
```

The `stime` statement should be executed before any on-match, on-cycle, or on-delay interrupts are enabled.

Executing `stime` with no parameters transfers the battery clock time into the internal computer clock. (The battery clock is slightly more accurate than the internal computer clock.)

The Read Clock-Time Function (I/O p. 5-15)

```
rtime
```

This function returns the current number of seconds held by the internal clock. The resolution is .01 seconds with a range of zero seconds to 180 years. The following routine displays the clock time assumed relative to year 1900:

```
:
36: "Read-time":
37: data JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
38: rstr "Read-time";for I=1 to 12;read M#[I];next I
39: (int((int(rtime)+86400}S)/86400}Z}D)*4-1}K
40: int((K-153*(int((5*int((K-1461*(int(K/1461)}Y)+4)/4)-3}K)/153}N)+5)/5}D
41: if M<10;M+3}M;gto 43
42: M-9}M;Y+1}Y
43: S-2*86400}S
44: S-(int(S/3600}H)3600}S
45: S-(int(S/60}N)*60}S;fxd 0
46: str(H)&": "&str(N)&": "&str(S)&" "&M#[M]&str(D)&","&str(Y+1896}Y}T#
47: ret
```

Interrupt Control (I/O p.5-5)

The On Interrupt Statement

```
on i select code [, label [, abort byte]]
```

The syntax of the `on i` statement has been extended to allow the service routine linkage for a select code to be cancelled. Executing `oni` with no service routine label cancels the interrupt service routine linkage for that select code.

Interrupt Lockouts (I/O p. 5-15)

Like the 9825, Series 200 HPL has certain critical operations that lock out interrupts for short periods. Unlike the 9825, however, Series 200 HPL does not lock out interrupts for long periods of time during tape drive accesses. (There is no tape drive.) Series 200 HPL is also capable of supporting two DMA operations simultaneously; therefore, utilizing one DMA channel does not prevent initiation of a second, concurrent DMA transfer.

The On Clock-Cycle Statement (I/O p. 5-15)

```
on cycle [period , label]
.
.
cret
```

This statement first zeroes the internal clock cycle count, then enables a periodic (cyclic) interrupt from the internal clock. The minimum period allowable is 10 milliseconds (.01 second) and the maximum period allowed is one day minus 10 milliseconds ($24*60*60 - .01$). The "label" parameter specifies the line label of the on-cycle service routine. The on-cycle service routine will be executed every "period" number of seconds as a result of an end-of-line branch, and must eventually terminate with a `cret` statement. The on-cycle service routine must access the current clock-cycle count via the `cycle` function described next. The on-cycle routine cannot be exited (via `cret`) until after the cycle function has been executed! Executing `on cycle` with no parameters disables on-cycle interrupt service.

The Cycle Function

The `cycle` function returns the number of clock-cycles that have occurred since the last-executed on cycle statement or cycle function. (The on cycle statement and the cycle function zero the clock-cycle count.) The cycle function can be executed by the user program at any time to read the current cycle count, but it (cycle) **MUST** be executed within an on-cycle service routine before `cret` will allow a return to the main program.

The On Clock-Delay Statement (I/O p. 5-15)

```

on delay [wait , label]
.
.
dret

```



This statement enables a “wait” clock interrupt from the internal clock. Once `on delay` is executed, the internal clock starts counting down the “wait” number of seconds specified (0.1 sec to $24*60*60 - .01$ sec) until it reaches zero. At that time an end-of-line branch is taken to the on-delay service routine specified by the “label” parameter. The on-delay service routine must terminate with a `dret` statement to return to the main program.

Executing the on-delay statement with no parameters disables pending on-delay interrupts and service.

The On Clock-Match Statement (I/O p. 5-15)

```

on match [alarm time , label]
.
.
mret

```

This statement establishes an “alarm” time interrupt for the internal clock. Once the on match statement is executed, the internal clock checks for a match between the “alarm time” and the current time. When the two times match, an end-of-line branch is taken to the on-match service routine specified by the “label” parameter. The on-match service routine returns to the main program via the `mret` statement.

The allowable range of the “alarm time” is .01 seconds to $24*60*60 - .01$ seconds (.01 seconds less than one day). Executing on-match with no parameters disables any pending match interrupt and service.

Buffered I/O

Terminating I/O Transfers (I/O p. 6-9)

An I/O transfer in progress to or from a select code can be aborted by resetting the interface:

- GPIO-type transfer: `w t c select code , 32`
- HP-IB type transfer: `w t c select code , 31`

I/O Buffer Extended Status (I/O p. 6-11)

The `rds` function can return optional parameters regarding buffer status. The syntax for extended buffer status is:

```
rds ( buffer name [ ,type [ ,empty [ ,fill [ ,dim]]] ] ) → status
```

where “type” is buffer type (0-5), “empty” is the value of the empty pointer, “fill” is the value of the fill pointer, and “dim” is the dimensioned length of the buffer. Note that the value returned as “type” will be negative if the buffer is busy. “Status” is set to -1 if the buffer is busy and set to the fill value $-$ the empty value if the buffer is not busy.

I/O Buffer Pointer Control (I/O p. 6-12)

The `w t c` statement can be used to set values of buffer parameters “fill”, “empty”, and “type”. You cannot execute `w t c` to a busy buffer, nor can the buffer ever be changed from a word-type to a byte-type buffer (or vice-versa). Syntax:

```
w t c buffer name [ ,type [ ,empty [ ,fill] ] ]
```

where “type” is buffer type (0-5), “empty” is the value for the empty pointer, and “fill” is the value for the fill pointer. A value of -1 for any parameter leaves that buffer parameter unchanged. The following relationship must always be true:

$$0 \leq \text{empty} \leq \text{fill} \leq \text{dimension}$$

The `w t c` statement now gives you the capability to re-transmit a buffer that has been emptied. The following example illustrates this:

```
0: buf "A",27,1
1:
2: % " Put data in buffer, and save value of fill pointer"
3: wrt "A","This could be waveform data"ir ds("A")>S
4:
5: tfr "A",705
6:
7: % " Wait for tfr completion"
8: jmp rds("A")#-1
9:
10: % " Rewrite fill pointer and restart transfer"
11: wtc "A",-1,0,S;sto -5
12:
13: end
```

Saving String Buffers to Disc (I/O p. 6-14)

There are now three methods of saving string buffers on a disc: `rcf` and two types of `sprt/rprt`. The `rcf` to disc is the fastest method, and all buffer pointers are preserved as they were for tape. To use `sprt/rprt`, either the buffer status must be written to the disc separately, or the `sprt/rprt` operation must be directed to a binary-type data file ("NBDATA"), which preserves the transfer pointers automatically. The following example illustrates how to save the transfer pointers explicitly (non "BDATA"-type data files):

```
rdsc ("Buffer",T,E,F)≧L; sprt 1,B$,E,F      Save Empty and Fill pointers with
                                           the string buffer.
```

(`B$` is the string buffer "Buffer"). Now to retrieve and restore the buffer status, use `wtc`:

```
sread 1,B$,E,F; wtc "Buffer",-1,E,F
```

The `wtc` leaves Type unchanged, and sets Empty and Fill to their original values.

The following examples illustrate saving only the active portion of a buffer (so you won't have to save 10 kbytes of string buffer when only 50 bytes are used).

- Byte-type buffer:

```
rdsc("Buffer",T,E,F)≧L; sprt 1,E,F,B#[E+1,F]
sread 1,E,F; sread 1,B#[E+1,F]; wtc "Buffer",-1,E,F
```

- Word-type buffer:

```
rdsc("Buffer",T,E,F)≧L; sprt 1,E,F,B#[2E+1,2F]
sread 1,E,F; sread 1,B#[2E+2F]; wtc "Buffer",-1,E,F
```

Plotter Control (I/O p. 7-4)

Internal Graphics Control

Series 200 computers can display graphics on their internal CRT. The Model 226 can display 300 vertical by 400 horizontal dots or pixels; Model 216 and 236 can display 390 vertical by 512 horizontal pixels. Operation of the internal graphics is much the same as operation of an external graphics device, only faster. Programs designed to drive an external plotter will operate nearly identically with the internal graphics. Obviously, there are certain physical differences between the two types of devices, such as pen selection, line resolution, and device size.

There are two capabilities of external plotters or digitizers not offered by the internal graphics device: digitize (`digit`) and line-type (`line`). These statements are not illegal when directed to the internal graphics screen, they merely have no effect. Thus, multi-line clusters that have a definite visual impact with a four-color plotter may appear confusing on the internal graphics. Additionally, the 9825 character set is not selected on the internal graphics screen when `plot` is executed.

All of the operations associated with graphics on the internal CRT can also be performed on an external color CRT using the 98627A Color Video Interface.

The Plotter Select Code Statement

Selecting the graphics screen as a plotting device is done by executing

```
plot 16
```

where 16 is the select code of the internal graphics screen. If an external plotter is selected, it must be an HPGL-type plotting device such as a 9872 or 7225 plotter, or a color video interface (98627A).

An extended syntax for the 98627A Color Video Interface is provided to allow maximum flexibility for color monitors:

```
plot select code [,TV]
```

The TV parameter determines how the interface will be set up. The values allowed are:

Value	Monitor Type
1	U.S. Standard, 512x390
2	European Standard, 512x390
3	U.S. TV, 512x474
4	European TV, 512x512
5	Hi-resolution, 512x512
6	JVC monitors
7	not used

The default setting for the TV parameter is 1.

The Pen Select Statement

The pen select statement `PEN#` has different meanings, dependent on the current plotting device. For the internal graphics screen, the `PEN#` syntax is:

```
PEN# type
```

where “type” selects pen off (0), pen on (>0), eraser (−1), and exclusive-OR (−2). An exclusive-OR pen turns off dots that are on and turns on dots that are off.

External plotter pen control is unchanged for HPGL plotters such as the 7225 and 9872.

For the color video interface, the pen number is a value between −15 and 15:

98627A Video Interface Pen Numbers

Color	Store	Erase	OR	XOR
no-pen	0	0	8	−8
white	1	−1	9	−9
red	2	−2	10	−10
yellow	3	−3	11	−11
green	4	−4	12	−12
cyan	5	−5	13	−13
blue	6	−6	14	−14
magenta	7	−7	15	−15

A program can be written to run all graphics devices if −9 is used for the XOR operation, −1 is used for erase, 0 is used for no pen, and 1 is used for white pen.

Graphics Screen Control Statements

The graphics screen may be turned on and turned off with the following two statements:

