HP 9800 Computer Systems

HPL Operating Manual and Programming Update

for the HP 9826 Computer

HEWLETT PACKARD

HPL Operating Manual and Programming Update

for the HP 9826 Computer

Manual Part No. 09826-90040
Microfiche No. 09826-99040

Start Here

This manual shows how to install, test and operate your new HP 9826 Computer.

Hewlett-Packard Desktop Computer Division
3404 East Harmony Road, Fort Collins, Colorado 80525
(For World-wide Sales and Service Offices see back of manual)

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.

June 1981...First Edition

Table of Contents

Introduction

This update guide serves to inform the experienced HPL programmer of differences between HP 9825A/B Computers and the HPL Language System of the HP 9826A Computer. As you gain experience using the 9826 HPL language system, you will begin to appreciate the changes made to the HPL programming language for the 9826. In the great majority of instances, these changes were made to enhance or expand the utility of the HPL programming language, and to provide access to some of the new hardware features of the 9826.

What To Do and When

First, read the "Installation" chapter of this manual so you'll be able to successfully turn on the 9826 and start programming when you are ready.

Second, read the chapter titled "Getting Started". This will familiarize you with the differences between operating the 9825 and the 9826 computers.

Third, read the chapter titled "Program Transfer" so you will know how to get your 9825 programs up and running on the 9826.

Fourth, read the chapter titled "9826 HPL Programming" to familiarize yourself with the programming differences and extensions of 9826 HPL.

Now that you know what you have to do, its time to get to it. Good luck!

Chapter 1 Installation

Introduction

The computer can be configured to automatically load a language system from a disc when you switch the computer on. Alternately, the language system can be built-in using read-only memory (called ROM). Regardless of whether HPL is soft-loaded or built-in, it's the same language once loaded.

The soft-loaded system allows a programmer to load one of many languages from a disc into the computer memory. This allows the programmer the flexibility to select the programming language and associated keyboard operating system best suited to his or her needs.

The built-in (ROM) language system is just that, always available immediately after power-up or system reset. There are no delays or extra steps to ready the system. The ROM-based system is the most convenient choice for simply running applications programs. For maximum flexibility, more than one built-in language system can be installed at the same time.

Knowing whether your computer has a soft-loaded or built-in language system is important. If your computer is soft-loaded (the system is loaded from disc at power-up), you need to insert the system disc before switching the computer on. On rare occasions, an error message may require you to reload the system from disc again. If your computer has more than one built-in language system, you need to choose the language system during power-up. Be sure to follow the correct power-up procedure later in this chapter.

IMPORTANT

If you have a soft-loaded system, be sure to produce at least one backup copy of the system disc right after switching the computer on. Refer to Copying Discs later in this section for more information.

Unpacking the Computer

Your HP computer was thoroughly tested and inspected before being shipped to you. All equipment should be in good working order. After removing the computer from its carton, carefully check it and the accessories for any damage caused by transit. You should also check the accessories against the packing list supplied. Notify your HP sales office if any damage is found. Also file a claim with the carrier. If any items are missing, use the reply card supplied to order the item(s) directly from the factory. HP sales office locations are listed at the back of this manual.

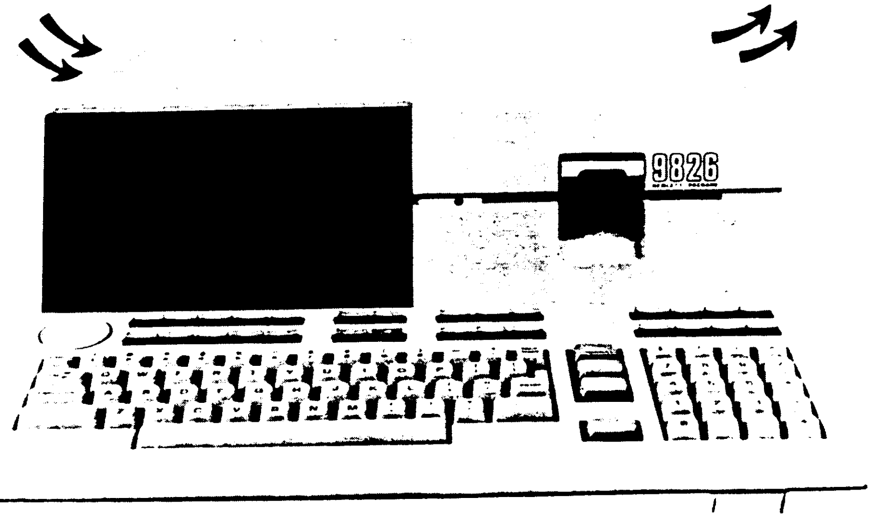
Now you're ready to install the computer, as explained next.

Installation Procedure

Please follow this procedure to install and power-up your computer for the first time. If the computer doesn't power-up as expected, refer to the Computer Testing section in chapter 5.

1. Position the Computer

Place the computer on any convenient work surface. Be sure to leave about 50 mm (two inches) free on each side for air flow through the computer. Do not operate the computer in an area with excessive dust or airborne particulates (smoke).



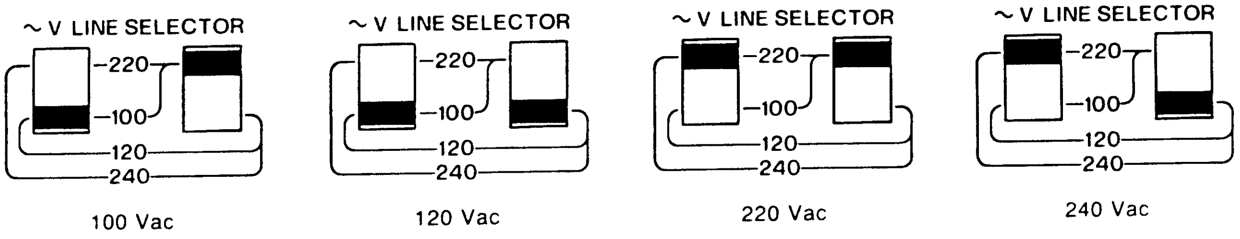
Position the Computer to Allow Free Air Flow

2. Check the Line-voltage Switches

CAUTION

THE COMPUTER CAN BE DAMAGED IF SET FOR 100 VAC OR 120 VAC AND A HIGHER VOLTAGE IS APPLIED. CHECK THE LINE VOLTAGE SWITCHES BEFORE APPLYING POWER.

The computer can be set to operate on one of four nominal line voltages: 100 Vac, 120 Vac, 220 Vac or 240 Vac. The switches on the back of the computer were set to the line voltage in your area when the computer was shipped from the factory. Check the switch settings to ensure they are set correctly:



Line Voltage Switch Settings

3. Check the Fuses

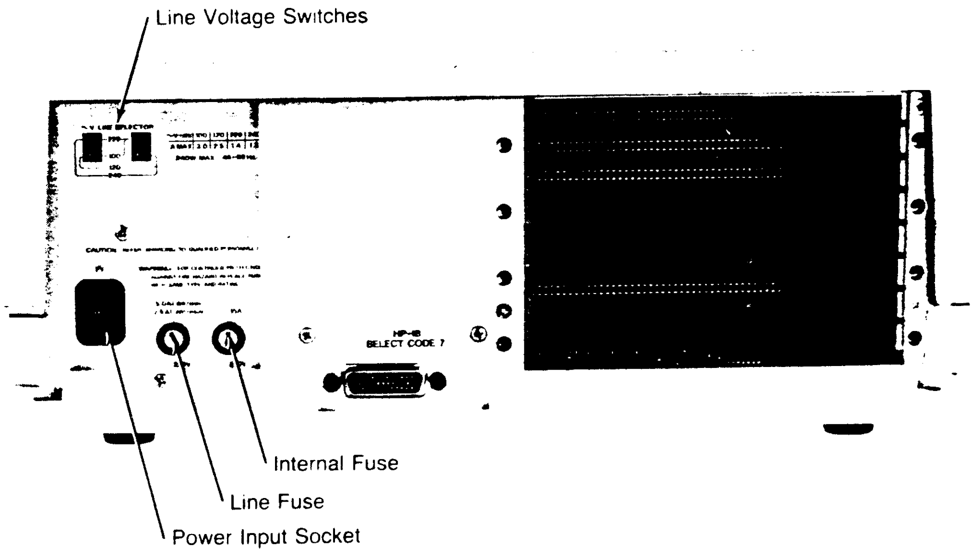
WARNING

TO AVOID THE POSSIBILITY OF SERIOUS INJURY, DISCONNECT THE POWER CORD BEFORE REMOVING OR INSTALLING A FUSE.

The computer has two fuses accessible on the back panel. See the next photo. One fuse protects the entire computer and should match the line voltage, either 100/120 Vac or 220/240 Vac. (See the next table). The other fuse protects the internal power supply; its value is the same for any line voltage: 15 A, HP part number 2110-0054.