```
g o f f          (turn off graphics screen)
g o n           (turn on graphics screen)
```

These statements have no effect on the contents of the graphics memory. On the color video interface, these two commands are used to turn the video off or on.

To clear the graphic screen or current “plotter is” device, use the clear graphics statement:

```
g c l r          (clear graphics)
```


The Dump Graphics Statement

A hard copy printout of the graphics screen can be obtained with a dot-matrix printer compatible with the HP Raster Scan Standard, such as the HP 9876 and HP 2631G printers. The following statement implements the hard-copy graphics dump:

```
sdump [select code]
```

The select code must include the bus address of the printer if an HP-IB printer is used, for example, "sdump 705". If the select code parameter is omitted, the system printer (prtsc) device is the default.

If the color video interface is the current plotting device, the graphics dump statement will "or" the three color planes (red, green, blue) and dump the image to the current print device.

The Graphics Pointer (Cursor) Statement

Interactive graphics can be achieved with the use of either an external digitizer (or data tablet) or the keyboard and knob. A graphics cursor can be drawn at point x,y using the statement:

```
sptr x-coordinate , y-coordinate [ ,type]
```

The x and y coordinates must be specified, and are measured in the units set by the `sc1` and `ofs` statements. The "type" parameter specifies whether the cursor is to be turned OFF ("type" = 0) or ON ("type" # 0). Default ("type" not specified) is ON.

The pointer can be plotted on the internal CRT or on an external CRT using the color video interface.

Binary Graphics

The high-speed nature of CRT graphics makes certain speed enhancements even more desirable for applications such as animation or real-time information display. These enhancements are EXTENSIONS to the plotting capability of the internal or an external CRT, and as such cannot be used with an external plotter. The first extension to be discussed is binary plot.

The Binary Plot Statement

`bPlt` string expression , bytes-per-line[,function]

The binary plot statement loads the binary information contained in the string expression into the graphics memory at the current pen position. Because there are no computations involved, this is a very high speed operation. The specified number of bytes are taken from the string and placed on a line, with the high-order bit of each eight-bit string character corresponding to the leftmost pixel of eight pixels. The plot statement can be used to position the pen to the upper left of the pixel. To conceptualize this, suppose that you want to draw the following character on the screen:

Character	Binary	Decimal
	00000001	(1)
	01111110	(126)
	10100100	(164)
	00100100	(36)
	00100100	(36)
	00100100	(36)
	00100100	(36)
	01001000	(72)

The graphics representation is on the left, the binary representation is in the middle, and the decimal equivalent is to the right. The following statements assemble the string and `bPlt` it:

```
0: dim A$(80);Psc 16;scrlrPen;Plt 3,3
1:
2:
3: char(1)&char(126)&char(164)&char(36)→A$
4: A$&char(36)&char(36)&char(36)&char(72)→A$
5: bPlt A$,1
```

Note that by specifying one byte-per-line, each byte, or character, of A\$ corresponds to one line of graphics pixels. Specifying two bytes-per line causes two bytes of A\$ to be drawn on a line, and 16 pixels are then defined on the line. To define 3 lines of 16 pixels per line requires a 6 byte string, with a correspondence as shown below:

```
A$[1,1] <.....> <.....> A$[2,2]
A$[3,3] <.....> <.....> A$[4,4]
A$[5,5] <.....> <.....> A$[6,6]
```

The optional “function” parameter specifies the interaction of the `bPlt` string with graphics memory. If not specified, the `bPlt` statement performs a binary OR of the string expression with graphics memory.

function=0: performs a binary OR function (default)
function=1: performs a binary AND function
function=2: performs a binary EOR function
function=3: performs a STORE (overwrite) function

The `bPlt` statement does not affect the pen coordinates (x,y) or the pen position (up or down).

The complete syntax for the binary plot statement is:

```
bplot string expression [ , string expression , string expression ] , bytes-per-line [ , function ]
```

The optional two additional string expressions allow for red, green, blue color planes respectively, for the color video interface. If only one string is shown as in `bplot A$`, then black and white is assumed.

If all three strings are used for `bplotting` on the internal CRT, the three planes will be ORed together before the `bplot` operation.

Note that if three string expressions such as `bplot R$, G$, B$` are plotted on the internal CRT, the operation will slow down the `bplot` operation.

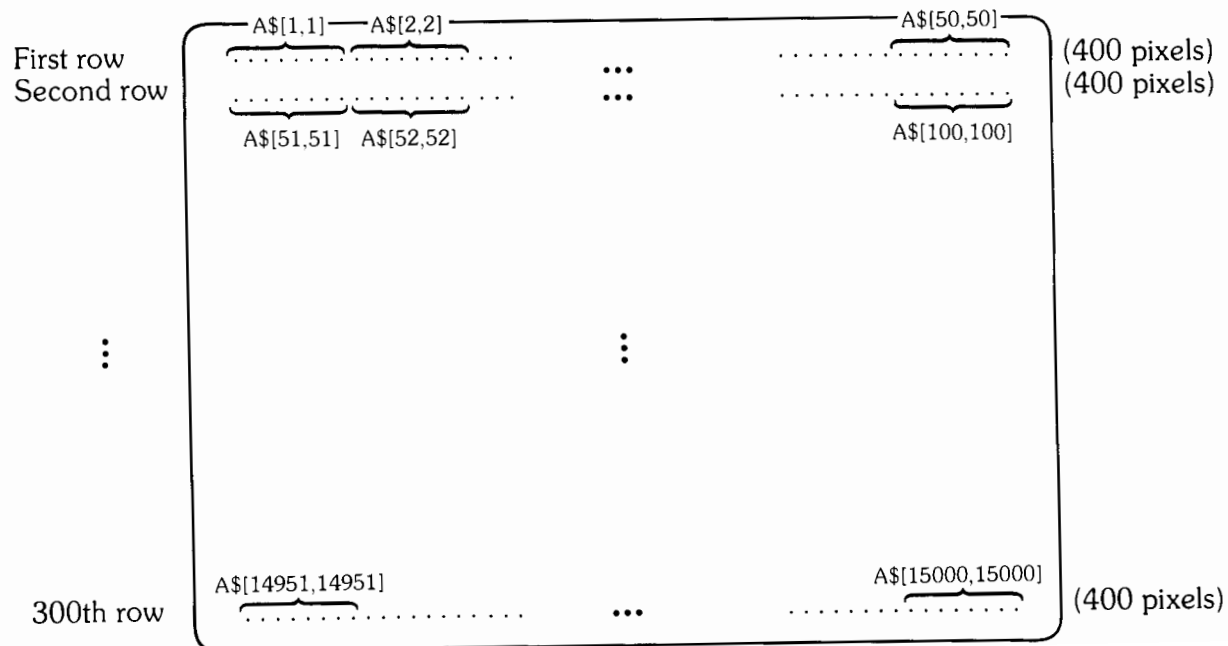
The Graphics Load and Graphics Store Statements

Two other statements facilitate high-speed graphics displays: `gstore` and `gload`. With these two statements it is possible to plot a series of pictures (using standard plotter commands), store each picture into a string variable, then later recall the picture series at high speed by sequentially loading each string variable into graphics memory. Because no computations or decisions are being made when loading the strings into the graphics display, this occurs very rapidly.

The syntax for these statements is shown below:

```
gstore string variable
gload string expression
```

The `gstore` statement saves the graphics display into a string variable with the leftmost bit of the top line corresponding to the most significant bit of the first character in the string. The following diagram illustrates this for the Model 226 CRT:



As you can see, a 15,000 byte string is required to hold the entire 50 byte by 300 line display. If a smaller string is specified, the display will be saved up to the point where the string becomes full.

The `gload` statement performs the converse operation of the `gstore` statement, loading the graphics memory from the specified string expression. The transfer always begins at the upper-left corner of the display and with the first byte of the string expression, filling the screen until either the string length is reached or until the screen is full.

The graphics strings can be loaded from disc and stored to disc, or saved in memory and sequenced through the graphics display. This sequencing can be used to create the illusion of motion or more simply to present information to the viewer more rapidly than is possible by using plotting commands.

An extended syntax of the `gload` and `gstore` statements is provided for the color video interface. The extended syntax is:

```
gstore string variableRED , string variableGREEN , string variableBLUE
gload string expressionRED , string expressionGREEN , string expressionBLUE
```

To illustrate these two commands here is an example:

```
dim A$[3,24960]
psc 12
gstore A$[1], A$[2], A$[3]
```

will cause red data to go into A\$[1], green data to go into A\$[2] and blue data to go into A\$[3].

Notes

Chapter 4

Powerfail Programming

Introduction

Aside from the advantage of being able to sustain short power “glitches” automatically, the powerfail option provides some powerful features that are found only in more expensive systems. Among these are the capability to programmatically detect a power failure and to backup all data and programs before any loss occurs. This chapter details these power-failure features along with the statements and functions provided to take advantage of this unique protection feature.

What Is It?

Put very simply, the powerfail system is a big rechargeable battery that automatically comes “on line” for up to 60 seconds in the event of a brown-out, “glitch” or power failure. When the power drops, the internal clock time is stored into the powerfail unit. The powerfail unit generates a system interrupt which can be detected in software. This interrupt can be used to trigger several actions, such as to store away the data to disc.

The sealed NICAD batteries trickle charge as long as the computer is switched on and ac power is up. The recharge rates for the NICAD depend on the temperature:

<u>Temperature Range</u>	<u>Max. Recharge Time</u>
0°C to 25°C	14 hrs.
26°C to 35°C	24 hrs.
36°C to 40°C	100 hrs.

A full charge will sustain a fully loaded computer through 2 power fail/reboot cycles spaced close together. The number of cycles that are possible on a single full charge depends on the complement of interfaces and memory cards as well as the amount of internal disc access during the power failure. A full 60 seconds is available if a powerfail has not occurred in the last 2 minutes.

When the power is turned on by the power on/off switch, the powerfail backup time defaults to 60 seconds, power failures (glitches) less than 100 milliseconds in duration are not recognized by the system (no interrupt generated) and the time stored in the powerfail unit is loaded into the internal computer clock. All times are accurate to 10 milliseconds.

Since a clock is part of the powerfail unit, it is possible to store the time a power failure occurs as well as the time the power is restored to the system.

Even though a maximum of 61 seconds is available during a power failure, there is no way to maintain power after 59.5 seconds even if AC power is restored. If the protection time is set to less than 60 seconds, subtract 0.5s from this time, and you can find the period in which AC power can be restored.

Detecting a Power Failure

```

on pfail ["label" [, protection [, glitch]]]
:
fret

```

The `on powerfail` statement is used to set up branching to a powerfail recovery routine or to turn off powerfail interrupts.

Without the optional parameters, `on pfail` turns off powerfail interrupts.

When a label is specified, a powerfail interrupt will cause a branch to the power failure routine to the line with that label.

The protection parameter is the time duration that power backup will be supplied by the nicad battery. This can be programmed from 0 to 60 seconds. The default is 60 seconds which is also the maximum backup duration.

The glitch parameter is the time duration the system waits before issuing a powerfail interrupt. This can have a value of 0 to 60 seconds. This is to prevent very short “glitches” that are not major from causing the system to execute the powerfail routine. The failure return statement, `fret`, should be the last statement in the interrupt routine.

Since the glitch time and protection time start at the same time, make sure that the protection time is greater than the glitch time plus the service routine execution time.

It is logical to place the `on pfail` statement at the very beginning of your program. The powerfail routine can be elsewhere in the program.