Line Fuses

Line Voltage	Fuse Needed	HP Part Number
100, 120	4 A (normal blow)	2110-0055
220, 240	2 A (normal blow)	2110-0002



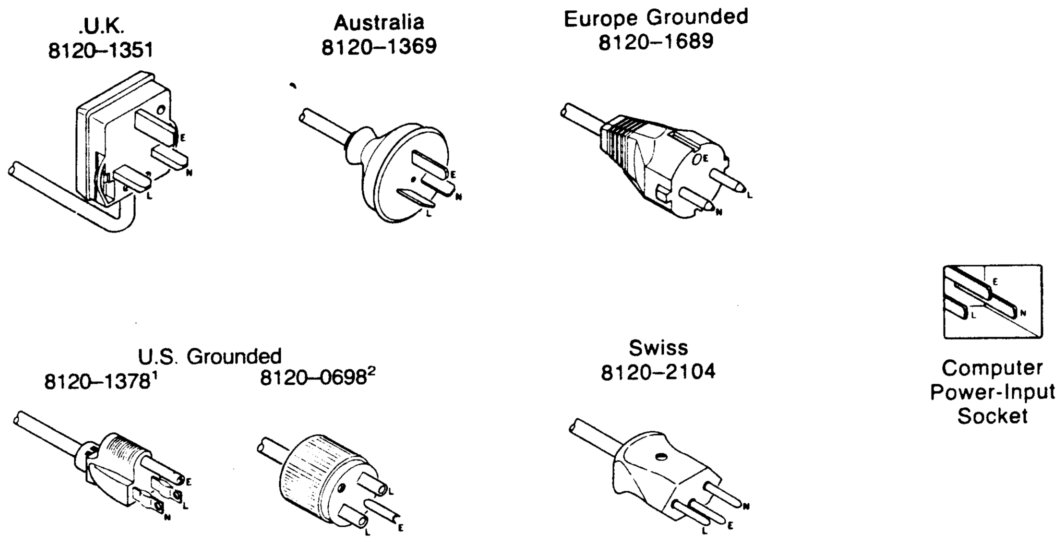
Computer Back Panel

4. Connect the Power Cord

The proper power cord was selected and packed with your computer when it was shipped from the factory. Each cord has a ground connector to protect the operator from electrical shock. Check to be sure you have the correct cord for your power outlet. The available cords are shown next.

WARNING

IF A REPLACEMENT POWER CORD IS NEEDED, IT MUST HAVE THE SAME POLARITY AS THE ORIGINAL. OTHERWISE, EITHER A SAFETY HAZARD FROM ELECTRICAL SHOCK TO PERSONNEL OR EQUIPMENT DAMAGE MAY RESULT.



Available Power Cords

After connecting the power cord to the back panel and the power outlet, go ahead and switch the computer on as explained next.

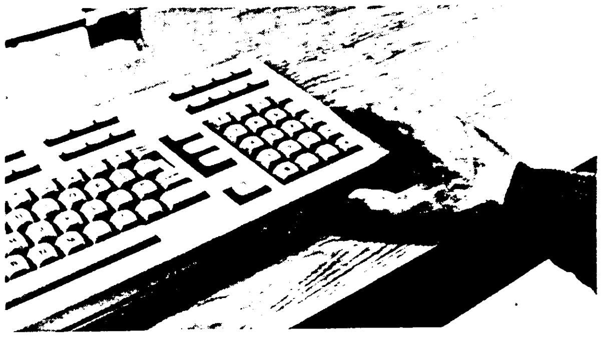
1 UL and CSA approved for 100/120 Vac operation.

2 UL and CSA approved for 220/240 Vac operation.

5. Initial Power Up

Now that you've checked the line-voltage switches, checked the fuses, connected the power cord and know which language system you have (soft-load or built-in), you are ready to switch the computer on.

With a Built-in Language System
If your computer has a built-in language system, first remove any disc in the drive and then press the power switch in. The computer display takes about 10 seconds to warm up. In the mean time, the computer tests its memory. Then a "READY" message is displayed. The computer is now ready for your use.



```
                                    HPL READY
```

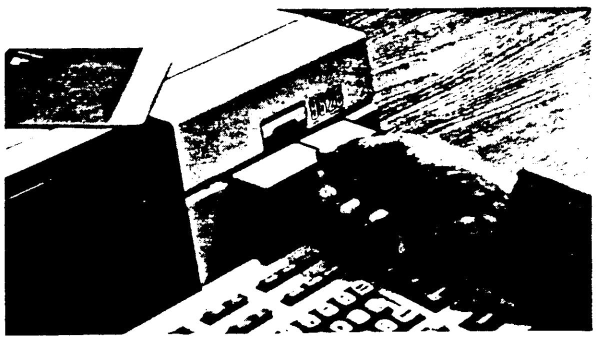
If more than one language system is built-in, the computer allows you to select one. For example:

```
WHICH SYSTEM?
BH
```

In this example the computer found two built-in systems, BASIC (B) and HPL (H). The computer will wait about 10 seconds for you to select the language system by pressing the appropriate key. (If you press the wrong key, the computer will just beep and continue waiting.) To select the HPL system, press the H key.

If an appropriate key isn't pressed in time, the language system listed first (BASIC in our example) is automatically loaded.

With a Soft-loaded Language System:
If your computer has a soft-loaded operating system, open the disc drive door and insert the Language System disc. Be sure the disc is inserted with its label up and facing you, as shown below. Then close the door and press the power switch in.



The computer automatically looks for a SYSTM-type file on disc at power-up. If one is found, it's loaded into memory and then a "READY" message is displayed. For example:

```
(RAM) HPL READY
```

When the Built-in System Does Not Load:

If the computer does not display the READY message after about 10 seconds, or if a system error is displayed, switch the computer off, wait a few seconds and switch it on again. If the READY message still doesn't appear, call HP for service. See the list of service locations at the back of the manual.

When the Soft-load System Does Not Load:

If the computer does not display the READY message after about 15 seconds, or if "UNABLE TO FIND SYSTEM" is displayed, try to re-load the system. First remove the disc, check to be sure it is a system disc, and re-insert it in the disc drive. Then close the drive door and press (SHIFT)-(PAUSE). If the computer still doesn't load its system, either the system disc is defective or the computer requires service. Call HP for service. See the list of service locations at the back of the manual.

After powering up the computer for the first time, you should verify its operation by running the computer tests explained in chapter 5. Once computer operation is verified, switch it off and install any additional accessories supplied. See the next sections.

FEDERAL COMMUNICATIONS COMMISSION
RADIO FREQUENCY INTERFERENCE
STATEMENT (U.S.A. ONLY)

The Federal Communications Commission (in Subpart J of Part 15, Docket 20780) has specified that the following notice be brought to the attention of the users of this product.

Warning: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

6. Install Additional Read/write Memory

CAUTION

THE COMPUTER MUST BE SWITCHED OFF BEFORE ANY ACCESSORY BOARDS ARE REMOVED OR PLUGGED IN. PLUGGING OR UNPLUGGING BOARDS WITH THE POWER APPLIED WILL DAMAGE THE BOARD OR THE COMPUTER.

The computer's program and data-storage memory can be expanded by installing additional read/write memory boards. Each memory board can be plugged into any available accessory slot at the back of the computer.

Before installing a memory board, note the amount of available read/write memory by first switching the computer on:

```
123456 Available bytes
```

Now switch the computer off and remove the memory board from its anti-static plastic package.

CAUTION

STATIC DISCHARGE CAN DESTROY COMPONENTS ON A MEMORY BOARD. HANDLE THE BOARD BY USING ITS ANTI-STATIC ENVELOPE. DO NOT TOUCH THE ELECTRICAL TRACES OR SET THE BOARD ON ANY STATICALLY CHARGED SURFACE (E.G., A CLOTH).

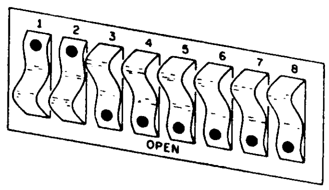
Each 64 kbyte memory board must be set to a consecutive starting address, beginning with the lowest address of any memory board(s) currently installed. Each board has a mini-switch labelled SW-1 for this purpose. The following table and drawing show how to set the switch.

The standard computer is supplied with 64 kbytes of built-in memory. The first additional memory board should be assigned the starting hexadecimal address FF. Each additional 64 kbyte board must be set to the next-lowest hexadecimal address: FE, FD, etc. Memory boards must not be set to the same address.

64 kbyte Memory Board Starting Addresses

Memory Board	Starting Address	Switch Setting* 12345678
1st memory board	FF	11111111
2nd additional board	FE	01111111
3rd additional board	FD	10111111
4th additional board	FC	00111111
5th additional board	FB	11011111
6th additional board	FA	01011111
7th additional board	F9	10011111
8th additional board	F8	00011111

* "1" indicates switch is open: "0" indicates switch is closed.



98254A Memory Board Starting Address Switch
(shown set to address FC)

After installing each memory board, switch the computer on and verify the new amount of available memory.