```

0: on pfail "outage"           Setup interrupt branch.
1: prt rtime                 Loop and print time.
2: gto -1
:
:
30: "outage":rtime+T         Interrupt Routine where time of
31: asgn "time",1           powerfail is stored.
32: rpt 1,1,T
33: fret

```

The Power Function

```
POWER
```

The power function is useful in detecting whether you are on backup battery power or ac power. The values returned are:

- 1 — ac power is up; powerfail hardware present
- 0 — power failure; nicad is supplying power
- 1 — no powerfail hardware option present, but ac power is up

The Power Shutdown Statement

`PshutdowN`

This statement turns the computer off, just as if the power switch was pressed off then on again. It turns the computer off until ac power is restored then the system reboots. If executed from the keyboard when ac power is present, this turns the computer off and back on again causing the system to go through the reboot procedure.

This statement should be used to reserve the battery charge should several power failures occur in succession.

The Power Time Function

`Ptime (option)`

Depending on the value of option, this function indicates the amount of the 60 second battery time used up (0) or the length of the current power failure (1).

`Ptime (0)` – returns amount of battery time used up.

`Ptime (1)` – returns duration of current power failure.

Using `ptime (0)` you can determine the maximum amount of battery time used. This is useful if power failures occur within 2 minutes of each other. The method used to restore the time to 60 seconds after a powerfail is via a counter. It takes up to two minutes (2 seconds for every second used in previous power failures) to reset this counter to allow a full 60 seconds of battery backup time.

Irregardless of the setting of the glitch time (set by `on pfail`) the internal timer begins the 60 second count when the powerfail is detected, not when the interrupt occurs.

Notes

Chapter 5

Color Interface Programming

The capabilities of the 98627A Color Output Interface when interfaced to a color monitor are similar to those of an external plotter and printer. This chapter goes over some of the programming concepts when using the color interface. The first topic we cover is graphics and the second is alpha.

The statements used to program an external plotter or the internal graphics on the CRT are the same ones used for the color interface. Some of the statements, such as `bplot`, have been enhanced to give access to the multiple color capabilities of this interface.

Unique to HPL are the necessary statements to generate formatted text on the external monitor. This means that program listings, disc catalogs and other printouts can be viewed in color.

One note of importance before we get into programming is the select code of the color interface. The factory setting of the interface select code is 28. Since this is outside the select code range used by HPL, be sure to set it to a lower value; in the program examples, we use select code 12.

The Graphics Instructions

All plotter statements that work with the internal CRT graphics are supported on the color interface. The following lists the statements (described in Chapter 3) which have been extended.

bplt - now has two optional string parameters. One string means black and white. Three strings mean red, green, and blue, respectively. Example: `bplt A$,B$,C$,Z`

gclr - acts on the current plotting device set by the `psc` (plotter select code) statement.

gdump - ORs the three color planes and dumps to the printer set by the `prtsc` (printer select code) statement.

gload/gstore - allows one or three strings in the same manner as `bplt`. `gstore A$[1],A$[2],A$[3]` causes red data to go into `A$[1]`, green data to go into `A$[2]`, and blue data to go into `A$[3]`.

gptr - puts a cursor on the video monitor if it is the plotting device set by the `psc` statement.

gon/goff - turns the video on or off.

pen# - selects the color of the “soft” plotter pen. pen# also allows you to erase, exclusive OR, or inclusive OR the colors together.

psc - the plotter select code statement has been enhanced to allow a wide range of color monitors to be attached to the color video interface.

Since the operation of all of these graphics operations are explained in the HPL Programming chapter, let’s go over using a few of them for programming.

Using the Statements in Programs

Since the computer assumes that you are using an HPGL plotter at 705 (9872 type plotter on HPIB) when it is turned on, the first step is to change these defaults. The psc (plotter select code) statement changes the select code from the default. So, the first statement in your program should be a psc statement.

For the Barco monitor we used, the psc statement was: `psc 12`

Then, to “clear” the monitor off, put a `gclr` statement next in your program.

Now, let’s just draw a few lines: one in red, one in cyan and one in blue. First scale the plotting area so that the end points of the lines are within the CRT limits:

```
sc1 0,100,0,100
```

Now to plot the individual lines:

```
4: pen# 2;% "select red pen"
5: plt 10,10,1;% "move to X=10, Y=10"
6: plt 50,50,2;% "draw to X=50, Y=50"
7: pen# 5;% "select cyan pen"
8: plt 0,50;% "draw to X=0, Y=50"
9: pen# 4;% "select green pen"
10: plt 10,10;% "draw to X=10, Y=10"
```

To save the entire image away, you can use the `gstore` instruction to put the image into three string variables:

```
gstore R$,G$,B$
```

R\$ contains the dot pattern for red, G\$ for green and B\$ for blue. Once the colors are in the string variables, it can again be loaded back onto the screen by the `gload` instruction:

```
gload R$,G$,B$
```

Final Program:

```

0: dim R#[24960],G#[24960],B#[24960]
1: psc 12
2: gclr
3: scl 0,100,0,100
4: pen# 2;% "select red pen"
5: plt 10,10,1;% "move to X=10, Y=10"
6: plt 50,50,2;% "draw to X=50, Y=50"
7: pen# 5;% "select cyan pen"
8: plt 0,50;% "draw to X=0, Y=50"
9: pen# 4;% "select green pen"
10: plt 10,10;% "draw to X=10, Y=10"
11: gstore R#,G#,B#;% "put graphics image into string variables"
12: stp
13: gclr
14: stp
15: gload R#,G#,B#;% "reload string image onto crt"
16: end
*24181

```



Character Animation

The next example shows how to use bplt with the color card and the knob to generate animation. The program is documented to show what each step in the program is doing.

```

0: dim R#[63],G#[63],B#[63],A#[4]
1: fxd 0
2: psc 12;% "Required to do Text/Graphics on color video interface"
3: gclr;% "Clear graphics image on monitor"
4: "Red":for I=1 to 21;% "21 rows in image"
5: for J=1 to 3;% "3 bytes - 24 columns in image"
6: read R;% "load an octal data byte"
7: str(R)->A#;% "put data byte into string"
8: R#&char(val(A#,8))->R#;% "convert octal to decimal"
9: next J
10: next I
11: "Green":for I=1 to 21
12: for J=1 to 3
13: read G
14: str(G)->A#
15: G#&char(val(A#,8))->G#
16: next J;next I
17: "Blue":for I=1 to 21
18: for J=1 to 3
19: read B
20: str(B)->A#
21: B#&char(val(A#,8))->B#
22: next J;next I
23: scl 0,100,0,100
24: pen# -1
25: on knob "move"
26: T+X
27: plt X,50
28: bplt R#,G#,B#,3,2
29: bplt R#,G#,B#,3,2
30: gto 27
31: "move":knob+X+X
32: kret
33: stp
34: "Red Data":
35: data 0,0,0,0,377,0,7,0,340,14,0,30
36: data 10,0,10,20,0,4,21,201,204,41,201,204
37: data 40,0,4,40,0,4,40,0,4,41,0,204
38: data 40,201,4,20,102,4,20,74,10,10,0,10
39: data 4,0,20,2,0,40,1,301,300,0,76,0,0,0,0
40: "Green Data":
41: data 0,0,0,0,0,0,0,0,0,0,0,0
42: data 0,0,0,0,0,0,1,201,200,1,201,200
43: data 0,0,0,0,0,0,0,0,0,0,0,0

```

```

44: data 0,0,0,0,0,0,0,0,0,0,0,0,0
45: data 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
46: "Blue Data":
47: data 0,0,0,0,0,0,0,0,0,0,0,0,0
48: data 0,0,0,0,0,0,0,0,0,0,0,0,0
49: data 0,0,0,0,0,0,0,0,0,0,1,0,200
50: data 0,201,0,0,102,0,0,74,0,0,0,0
51: data 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
*12092

```

This program makes a three-color happy face. Using the knob allows you to move the happy face back and forth across the middle of the screen.

Alpha Text Programming

In this section, the text capabilities of the color interface are described. The examples show how to print text in different colors and how to print text in any location on the monitor.

The distinction between alpha and graphics is the way characters are placed on the monitor. Using the `lbl` (label) statement, the character is plotted, using short vectors, on the CRT. This is somewhat slow. Using `prt` (print) or `list#` (or several other statements), characters are generated from dots, just as you would see on a dot-matrix printer. Labeled characters, even though they are slower, can change size using `csiz` (character size) and can be plotted in different directions, too. Text which is printed cannot be enlarged or rotated, only printed as on a printer.

Even though we are describing alpha operations, it is important that you place the `psc` statement and `prtsc` in your programs even if you are not doing graphics.

All of the alpha operations described use the `wti` (write interface) statement to access registers on the color interface. Before you can access these registers, first execute:

```
w t i 0 , select code
```

where `select code` is the select code of your color interface. This “opens” the interface to give access to the individual registers. Once the `w t i 0 , select code` operation is performed, other `wti` or `rdi` operations access that interface, so only one `wti 0, select code` operations is needed to access several registers.

Setting the Alpha Print Position

The internal CRT uses the `tabxy` statement to set the alpha print position. On the color interface, the syntax for the same operation is:

```
w t i 1 , alpha print position
```

The alpha print position parameter is a 16-bit word of the format:

REGISTER 1		Alpha Print Position	
Most Significant Bit		Least Significant Bit	
Bit 15	Bit 6	Bit 5	Bit 0
y Position		x Position	

This is a read/write register. So, if you need to know what the current print position is, you can check it by using:

```
rdi 1, V
```

and decoding the value of V to find the current x,y coordinates:

```
V mod 64 → X          (x coordinate)
int (V/64) → Y        (y coordinate)
```

The 0,0 location on the CRT is the upper left corner of the CRT. The maximum characters per line is 64 (0 to 63) as you might expect. The number of lines (y positions) depends on the number of scan lines (this depends on the monitor you have). To calculate the maximum number of y positions use this:

```
y positions = int (height scan lines/12)
```

For the Barco monitor we used, the number of scan lines is 390. So, the number of lines is:

```
int (390/12) or 32
```

This means that lines 0 through 31 can be used.

The following example is a simple program that asks for an x,y coordinate and places an asterisk at the x,y position on the monitor.