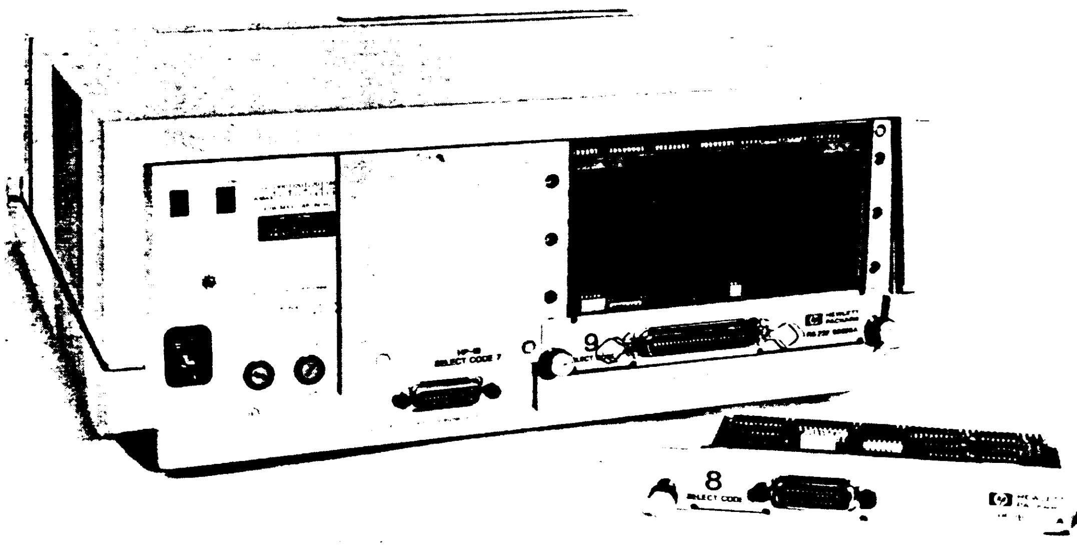
If the available memory does not increase with each added memory board, switch the computer off and verify that the board is properly seated in the accessory slot. Also check the setting of the starting address switch. If it's not set to the highest-available address, the computer cannot address the board.

If the computer still doesn't indicate an increase in available memory, or the computer does not power-up correctly when an additional memory board is installed, switch the computer off, remove the board and replace it in its anti-static envelope. Then call HP for details on replacing the board. Office locations are listed at the back of this manual.

7. Install Interface Cards

Now that your computer is installed and configured with any additional read/write memory, you can install interface cards and connect peripheral components to the computer. Be sure to switch the computer off before plugging in or removing any cards or memory boards.

The computer has eight accessory slots. Each can hold a memory board, while every other slot is designed to accept an interface card. This allow installing up to four interface cards and at least four memory boards. The built-in HPL language system is contained on one board already installed. This board must not be removed.



Installing Interface Cards

A manual provided with each interface card explains how to configur the card for your system. Follow those instructions carefully to ensure a smooth installation:

Be sure each interface card is set to a unique address or select code. A switch on each card sets its select code. These codes are already reserved by the computer:

HPL Internal Select Codes

0	Keyboard and CRT Display Line
7	HP-IB interface (built-in)
16	CRT Print Area

As shown, select codes 0,7,and 16 are reserved for the computer's internal use. That leaves select codes 1 thru 6 and 8 thru 15 for external interface cards.

Maintaining Your Computer

Cleaning the Computer

The computer should be cleaned with a soft cloth lightly dampened either in clean water or a mild detergent. Don't allow water to get in the computer case. Don't use any abrasive cleaners.

Clean the Disc Drive Heads

The disc drive's read/write heads should be cleaned periodically to ensure trouble-free operation. A head-cleaning kit is available from HP for use with your computer. Order HP accessory number 92193A. HP does not recommend use of other head-cleaning discs or equipment.

CAUTION

DO NOT ATTEMPT TO CLEAN THE DISC READ/WRITE HEADS MANUALLY OR WITH MATERIALS OTHER THAN THOSE SUPPLIED BY HP. OTHERWISE HEAD DAMAGE OR MIS-ALIGNMENT COULD OCCUR.

To clean the disc read-write heads:

1. Switch the computer off.
2. Insert the 9826 System Test Disc in the disc drive and close the drive door.
3. Switch the computer on. The system text program is automatically loaded.
4. When the initial System Test menu is displayed, press either CLEAN DISC softkey, **k1** or **k6**. Then follow the displayed instructions.

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 Chapter 4.

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 one.

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.

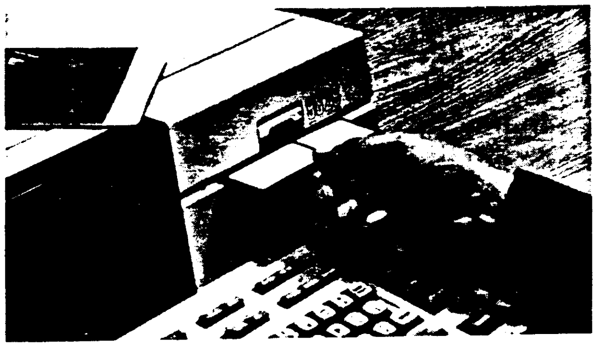
Disc Handling Precautions

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.

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.

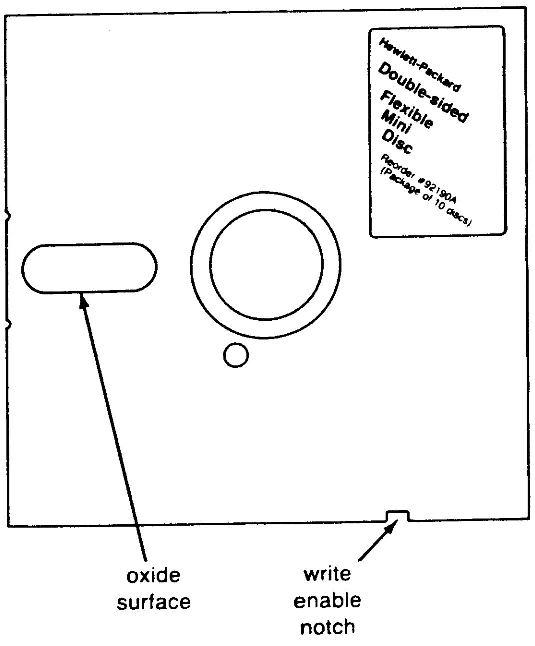
CAUTION

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.



Data File Compatibility

The built-in disc drive initializes discs in a standard HP mass storage format called LIF, ensuring that files originated by one HP 9826 will be compatible with other 9826 computers and Language Systems. The standard format also allows the computer to identify and read certain files from other HP computers and terminals. For example, type ASCII files (containing data and programs) originated by an HP 2642 Terminal can be read by the HP 9826. Type ASCII files originated by the HP 9826 Computer can be read by the HP 2642A Terminal. Other type files originated by the computer may be identified but may not be read by the terminal.

For details on mass storage compatibility with other HP equipment, refer to the Disc Programming Technical Appendix or contact your HP sales office. Locations are listed at the back of the manual.

Using Discs

The first step to take to use a new disc (NOT your HPL Language Disc!) is to insert it as described previously, and initialize it. You must initialize a disc whether you are using tape file commands or disc programming commands. To initialize a disc in the internal disc drive, execute

```
init ":I,0"
```

The initialization process takes about two and a half minutes. Once initialized, the disc can be used to save and get programs and data. It can be also used to store and load programs just as if it were a tape cartridge.

To use the disc with disc statements, you should first become familiar with disc structure and usage. The basics of disc programming are discussed in the 9825 Disc Programming Manual included with your 9826. Additions and extensions to 9825 disc programming that are provided by the 9826 are described in the Disc Programming section of Chapter 4 of this manual. Some of the technical aspects of 9826 disc programming are discussed in the Disc Programming Technical Appendix to this manual.

To simply save a program you have typed in, the procedure is simple. With the initialized disc in the drive, your program in memory, and assuming you wish to call your program "PROGRAM1"

press:	`save`	(this is special function key **k0**)
type:	`PROGRAM1`	(this displays `save "PROGRAM1"`)
press:	EXECUTE	(the program is saved to disc)

To retrieve that program off the disc, a similar procedure is followed.

press:	`get`	(this is a special function key **k1**)
type:	`PROGRAM1`	(this displays `get "PROGRAM1"`)
press:	EXECUTE	(the program is loaded into memory)

To determine what programs are on the disc, you merely press one key.

press:	`*cat`	(this is special function key **k4**)

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 Chapter 4 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 in the Utilities Pack to copy the files from one disc to another. 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. For details on running "ibackup"and the many other HPL utilities, refer to the HPL Utilities Manual.

The "cbackup" program can be used to copy all files from a disc originated by an HP 9826 Computer. Although the program runs on the HPL language system, it copies disc files containing BASIC, HPL and other programs originated on an HP 9826. "cbackup" will also copy files from mini discs recorded on other HP equipment which conforms to LIF (Logic Interchange Format) standards. Your HP sales office can furnish a list of LIF-compatible equipment.