```
0: "Color Video Alpha Print Position":
1: psc 12;% "Required to use Text/Graphics on color video i/f"
2: prtsc 12;% "Printer is the color video interface"
3: fxd 0
4: prt char(12);% "Clear screen with form feed"
5: "loop":ent "Enter X print position (0 to 63): ",X
6: if X<0 or X>63;dsp "%";gto "loop"
7: "loop1":ent "Enter Y print position (0 to 33): ",Y
8: if Y<0 or Y>33;dsp "%";gto "loop1"
9: if X<0 or X>63 or Y<0 or Y>33;dsp "%";gto "loop"
10: X+Y*2^6+P;% "Compute value to store in reg. 1"
11: wti 0,12;% "Select color i/f for register access"
12: wti 1,P;% "Set alpha print position on color i/f"
13: prt "*"
14: gto "loop"
15: end
*22355
```

Writing to the Holding Register and CRT

To place a character into the alpha holding register, the syntax is:

```
w t i 4 , character
```

The character parameter is an 8-bit character that corresponds to an ASCII character. If you were to execute:

```
w t i 4 , 65
```

the letter "A" would be stored in the alpha holding register. To transfer this to the CRT, the following syntax is used:

```
w t i 7 , dummy parameter
```

Any 16-bit value can be used for the dummy parameter since the value isn't used. The wti operation to register 7 is often referred to as the trigger operation, transferring data from register 4 to the display.

The Interface Set-up Instruction

The configuration of the Color Video Interface can be set up by writing to register 5 of the color interface. The syntax for the setup operation is:

```
w t i 5 , 8-bit register value
```

The diagram for this register is:

REGISTER 5					Interface Setup		
Most Significant Bit					Least Significant Bit		
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Scroll Bit 1 = $\overline{\text{Scroll}}$ 0 = Scroll (default)	Don't Care	Reset 1 = $\overline{\text{Reset}}$ 0 = Reset	Graphics On/Off 1 = gon 0 = goff	Reset Monitor Type	Monitor Type 0 Not Used 1 U.S. Standard (default) 2 European Standard 3 U.S. TV 4 European TV 5 Hires 6 JVC 7 Not Used		
Value = 128	Value = 64	Value = 32	Value = 16	Value = 8	Value = 4	Value = 2	Value = 1

To reset the interface, bit 5 is set. All other bits are ignored if bit 5 is set. To set this bit use:

```
w t i 5, val ("00100000",2)
```

The scroll bit, bit 7 allows you to provide scrolling where new lines of text stack up from the bottom and move the top line off the CRT. Also, you can turn off scrolling. In this mode, the text sent to the CRT “wraps”. This means that after the CRT is filled, new lines of text write over from the top down.

Bit 4 does the same operation as the gon and goff instructions performed. We recommend using gon-goff for compatibility reasons.

Bit 3 is only useful if bits 0 thru 2 are being changed. The psc statement performs the same operation and for compatibility should be used instead.

Select Text Color and Inverse Video

Writing to register 6 on the color video interface sets the color that text appears on the CRT. In addition, the text can be displayed in inverse video if desired.

```
w t i 6, 8-bit register value
```

The diagram of register 6 is:

REGISTER 6				Text Color & Inverse Video			
Most Significant Bit				Least Significant Bit			
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not Used	Red 1 = $\overline{\text{Red}}$ 0 = Red	Green 1 = $\overline{\text{Green}}$ 0 = Green	Blue 1 = $\overline{\text{Blue}}$ 0 = Blue	Not Used			Inverse Video 1 = Inverse Video 0 = Normal Video
Value = 128	Value = 64	Value = 32	Value = 16	Value = 8	Value = 4	Value = 2	Value = 1

The default value is 0: normal video, with red, green and blue on (white text).

The following example shows the different combinations of color and inverse video.

```

0: "Color Video Text":
1: psc 12;% "Set select code for color video interface"
2: dim Z#[4],T#[4]; " "+Z#
3: prtsc 12;% "Printer is the color video interface"
4: fxd 0
5: prt char(12);% "Clear screen with form feed"
6: wti 0,12;% "Select color i/f for register access"
7: for X=0 to 1
8: for I=X to 2^7-1 by 16;% "Loop to set bits 4, 5 & 6 of color video i/f"
9: len(str(I)+T#)+T
10: wti 6,0;% "Select White text in reg. 6"
11: wtb 12,Z#[I]&T#&" ";% "Print Register value"
12: wti 6,I;% "Set color/inverse video on color i/f"
13: prt "RaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz"
14: next I
15: next X
16: wti 0,12;wti 6,0;% "Restore reg. 6 to white on black"
17: end
*13492

```

Summary of Alpha Instructions

<code>wti 0</code> , select code	Gives access to the color video instructions for the color interface at the specified select code.
<code>wti 1</code> , position	Sets the x and y position for printing text (similar to <code>tabxy</code>).
<code>rdi 1</code> , variable	Reads the current text position into the variable.
<code>wti 4</code> , character	Places the ASCII character into the holding register. The trigger operation (<code>wti 7,0</code>) transfers this character to the CRT.
<code>wti 5</code> , value	Sets parameters on the color interface such as TV type, and scroll or wrap.
<code>wti 6</code> , value	Selects the text color and inverse video or normal video display.
<code>wti 7</code> , dummy value	Used in conjunction with register 4, this operation triggers the interface and transfers the ASCII character in the holding register to the display.

Appendix A

Disc Programming Technical Appendix

Supported Device/Format Combinations

HPL 2.1 supports many HP disc drives and two different mass storage directory formats. The supported drives are:

- Model 226 or 236 internal discs (5¼-inch double-sided discs)
- 82900 Series drive (5¼-inch double-sided discs)
- 9885 drive (8-inch single-sided discs)
- 9895 drive (8-inch double or single-sided discs)
- 9133V Opt. 004 Winchester/3½-inch micro-disc drive



The supported directory formats are:

- HP Corporate Logical Interchange Format (LIF)¹
- 9825 format (also 9835/9845)

The 9825 format is supported on the 9885 and 9895 drives as "F" and "H" device format specifiers, providing backward compatibility with the 9825/9835/9845 computers. Program, data, and key files are fully compatible with 9825 files. Data files are compatible with 9835/9845 data files, provided the logical record length is 256 bytes per record (the default). Moreover, 9825 bootstraps are not required on discs, enabling compatibility with 9835/9845 initialized discs.

LIF is supported on the internal, 9133V and 8290x-series drives, enabling data interchange with Series 200 BASIC discs and others via ASCII (INTERCHANGE) files. LIF is also supported on the 9885 and 9895 discs, both for the unique features it offers, and for compatibility with future mainframes.

Device Format Specifier	Disc Type	Disc Format	Select Code Range	HP-IB Address Range	Default Select Code Value	Unit Number Range
I	Internal	LIF ¹	—	—	—	0*
M	8290x	LIF ¹	1-15	0-7	700	0-3
M	9133V micro disc	LIF ¹	1-15	0-7	700	0
F	9885	9825 Compatible	1-15(not 7)	—	8	0-3
G	9885	LIF ¹	1-15(not 7)	—	8	0-3
H	9895	9825 Compatible	1-15	0-7	707	0-3
H	9133V Winchester	9825 Compatible	1-15	0-7	700	0-3
J	9895	LIF ¹	1-15	0-7	707	0-3
J	9133V Winchester	LIF ¹	1-15	0-7	700	0-3

* Unit numbers 0 and 1 on a Model 236.

¹ LIF: Hewlett-Packard Logical Interchange Format: File type ASCII is compatible with BASIC language ASCII files and with HP 2642A Terminal files.

HPL Implementation of LIF

HP Corporate Logical Interchange Format (LIF) is a standard defining the logical structure of mass storage media, notably discs. The standard concerns itself with:

1. Layout of file structure on the disc
2. Location and configuration of directories
3. Interchange data types
4. Location and configuration of the volume label

Most of these concerns are transparent to the user, since they are automatically handled by the HPL operating system, however, there are several topics the user should have a knowledge of with respect to how LIF affects him. These topics are discussed below.

LIF Disc Structure

The structure of LIF discs is simple compared to 9825-format discs. Sector 0 contains the LIF volume label (system record); sector 1 contains all 0s as defined by the standard; and sector 2 contains the first record of the directory. The user file space begins following the last record of the directory, and spans the remainder of the disc. There is no backup directory.

Beginning with HPL 2.0, the default directory size is the number of records required to fill up the first track. For 5¼-inch discs (16 sectors per track), this yields 14 directory records; for 8-inch discs (30 sectors per track), this yields 28 directory records. The initialize statement has been extended to allow the user to specify how many directory records he wants. Refer to the section on the initialize statement for details.

Rules for Legal File Names

The LIF standard has rather stringent restrictions for file names. File names are from 1 to 10 characters in length; the characters are to be of the set of uppercase English alphabets and the digits 0 through 9; furthermore, the first character is to be alphabetic.

HPL's LIF implementation has relaxed the file name restrictions, allowing 9825 type file names to be specified. If you wish to interchange a Series 200 LIF disc with that of another computer, however, you **MUST** adhere to the restrictions described in the paragraph above!

The test for legal file names is applied only when Series 200 HPL creates a file name entry, not when HPL merely accesses a file name. Any file name can be referenced, as long as it does not contain a colon, a null character, or imbedded blanks. Blanks preceding, imbedded in, or trailing the file name are ignored.

If the user wishes for his disc to conform fully with the LIF standard, then it is his responsibility to name his files accordingly.

LIF Directory Entries Support Larger File Sizes

Due to limitations with the 9825-format directory, most files on these discs are restricted to be 65536 bytes or smaller in size. This applies to PROGRAM, KEYS, SBDATA, NBDATA, and BINARY files; this does not apply to TDATA files! Since the maximum user memory size for the 9825 was 62K bytes, this size restriction was never a problem. However, with Series 200 HPL supporting a MUCH larger user memory size, it is easily possible to run into the 65K byte limit on 9825-compatible discs.

Because of larger fields in the LIF directory, LIF discs do NOT have this file size restriction; and are capable of handling any size file required by Series 200 HPL.

LIF Discs May Require More Frequent Repacking

Under certain conditions, attempting to create a file on a LIF disc may result in error D8, (insufficient storage space on disc), even though the catalog listing shows that enough space is available. This same error can occur with 9825-format discs; stemming from the fact that the available space may not be contiguous. If a repack is performed (with the `REPK` statement), the available space will all become contiguous, and files can be created to consume the remaining available space.

With LIF discs, though, there is an additional complicating factor associated with the creation of files: the order of the actual files is required to be exactly the same as the order of the directory entries of those files. Thus, even if a large enough block of contiguous space exists out in the user area, an open slot must also exist in the proper place in the LIF directory, else error D8 will be issued. A simple disc repack will cure this problem, allowing the creation of the additional files. The bottom line is: LIF discs may require repacking more often than do 9825-format discs.

LIF ASCII Files

“ASCII” (or interchange) files are defined by the LIF standard, and are fully supported beginning with HPL 2.0. “ASCII” files created by all mainframes that implement the LIF standard should be completely interchangeable. LIF “ASCII” files are the primary data interchange mechanism between Series 200 HPL and BASIC. ASCII files are not supported on 9825-format discs.

Creating ASCII Files

ASCII files are created with the `OPEN` statement. Refer to the section for the `OPEN` statement for a discussion of syntax.

Example:

```
OPEN "ascii_file",10,"ASCII"
```

Assigning ASCII Files

ASCII files are assigned with either the `files` or the `assign` statements, exactly in the same manner as typed data files are assigned, with the exception that if a return variable is present with the assign statement, it will return a value of 10 instead of a value of 0.

Examples:

```
files ascii_file
assign "ascii_file:M70Z,2",2,-1,R
```

Accessing ASCII Files

ASCII files are accessed via the `sprt` and `sread` statements. ASCII files are, by definition, strictly serial in nature. Any attempt, with one exception, to access them with `rprt` or `rread` will result in an error. The one exception is "`rread N,i`"; it is allowed for the purpose of resetting the file pointer back to the beginning of the file. All syntaxes and variable list items allowed on typed data files with `sprt` and `sread` and also allowed on ASCII files. This includes simple numerics, numeric arrays, simple strings, string arrays, and others. As for typed data files, arrays are printed and read on an individual element basis.

Autoverification of `sprt` to ASCII files is governed by `von/voff`.

Each data element in an ASCII file is, by definition, a string of ASCII characters. Though referred to as an "ASCII character", each character is allowed to have any 8-bit binary value (0 to 255 if interpreted as 8 bit unsigned; -128 to 127 if interpreted as 7-bit signed). Numerics printed to an ASCII file undergo a BCD to ASCII conversion similar to the `str` function conversion. The precision and format are determined by the current `fxd/flt` setting. Unlike the `str` function, however, the "E" produced as a result of a `flt` setting will be uppercase as opposed to lowercase; this is required by the LIF standard. Numerics read from an ASCII file undergo an ASCII to BCD conversion similar to the `val` function conversion. Unlike the `val` function, however, either a lowercase or an uppercase "E" will be accepted as being part of a floating point number.

Each ASCII file data element consists of a two-byte header, designating the number of characters in the element, followed by the actual characters. The character count must be between 0 and 32 767, and the header itself does not contribute to the character count. If the count is odd, a null byte is inserted after the last character, so that the next data element header will always begin on a word boundary. A character count of -1 designates a logical end of file. The standard defines all other negative character counts to be illegal; Series 200 HPL treats them as logical end-of-file marks, and no error is issued.

ASCII data elements pay no regard to physical record boundaries (except the physical end of file), and thus they incur no additional overhead when they cross physical record boundaries. However, this implies that the only way to traverse an ASCII file is to start at its beginning, and read or print serially. In other words it is impossible to look at an arbitrary record and determine where data elements begin or end. Random reads and prints are disallowed.

With ASCII file serial prints, the “end” and “ens” parameters work in a way as identical as possible to the way they do with typed data files, considering the fact that ASCII files have no logical end-of-record mark. Here are the rules:

1. If “end” or neither parameter is specified, a logical end of file mark is printed after the last data item, if physically there is room in the file.
2. If “ens” is specified, nothing is written after the last data item. Whatever was there originally will remain. (As is with typed data files, this powerful yet dangerous feature requires a thorough knowledge of file structure and access methods.)

Examples:

```
sprt 1,"hi there!"

dim A#[50],B#[5,100],A[10];sread 1,A,A[*],A#,B#

rread 1,1
```

Using the Type Function with ASCII Files

Use of the type function is allowed with ASCII files. The type function will always return a value of 2 (meaning “full string”) or 3 (meaning “end-of-file mark”).

Example:

```
if type(1)#3;sread 1,A#;prt A#;jmp 0
```

Using On End with ASCII Files

Use of the on end statement is allowed with ASCII files. The on-end branch will be taken if either a physical or logical end of file is reached.

Example:

```
on end 1,"end of file reached"
sread 1,A#;prt A#;jmp 0
```

Binary Data File Support

Binary data files, corresponding to the 9825's tape file types 2 (numeric) and 3 (string), are fully supported beginning with HPL 2.0 tape statement emulation. In addition, the HPL disc-oriented data file statements have been extended to allow them to access these same binary data files. This enables the user to utilize binary data files without being forced to use tape statements.

Series 200 HPL's binary data files are in no way compatible with 9835/9845 binary data files. However, they have a similar set of advantages and disadvantages, when compared with typed data files.

Advantages

- Binary data files can be accessed efficiently on discs that are initialized to their suggested minimum interleave factor; efficient TDATA and ASCII file access requires a larger interleave factor. Thus, up to a 2x speed performance can be achieved if the disc is initialized to the minimum suggested interleave factor **AND** binary data files are used.
- Strings are recorded in their entirety, rather than by their current length. Thus, with string buffers, the type, empty pointer, and fill pointer are always automatically recorded along with the string.

Disadvantages

- The variables in the variable list must be a contiguous block in memory, as required when recording data to the 9825 tape cartridge.
- The entire file is recorded/loaded as one block operation; individual portions of the file cannot be accessed separately.
- If the file is string or mixed data (SBDATA), the data list for the read operation must be **identical in structure** with the data list used to print the file. There is no way to determine the data list structure of an unknown file.
- Strings are recorded in their entirety. If a string is dimensioned 10,000 bytes long, all 10,000 bytes (plus overhead) will be recorded, even if the current length is only 10.

Creating Binary Data Files

Binary data files are originally created as a NULL files, corresponding to the 9825 tape file type 0. When data is written into the NULL file, its type is changed to either NBDATA or SBDATA, depending upon whether the data is strictly numeric or not. NULL files are created with either the `MARK` statement or the `OPEN` statement. The `OPEN` statement is the more flexible of the two, allowing arbitrary file names (and optional `msus`), whereas the `MARK` statement creates file names corresponding to pseudo track and file numbers.

Examples:

```
OPEN "b_file:M",10,"NULL"

REWITRK 0;MARK 5,2000
```

Assigning Binary Data Files (for Disc-oriented Statement Access)

Binary data files are assigned with either the `files` or the `assign` statements, exactly in the same manner as typed-data files are assigned, with the exception that if a return variable is present with the assign statement, it will return a value other than 0. The possible return values for binary data files are:

- 4 SBDATA (String or Mixed Binary Data)
- 7 NBDATA (Numeric Binary Data)
- 12 NULL (Null - no data written yet)

Examples:

```
files b_file
assign "b_file:F",5
```

Accessing Binary Data Files

Binary data files are accessed with the `sprt` and `sread` statements exclusively. The other disc data statements, `rprt`, `rread`, `type`, and `on end` are not allowed due to the nature of binary data files.

An `sprt` or `sread` access to a binary data file is identical to a `rcf/ldf`, with the following exceptions:

- the “file number” parameter refers to an assigned file number instead of a tape marked file number.
- autoverification of `sprt` is controlled by `von/voff` instead of `ave/avd`.
- hardware errors will result in disc hardware error messages instead of tape hardware error messages.

Assigned File Pointers on 8290x Series Discs

In HPL, when a data file is assigned, a set of internal pointers is set up, pointing to the assigned file’s area on disc. Prints and reads to the data file then use the internal data file pointers, not the disc directory. If HPL detects that the disc drive’s door has been opened, it clears all pointers associated with that drive, since they are now meaningless. Then, if the user attempts to use those file pointers without reassigning them, he gets error F6 (Unassigned data file pointers).

Current 8290x Series drives, however, cannot detect if a door is opened between disc accesses. This makes it impossible for HPL to reliably know when it is necessary to clear the file pointers when discs are swapped. A hazard exists if a user assigns data file pointers to an 8290x Series drive, swaps discs, and then prints using those same file pointers without reassigning them. Instead of error F6 being issued, the print will take place, writing on the new disc where the assigned file existed on the old disc. Unpredictable results can occur, resulting in a loss of programs and/or data!

9885 Disc Access Requires a DMA Card

The 9885 driver on the 9825 used the 9825's built-in DMA channel to access the 9885 disc. The 9885 driver on with Series 200 HPL also needs a DMA channel; however, since the DMA card is an option, it may not be present. If the DMA card is not present, and an attempt is made to access the 9885, error f1 will be issued.

HP-IB Programming Considerations (9895 and 8290x Series Discs)

Most of the HP-IB Programming Considerations mentioned in the *9825 Disc Programming Manual*, 09825-90220, are valid for Series 200 HPL. However, due to hardware and low-level driver differences, items 1, 8, & 9 are not valid for Series 200.

HPL Disc Programming Error Messages

When HPL is unable to access a disc controller, it issues error message f0 instead of flashing "DISK IS DOWN" or "UNABLE TO ACCESS DISC CONTROLLER". This means that the error is now trappable from your user program. There are 9 other new error messages associated with HPL mass storage, as shown below:

- f0 Unable to access disc controller
- f1 9885 driver requires a DMA card present
- f2 Invalid msus syntax; illegal device/format specifier
- f3 Directory entry field overflow
- f4 Illegal structure on LIF disc; cannot be repacked
- f5 Attempted disc copy to significantly larger disc
- f6 Attempted disc copy of 9825-compatible disc to LIF disc or vice-versa
- f7 System record not valid for LIF disc
- f8 System record not valid for 9825-compatible disc
- f9 Statement not implemented

Appendix B

Code Charts

The table on the following page lists the ASCII control codes of the Series 200 command and cursor-control keys. This table is useful when developing programs that use the `PKBD` statement, and also when controlling the computer from a remote ASCII keyboard (such as a terminal).

The Keycode Conversion Table on pages B-3, B-4 is useful for designing a Series 200-to-9825 hardware keycode conversion table. This is necessary only when a 9825 program was designed to work with hardware keycodes (`rdb(0)→K`). To implement the conversion, two methods are possible. One is to use a jump table (requiring no variables, but expensive in memory used). This method is used in the utility "9825 key" for keycode conversion. The other method involves a 256 byte string, with the **position** in the string corresponding to the Series 200 8-bit keycode, and the **character** at that position corresponding to the 9825 keycode.

ASCII Control Codes

CTRL of	ASCII Value	ASCII Character	Key Pressed ¹	Displayed Character ³
@	0	NUL	reserved	N U
A	1	SOH	PAUSE	S H
B	2	STX	REWIND	S X
C	3	ETX	HOME LEFT ←	E X
D	4	EOT	HOME RIGHT →	E T
E	5	ENQ	TO TOP ↑	E Q
F	6	ACK	TO BOTTOM ↓	A K
G	7	BEL	RESULT	B E L
H	8	BS	INSERT LINE	B S
I	9	HT	DELETE LINE	H T
J	10	LF	EXECUTE	L F
K	11	VT	RECALL	V T
L	12	FF	RUN	F F
M	13	CR	ENTER	C R
N	14	SO	CLR TO END	S O
O	15	SI	CLR SCREEN	S I
P	16	DLE	DOWN ARROW	D L
Q	17	DC1	UP ARROW	D 1
R	18	DC2	CLEAR LINE	D 2
S	19	DC3	PRINT ALL	D 3
T	20	DC4	LEFT ARROW	D 4
U	21	NAK	RIGHT ARROW	N K
V	22	SYN	INSERT CHAR	S Y
W	23	ETB	DELETE CHAR	E B
X	24	CAN	STEP	C N
Y	25	EM	CONTINUE	E M
Z	26	SUB	DUMP GRAPHICS	S B
[27	ESC	DISPLAY FUNCTIONS	E C
√	28	FS	EDIT	F S
}	29	GS	CAPS LOCK	G S
^	30	RS	ALPHA	R S
_	31	US	GRAPHICS	U S

¹ This is the pkbd keypress and the key pressed from a remote ASCII keyboard.

² This is the shift(") key on the Numeric Keypad.

³ This is the displayed character if "DISPLAYED FUNCTIONS IS ON".



Keycode Conversion Tables

Series 200 Hardware Keycodes *			ASCII		9825 Hardware Keycodes			Series 200 Hardware Keycodes *			ASCII		9825 Hardware Keycodes		
DEC	OCT	HEX	CHAR	DEC	DEC	OCT	HEX	DEC	OCT	HEX	CHAR	DEC	DEC	OCT	HEX
000	000	00	0	000	000	000	00	076	114	4C	e	101	096	140	60
001	001	01		032	000	000	00	077	115	4D	l	040	040	050	28
002	002	02		032	000	000	00	078	116	4E	l	041	041	051	29
003	003	03		032	000	000	00	079	117	4F	^	094	094	136	5E
004	004	04		032	000	000	00	080	120	50	1	049	049	061	31
005	005	05		032	000	000	00	081	121	51	2	050	050	062	32
006	006	06		032	000	000	00	082	122	52	3	051	051	063	33
007	007	07		032	000	000	00	083	123	53	4	052	052	064	34
008	010	08		032	000	000	00	084	124	54	5	053	053	065	35
009	011	09		032	000	000	00	085	125	55	6	054	054	066	36
010	012	0A		032	000	000	00	086	126	56	7	055	055	067	37
011	013	0B		032	000	000	00	087	127	57	8	056	056	070	38
012	014	0C		032	000	000	00	088	130	58	9	057	057	071	39
013	015	0D		032	3C5	000	00	089	131	59	0	048	048	060	30
014	016	0E		032	000	000	00	090	132	5A	-	045	045	055	2D
015	017	0F		032	000	000	00	091	133	5B	=	061	061	075	3D
016	020	10		032	000	000	00	092	134	5C	[091	184	270	FA
017	021	11		032	000	000	00	093	135	5D]	093	185	271	FB
018	022	12		032	000	000	00	094	136	5E	;	059	059	073	3B
019	023	13		032	000	000	00	095	137	5F	>	039	176	260	FA
020	024	14		032	000	000	00	096	140	60	,	044	044	054	2C
021	025	15		032	000	000	00	097	141	61	.	046	046	056	2E
022	026	16		032	000	000	00	098	142	62	<	047	047	057	2F
023	027	17		032	000	000	00	099	143	63	>	032	032	040	20
024	030	18		032	000	000	00	100	144	64	o	111	111	157	6F
025	031	19	0	125	125	175	7D	101	145	65	p	112	112	160	70
026	032	1A	1	128	065	101	41	102	146	66	k	107	107	153	6B
027	033	1B	2	129	066	102	42	103	147	67	l	108	108	154	6C
028	034	1C	3	130	067	103	43	104	150	68	q	113	113	161	71
029	035	1D	4	133	070	106	46	105	151	69	w	119	119	167	77
030	036	1E	5	134	071	107	47	106	152	6A	e	101	101	145	65
031	037	1F	6	135	072	110	48	107	153	6B	r	114	114	162	72
032	040	20	7	131	068	104	44	108	154	6C	t	116	116	164	74
033	041	21	8	132	069	105	45	109	155	6D	y	121	121	171	79
034	042	22	9	016	016	020	10	110	156	6E	u	117	117	165	75
035	043	23	0	017	017	021	11	111	157	6F	r	105	105	151	69
036	044	24	1	136	073	111	49	112	160	70	a	097	097	141	61
037	045	25	2	137	074	112	4A	113	161	71	s	115	115	163	73
038	046	26	3	020	020	024	14	114	162	72	d	100	100	144	64
039	047	27	4	021	021	025	15	115	163	73	f	102	102	146	66
040	050	28	5	008	008	010	08	116	164	74	g	103	103	147	67
041	051	29	6	009	009	011	09	117	165	75	h	104	104	150	68
042	052	2A	7	011	011	013	0B	118	166	76	j	106	106	152	6A
043	053	2B	8	022	022	026	16	119	167	77	m	109	109	155	6D
044	054	2C	9	023	023	027	17	120	170	78	z	122	122	172	7A
045	055	2D	0	014	000	000	00	121	171	79	x	120	120	170	78
046	056	2E	1	020	020	024	14	122	172	7A	c	099	099	143	63
047	057	2F	2	012	012	014	0C	123	173	7B	v	118	118	166	76
048	060	30	3	028	028	034	1C	124	174	7C	b	098	098	142	62
049	061	31	4	030	000	000	00	125	175	7D	n	110	110	156	6E
050	062	32	5	031	000	000	00	126	176	7E		032	000	000	00
051	063	33	6	024	024	030	18	127	177	7F		032	000	000	00
052	064	34	7	018	018	022	12	128	200	80		032	000	000	00
053	065	35	8	007	007	007	07	129	201	81		032	000	000	00
054	066	36	9	019	019	023	13	130	202	82		032	000	000	00
055	067	37	0	252	000	000	00	131	203	83		032	000	000	00
056	070	38	1	001	001	001	01	132	204	84		032	000	000	00
057	071	39	2	013	013	015	0D	133	205	85		032	000	000	00
058	072	3A	3	025	025	031	19	134	206	86		032	000	000	00
059	073	3B	4	010	010	012	0A	135	207	87		032	000	000	00
060	074	3C	5	048	078	116	4E	136	210	88		032	000	000	00
061	075	3D	6	046	068	130	58	137	211	89		032	000	000	00
062	076	3E	7	044	069	131	59	138	212	8A		032	000	000	00
063	077	3F	8	043	043	053	2B	139	213	8B		032	000	000	00
064	100	40	9	049	079	117	4F	140	214	8C		032	000	000	00
065	101	41	0	050	080	120	50	141	215	8D		032	000	000	00
066	102	42	1	051	081	121	51	142	216	8E		032	000	000	00
067	103	43	2	045	045	055	2D	143	217	8F		032	000	000	00
068	104	44	3	052	082	122	52	144	220	90		032	000	000	00
069	105	45	4	033	033	123	53	145	221	91		032	000	000	00
070	106	46	5	054	084	124	54	146	222	92		032	000	000	00
071	107	47	6	042	042	052	2A	147	223	93		032	000	000	00
072	110	48	7	055	085	125	55	148	224	94		032	000	000	00
073	111	49	8	056	086	126	56	149	225	95		032	000	000	00
074	112	4A	9	057	087	127	57	150	226	96		032	000	000	00
075	113	4B	0	047	047	057	2F	151	227	97		032	000	000	00

* Lower 8 bits of 9826 keycodes only.

B-4 Code Charts

Series 200 Hardware Keycodes *			ASCII		9825 Hardware Keycodes			Series 200 Hardware Keycodes *			ASCII		9825 Hardware Keycodes		
DEC	OCT	HEX	CHAR	DEC	DEC	OCT	HEX	DEC	OCT	HEX	CHAR	DEC	DEC	OCT	HEX
152	230	98	0	029	000	000	00	230	346	E6	K	075	235	353	E8
153	231	99	+	125	125	175	7D	231	347	E7	L	076	236	354	EC
154	232	9A	x	138	075	113	4B	232	350	E8	O	081	241	361	F1
155	233	9B	x	139	076	114	4C	233	351	E9	W	087	247	367	F7
156	234	9C	y	140	193	301	C1	234	352	EA	E	069	229	345	E5
157	235	9D	y	143	196	304	C4	235	353	EB	R	082	242	362	F2
158	236	9E	-	144	197	305	C5	236	354	EC	T	084	244	364	F4
159	237	9F	7	145	198	306	C6	237	355	ED	Y	089	249	371	F9
160	240	A0	0	141	194	302	C2	238	356	EE	U	085	245	365	F5
161	241	A1	1	142	195	303	C3	239	357	EF	I	073	233	351	E9
162	242	A2	2	005	144	220	90	240	360	F0	A	065	225	341	E1
163	243	A3	3	006	145	221	91	241	361	F1	S	083	243	363	F3
164	244	A4	4	146	199	307	C7	242	362	F2	D	068	228	344	E4
165	245	A5	5	147	200	310	C8	243	363	F3	F	070	230	346	E6
166	246	A6	6	003	148	224	94	244	364	F4	G	071	231	347	E7
167	247	A7	7	004	149	225	95	245	365	F5	H	072	232	350	E8
168	250	A8	8	008	136	210	88	246	366	F6	J	074	234	352	EA
169	251	A9	9	009	137	211	89	247	367	F7	N	077	237	355	ED
170	252	AA	0	011	139	213	8B	248	370	F8	Z	090	250	372	FA
171	253	AB	1	022	150	226	96	249	371	F9	X	088	248	370	F8
172	254	AC	2	023	151	227	97	250	372	FA	C	067	227	343	E3
173	255	AD	3	251	000	000	00	251	373	FB	V	086	246	366	F6
174	256	AE	4	020	148	224	94	252	374	FC	B	066	226	342	E2
175	257	AF	5	012	140	214	8C	253	375	FD	N	078	238	356	EE
176	260	B0	6	027	156	234	9C	254	376	FE		032	000	000	00
177	261	B1	7	026	000	000	00	255	377	FF		032	000	000	00
178	262	B2	8	253	000	000	00					032	000	000	00
179	263	B3	9	253	152	230	98								
180	264	B4	0	015	146	222	92								
181	265	B5	1	252	135	207	87								
182	266	B6	2	252	147	223	93								
183	267	B7	3	001	001	001	01								
184	270	B8	4	032	129	201	81								
185	271	B9	5	013	141	215	8D								
186	272	BA	6	025	153	231	99								
187	273	BB	7	010	138	212	8A								
188	274	BC	8	048	206	316	CE								
189	275	BD	9	046	216	330	D8								
190	276	BE	0	044	217	331	D9								
191	277	BF	1	043	171	253	AB								
192	300	C0	2	049	207	317	CF								
193	301	C1	3	050	208	320	D0								
194	302	C2	4	051	209	321	D1								
195	303	C3	5	045	173	255	AD								
196	304	C4	6	052	210	322	D2								
197	305	C5	7	053	211	323	D3								
198	306	C6	8	054	212	324	D4								
199	307	C7	9	042	170	252	AA								
200	310	C8	0	055	213	325	D5								
201	311	C9	1	056	214	326	D6								
202	312	CA	2	057	215	327	D7								
203	313	CB	3	047	175	257	AF								
204	314	CC	4	036	000	000	00								
205	315	CD	5	124	251	373	FB								
206	316	CE	6	092	222	336	DE								
207	317	CF	7	126	000	000	00								
208	320	D0	8	033	177	261	B1								
209	321	D1	9	064	193	267	B7								
210	322	D2	0	035	179	263	B3								
211	323	D3	1	036	180	264	B4								
212	324	D4	2	037	181	265	B5								
213	325	D5	3	094	094	136	5E								
214	326	D6	4	038	182	266	B6								
215	327	D7	5	042	042	052	2A								
216	330	D8	6	040	168	250	A8								
217	331	D9	7	041	169	251	A9								
218	332	DA	8	095	000	000	00								
219	333	DB	9	043	171	253	AB								
220	334	DC	0	123	123	173	7B								
221	335	DD	1	125	125	175	7D								
222	336	DE	2	058	178	262	B2								
223	337	DF	3	034	191	277	BF								
224	340	E0	4	060	172	254	AC								
225	341	E1	5	062	174	256	AE								
226	342	E2	6	063	063	077	3F								
227	343	E3	7	032	160	240	A0								
228	344	E4	8	079	239	357	EF								
229	345	E5	9	080	240	360	F0								

* Lower 8 bits of 9825 keycode only.

Appendix C

Series 200 HPL Differences

This appendix lists the differences between the 1.0, 2.0 and 2.1 versions of Series 200 HPL.

Programs written on HPL 1.0 can be loaded and run under HPL 2.0. All HPL 2.0 programs will run under HPL 2.1 except for the gload change explained below.

The major changes at HPL 2.0 were:

- Enhancements for the Model 236 computer.
- Support for the powerfail option and 98627A Color Video Interface.
- Support for multi-pen plotters.
- Error message trapping.

The next sections describe the changes and additions at HPL 2.0.

The major change at HPL 2.1 was to correct the order of the gload parameters to match the order shown in the manual. See "gload" on the next page.

Mass Storage Operations

The Model 236 Computer has two 5¼ inch disc drives built-in. The right-hand drive is the default drive; be sure to insert the system disc in the right-hand drive to boot the system. To specify each drive within HPL mass storage operations, use these specifiers:

```
:I,0  Right-hand drive
:I,1  Left-hand drive
```

For example:

```
copy ":I,0", "to", ":I,1"
```

copies the entire media from the right-hand drive to the left-hand drive.

Graphics Changes

Gload in HPL 2.0 is about 3 times faster than with HPL 1.0 (28 frames/sec on 9826). Gload from odd-byte boundaries, however, does not change.

Bplt in HPL 2.0 is about twice as fast as with HPL 1.0.

The full National Character sets are supported in labeling.

Color Interface Support

All plotter statements that work with the internal graphics are supported on the color card:

bplt — now has two optional, additional strings as parameters. One string means black and white. Three strings stand for red, green, blue. Three strings on the Series 200 internal screen will OR the three values together before the bplt. For example:

```
bplt R$,G$,B$,Z
```

R\$, G\$, and B\$ give data for red, green and blue color planes, respectively. If G\$ and B\$ are not given, the bplt will be black and white.

gclr — acts on the current plotter is (psc) device.

gdump — ORs the three planes and dumps to the specified device.

gload and gstore — allow one or three strings, as with bplt.
For example:

```
dim A$[3,24960]
```

```
psc 12 (if the color card is on select code 12)
```

```
gstore A$[1],A$[2],A$[3]
```

stores red data in A\$[1], green data in A\$[2] and blue data in A\$[3].

gpvt — puts a cursor on the color video monitor.

gon and goff — turns the video on and off.

pen# — the pen number can be a value from -15 thru 15:

98627A Video Interface Pen Numbers

Color	Store	Erase	OR	XOR
no-pen	0	0	8	-8
white	1	-1	9	-9
red	2	-2	10	-10
yellow	3	-3	11	-11
green	4	-4	12	-12
cyan	5	-5	13	-13
blue	6	-6	14	-14
magenta	7	-7	15	-15

A program can run all graphics devices if -9 is used for the XOR operation, -1 is used for the erase, 0 is used for no-pen and 1 is used for the white pen.

psc — The syntax is:

```
psc select code [ ,TV]
```

The TV parameter configures the interface for:

- 1 U.S. STD, 512 x 390 pixels (default)
- 2 EURO STD, 512 x 390 pixels
- 3 U.S. TV, 512 x 474 pixels
- 4 EURO TV, 512 x 512 pixels
- 5 HI RES, 512 x 512 pixels
- 6 JVC monitors



Color Alpha

Precede these write interface operations with `w t i 0 , select code` to control alpha on the 98627A Interface:

Select Alpha Print Position

```
w t i 1 , expression
```

```
r d i 1 , variable
```

Register 1 is read/write, so you can determine where the next character will go, and/or change the position. Upper-left screen is 0,0. Bits 15 thru 6 are Y position. Bits 5 thru 0 are the X position.

All supported screens have 64 characters across, X value can be from 63 thru 0. The number of lines (Y positions) depends on the number of scan lines. The number of lines allowed is:

$$\text{int} (\text{height scan lines}/12)$$

For example, 512 high has 42 lines.

Write Character to Holding Register

```
w t i 4 , expression
```

Register 4, bits 7 thru 0 is the character (write only); use in conjunction with `w t i 7 , 0`.

Set Up Color Interface

```
w t i 5 , expression
```


Register 5 is write-only:

Bit	Contents
7	1 = no scroll; 0 = scroll (default)
6	don't care
5	1 = reset; 0 = do ops specified by other bits
4	1 = GON; 0 = GOFF
3	1 = update interface type; 0 = don't update
2 } interface type	{ 0,7 = not used 3 = U.S. TV 6 = JVC { 1 = U.S. STD 4 = EURO TV { 2 = EURO STD 5 = HI RES
1 }	
0 }	

Transfer Holding Register to Screen

`w t i 7`, expression

Transfers the character in the holding register to the color monitor. This is used in conjunction with `w t i 4`.

You can do any standard output to the color video interface. The interface recognizes these control characters: backspace, linefeed, vertical tab, and horizontal tab. For example:

```
cat 12
list #12
wrt 12,"HI"
prtsc 12;listk    (color card is set at slect code 12)
```

Note

The 98627A usually uses two select codes. This can be avoided by setting it to select code 6.

Powerfail Support

The powerfail protection defaults to 60 seconds at power-up. The glitch time (time delay before a powerfail is recognized) is set to 60 ms. Power back time is set to 1/2 second.

The `stime` statement (no parameters) transfers battery clock time into the internal clock. The time March 1, 1900, is the same for all languages. Executing `stime 0` sets the time to March 1, 1900.

The `on pfail` statement is modified with HPL 2.0:

```
on pfail [line label [ ,protection [ ,glitch ]]]
```

Executing `on pfail` with no parameters turns off powerfail interrupts. Line label is the pointer to the powerfail routine. Protection is the time to preserve the machine (maximum and default is 60 seconds). Glitch is the time to wait before issuing a powerfail interrupt (default is 100 ms).

Use the `fret` statement to return execution from the powerfail routine.

The power function returns the powerfail state:

- 1 = powerfail installed and operating.
- 0 = power failed.
- 1 = power up but no powerfail hardware.

The pshutdown statement switches the computer off and on again (to re-boot the system).

The ptime function returns the amount of battery time used:

- `dsp ptime (0)` — displays the amount of battery time used.
- `dsp ptime (1)` — displays the length of current powerfail.

Display Control

The Model 216 and 236 display has an 80-column line width. Display enhancements such as inverse video and underlining can be controlled using the crt statement:

```
crt value
```

The value specifies which enhancement(s) to set:

- bit 0 - inverse video
- bit 1 - blinking
- bit 2 - underline
- bit 3 - half bright

The machine function returns details of the host computer configuration:

- `bit (0,machine)` 0 = 80-column alpha; 1 = 50-column alpha
- `bit (1,machine)` 0 = 400 x 300 graphics; 1 = 512 x 390 graphics
- `bit (2,machine)` 0 = no highlights on CRT; 1 = highlights
- `bit (3,machine)` 0 = keyboard present; 1 = no keyboard

Data Transfers

The tfr statement now works to select code 0 (dsp line) and to/from select code 16 (crt print/scroller).

Error Trapping

- `errmsg ("XX")` returns the string "error XX message".
- `errmsg (err , rom)` returns the string "error XX message".
- `errmsg (err , rom , err1)` returns the string "error XX in LLLL message".

String Search

The find statement lists all lines containing a specified string expression to the printer is device:

```
find string expression [ , line1 [ , line2]]
```

The search occurs from line1 thru line2.

On-event Changes

New parameters are added to the on key and on knob statements:

```
on key [ line label [ , flag [ , rate [ , delay ]]]]
```

The rate parameter is the time between repeats when a key is held down. Range is 0 thru 2.55 seconds. Default is 0.08 sec.

The delay parameter is the time before starting to auto-repeat keys. The range is 0.1 thru 2.56 seconds. The default is 0.7 sec.