Follow these steps:

1. Switch the computer on and load the HPL language system (see Chapter 1).

   If the HPL system is already loaded, execute this command to clear the computer memory: `erase a` **EXECUTE**

2. Insert the 9826 HPL Utilities Disc in the drive and close the drive door.
3. Load and run the "cbackup" program: `get "cbackup"` **EXECUTE** **RUN**
4. Follow the displayed instructions.

Note

Be sure to press the **CONTINUE** key when instructed, not the **RUN** key. If **RUN** is pressed after the program has started, the program must be stopped and restarted using the Utilities Disc. Return to step 1 above.

The ''cbackup'' program assumes the backup disc is an initialized disc containing either no files or no wanted files. If files are found, a catalog of the backup disc is displayed for you. To automatically purge the files and begin the disc copy, press **CONTINUE**. If the backup disc is not initialized, the program automatically initializes the disc before copying files from the master disc. The initialization routine takes about three minutes.

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 catalogs the files and determines how many disc exchanges will be needed to copy all files to the backup disc.

A catalog of the backup disc is displayed after the backup is complete. To backup another disc, using either the same master or another master disc, enter Y and press **CONTINUE**. To exit the program, enter N.

Chapter 2
Getting Started

Introduction

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.

Daily Power-up

After the computer has been installed as covered in Chapter 1, daily power-up is simply a matter of either switching the power on (if the language system is built-in) or inserting the System Disc and switching power on (if your system is soft-loaded). In either case, the computer automatically tests its memory and then loads its language system.

When "HPL READY" is displayed, for example, the computer is ready to accept keyboard commands. Now you can load and run programs or develop your own HPL language programs. If you're running pre-recorded programs, you may be able to let the computer automatically load and run a program by using the Autostart feature.

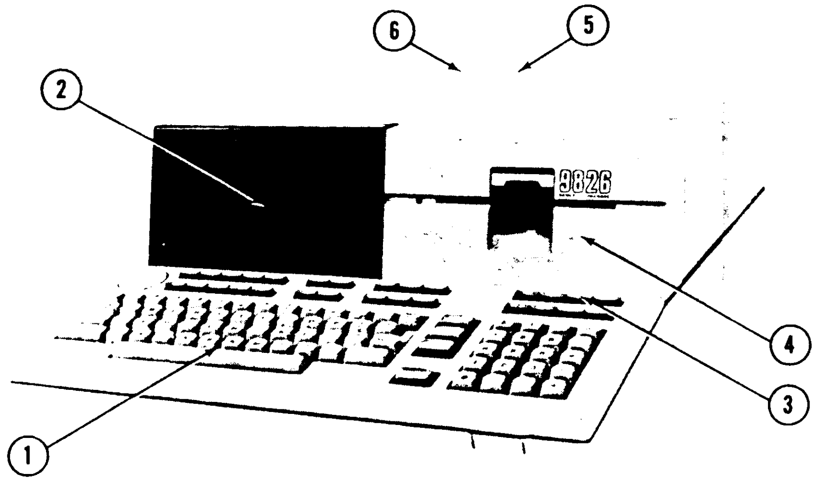
Program Autostart

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 T0F000, 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 (700-series options), the autostart disc must have both the SYSTM-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.

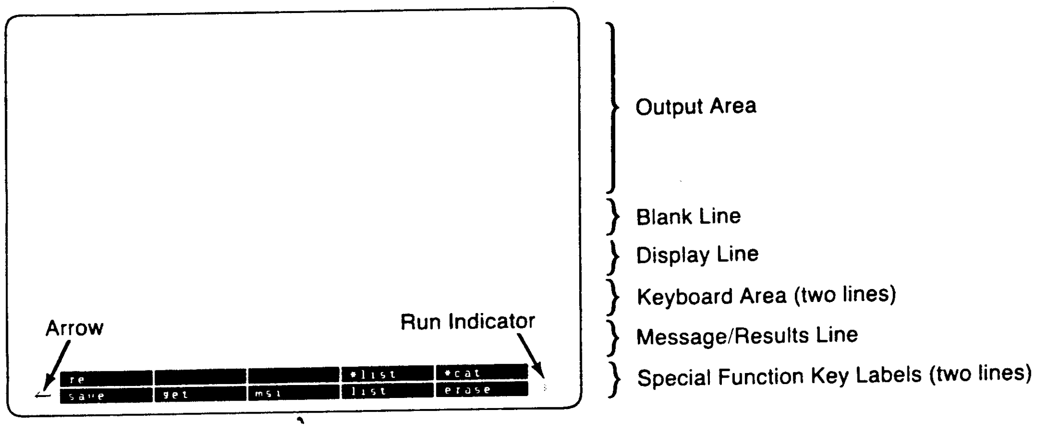
(4) Mini Disc Mass Storage. The built-in disc drive uses standard 130mm (5-1/4 inch) discs for storing data, programs and other computer information. Each disc can hold about 1/4 million bytes (characters) of information.