```
on knob [ line label [ , rate ]]
```

The rate parameter specifies the number of seconds between knob interrupts. The range is from 0.01 thru 2.56 seconds. The default is 0.01 sec.

For both on key and on knob, the defaults for rate and delay are in effect unless a program is running which sets new rate and/or delay values. When a program is paused, the defaults are in effect until after execution resumes and a key or knob is used.

HPL 1.0 Bugs Fixed in HPL 2.0

- Executing tabxy from edit screen destroyed the scroller.
- Scaling on psc 16 was inaccurate for small deltas.
- Interrupt and fast read/write HP-IB transfers could leave the HP-IB in a non-holdoff mode.
- **CLR SCR** key on edit screen could leave the insert cursor in the middle of screen.
- Continuing a null program could hang the machine.
- Loading a program larger than the available memory could hang the machine.
- File names could not use characters greater than ASCII 127.
- bred garbaged a word.
- rss, wsc, and wsm garbaged r5.
- pen; plt X, Y didn't put a dot at X, Y on the internal CRT.
- int (10000000000) returned 99999999999.
- gpnr was clipped by lim and hard-limited screen edges.
- gpnr turned off existing graphics cursor on an error.
- Executing line-feed in the middle of the last scroller line (CRT) could collapse and wrap the scroller memory.
- gpnr was not turned off by reset and was forgotten when a reset was performed.

Human Interface Changes

Capslock of Ä Ö Ü À É Ñ. (Bit 13 of keycodes is now Capslock bit and is used by **asc** function for keycodes generated by **rdb(0)** and **key**.) (Note that the “cap” function still only works on “a-z” and “A-Z”.)

The cursor is non-advancing for diacriticals `´ ˘ ˆ` for a e i o u.

Katakana mode shift in/out on Katakana keyboard (Bit 12 of keycodes is now Katakana bit.) (250 is **pkbd** shift to Katakana. 249 is shift to ASCII.)

Shift-backspace is 127, rubout.

There is a 10 deep recall stack now. Shift-recall/recall is like BASIC. (248 is **pkbd** value for shift-recall.)

Delete-line key puts line on the recall stack.

Error messages are enunciated in English.

Extra edit line on Model 216 and 236.

Form feed (dec. 12) is recognized by the printer/scroller (select code 16).

Keyboard line is not cleared by a stopping program or by an **ent** or **enp** statement. (i.e. edit A\$ is now available by `PKbd A$;ent A$`).

80 character support on Model 216 and 236 including scrolling of CRT highlights.

Errors automatically turn on alpha. (This will not happen if error is trapped by the **on err** statement.)

HPL 2.0 Bugs Fixed in HPL 2.1

- TDATA files copied to a LIF medium can now be accessed with `asgn`.
- Keyboard command `'kbd char(250)` does not put the keyboard into a bogus state.
- The message for error 13 now says "ent" instead of "end".
- A `gload` to the internal Model 216 and 226 graphics display now displays the entire string on the CRT. The bottom twelve lines of the graphics display are not affected.
- A `get` or `chain` command does not destroy the structure of the program in memory when the internal disc drive door is opened.
- The sequence of colors in a `gload A$, B$, C$` or a `gstore A$, B$, C$` now corresponds to the order listed in this manual.
- If an expression's intermediate result causes an underflow, i.e. less than $1e-512$, the system does not hang up.
- A `gstore` of a 512 by 512 graphics screen now does not give error 53.
- A `cat` of a disc in a 9895 drive now always indicates the proper type of disc (single or double sided).
- HP-IB input transfers now terminate correctly. The system does not hang up if EOI comes at the same time as the first byte of data, whether using interrupt, fast handshake, or DMA.
- Using `flt` before doing axes on external plotters now works correctly.
- Unusual error messages do not occur when the GPIO card is misconfigured.
- A reset does not produce a system error if the graphics cursor is on an external color CRT.
- The operation x^y now gives results compatible with the 9825. (See the next item.)
- On a Model 216 with boot ROM 3.0L, the expression x^y does not give Error 76 if both x and y are fractions. Similar fixes were made in the `trig` and logarithmic functions.
- Turning the printer off during a `gdump` does not cause the system to hang up.
- The expression `val(-0)` now gives the correct result, 0, rather than -0.
- A `repack` on a 9134 drive does not destroy string length information.
- The sign of the determinant is now always correct after a matrix inversion.
- A `store` of a modified string does not cause the system to hang up.
- Initializing a 9134A volume to 9825 format does not leave trash in the directory.

HPL 2.0 Bugs Not Fixed

You should be aware of these problems with HPL 2.0 and 2.1:

- A RESET during a running `aclr` displays the last line plus one. No data is lost. A second RESET will correct the display.
- A re-save destroys the file on disc before checking for adequate room on the disc. No data is lost. You should either make suitable space on the current media or insert media which has sufficient space before doing a re-save.
- Misconfiguring the GPIO card's option switches can cause the system to hang up. A RESET will recover from this condition.
- A `killall` on a disc initialized by the Pascal operating system will leave two mutually exclusive directories. Initializing the media will remove the Pascal directory.



Subject Index

a

Alpha 34, 35
Alpha clear (aclr) 34
Alpha dump (adump) 35
Alpha mode 11
Alpha off (aoff) 35
Alpha on (aon) 35
Alpha text color 70,87
Arithmetic 24
Array output (aprt) 45
ASCII codes B-2
ASCII file type 52
ASCII LIF files A-1
Assign (asgn) 48
Autostart 8

b

Backup, disc 6, 54
Battery backup 77
Battery clock 65,77
Beep (pbeep) 33
Binary data files 56
Binary graphics 73
Binary plot (bplt) 73
Binary programs 40, 56
Binary statements, unimplemented 57
Blinking 34
Boot system 45
Buffer empty 68
Buffer full 68
Buffer pointer 68
Buffer status (rds) 68
Buffer storage 69
Buffer types 69

c

Cartridge tape operations 38
Catalog (cat) 39, 52
Clock, internal 65
Clock, powerfail 77
Color video interface 81, C-2
Control codes B-2
Copy files and discs 6, 54
Cross reference (xref) 37
CRT alpha 34
CRT display 10
CRT dump (adump, gdump) 35, 71
CRT graphics 34
CRT highlights (crt) 34
Cursor graphics 71
Cycle return (cret) 66

d

Data file compatibility A-1
Data statement 32
Data transfer 29
Define sfk 18, 44
Delay return statement (dret) 67
Delete files (killall) 53
Differences, HPL 1.0/2.0/2.1 C-1
Digitize (gptr) 72
Directory size A-3
Disc formats 46, A-1
Disc handling 3
Disc initialization 49
Disc read statement (dred) 56
Disc statements, unimplemented 57
Disc type function (dtype) 51
Disc write (dwrt) 56
Disc write statement (dwrt) 56
Display organization 10
DMA (direct memory access) 68
Drive statement 49
Dump alpha (adump) 71
Dump graphics (gdump) 71

e

Edit mode 13
Empty buffer 68
Error messages see condensed reference

f

Fail return statement (fret) 78
File backup 6, 54
File types 52
Find statement 37
Flags 37
Flexible discs 3
Full buffer 68

g

Graphics clear statement (gclr) 71
Graphics dump statement (gdump) 72
Graphics load statement (gload) 74
Graphics mode 11, 70
Graphics off statement (goff) 71
Graphics on statement (gon) 71
Graphics operations 70
Graphics pointer statement (gptr) 72
Graphics store statement (gstore) 74

h

Halfbright 34
HP-IB interface 60
HPGL plotters 70
HPL extensions C-1
HPL syntax see condensed reference

i

I/O operations 58
Ibackup utility 6
Immediate continue sfk 20
Immediate execute sfk 20
initialize disc statement (init) 49
Interleave, disc 50
Internal clock 65
Interrupts 66
Inverse video 34

k

Key buffer (key) 42
Key editing 13
Key labels 18, 41
Keyboards 12
Keycodes B-2
KEYS file type 52
Killall statement 53
Knob statement 41
Knob status statement (kstat) 42

l

Labels, international characters C-1
Labels, key 18, 41
LIF 46, A-1
Load graphics (gload) 74

m

Machine function 33
Mass storage is statement (msi) 48
Mass storage unit specifier (msus) 47
Match return statement (mret) 67
Math functions (sqr, pi) 37
Multi-statement sfk 21

n

NBDATA (N) file type	52
NULL (Z) file type	52

O

On cycle statement	66
On delay statement (on delay)	67
On interrupt statement (oni)	66
On powerfail statement (on pfail)	78
Open statement	53

p

Pen numbers	71
Pen# statement	71
Pi function	37
Plotter select code statement (pcs)	70
Plotter select code statement (psc)	70
Plotters	70
Power function	78
Power shutdown statement (pshutdown)	79
Power time statement (ptime)	79
Powerfail	77
Press keyboard statement (Pkbd)	43
PROGM file type	52
Program transfer	27
Programmable beep statement (pbEEP)	33

r

Read clock (rtime, ptime)	65, 79
Read CRT	36
Read statement (red)	36
Read/data/restore	32
Record binary statement (rcb)	40
Remote keyboard	41
Restore statement (rstr)	32
RS-232 interface	62

s

Save binary statement (saveb)	56
SBDATA (S) file type	52
Scrolling buffer	35
Search (find)	37
Secure	52
Serial interface	62
Set clock (stime)	65
Special function keys (sfk)	18, 44
Square root function (sqr)	37
Status, buffer (rds)	68
Status, knob (kstat)	42
Store graphics (gstore)	75
String function (str)	40
String variables	40
Supported discs	46, A-1
System boot statement (sysboot)	45
System printer statement (prtsc)	36
System programming	41

t

Tabxy statement	35
Tape cartridge operations	38
TDATA file type	52
text, color video	84
Timer, internal	65, 77
Transfers, I/O	68
Type function (type)	A-5

u

Underlining	34
Unimplemented binary statements	57
Unimplemented disc statements	57

v

Value function (val)	40
--------------------------------	----

W

Write control statement (wlc)	68
Write protecting discs.	3

X

Xref statement	37
--------------------------	----