(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 standard HP-IB connector, the computer has eight interfacing connectors on its backplane. Each connector can accept a memory board (for additional user memory), a language sysem board (for the operating system and add-on language ROMs) or an interface card. As explained in chapter 1, however, interface cards cannot be installed in adjacent connectors.

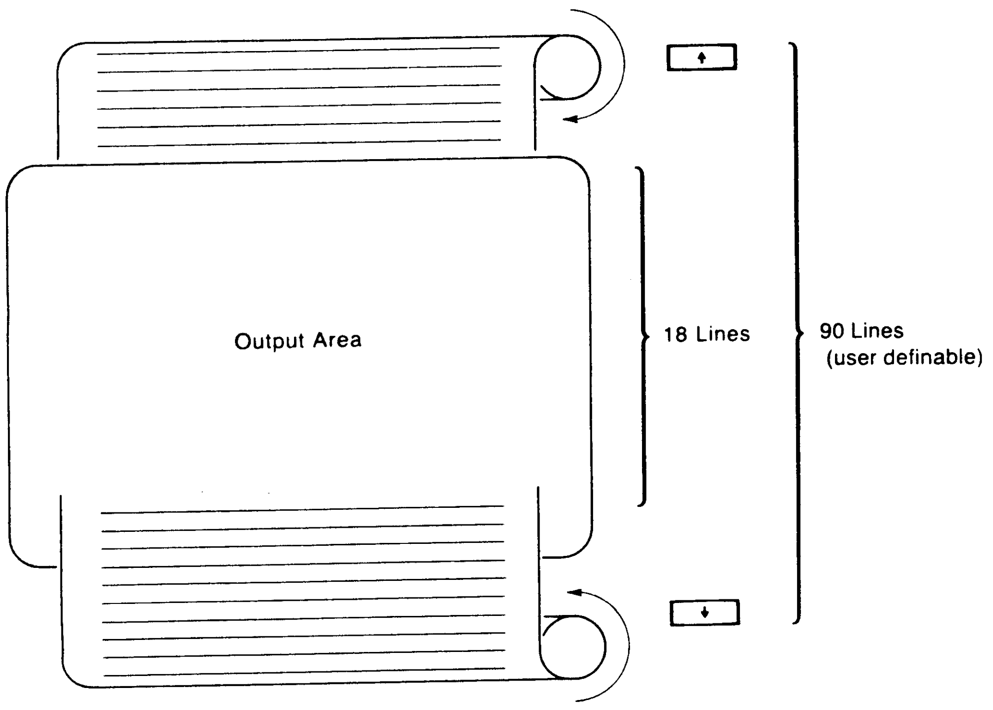
Display Organization

The built-in display (CRT) has a 50-character wide by 25-line work area. The HPL language system partitions the display into five areas:



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.

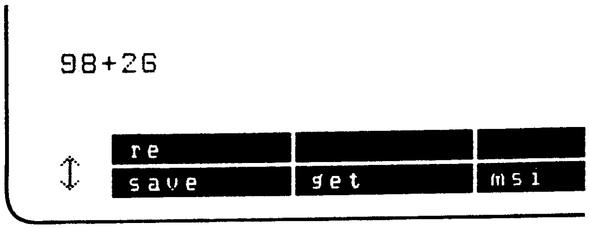


The Output Display Area and Buffer

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

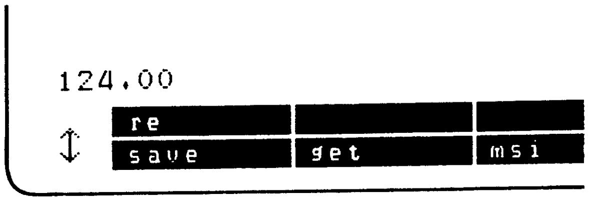
The **keyboard area** is where you enter responses to program prompts or type in commands. Press CLR LN to clear the keyboard area. Now do a simple arithmetic problem:

Type: 98+26
Press: EXECUTE



```
98+26
```

The operation is first entered in 124.00 the keyboard area. When EXECUTEd, the result relaces the operation in the message/results line.



```
124.00
```

Now try repeating the operation. Press RECALL and EXECUTE.

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.

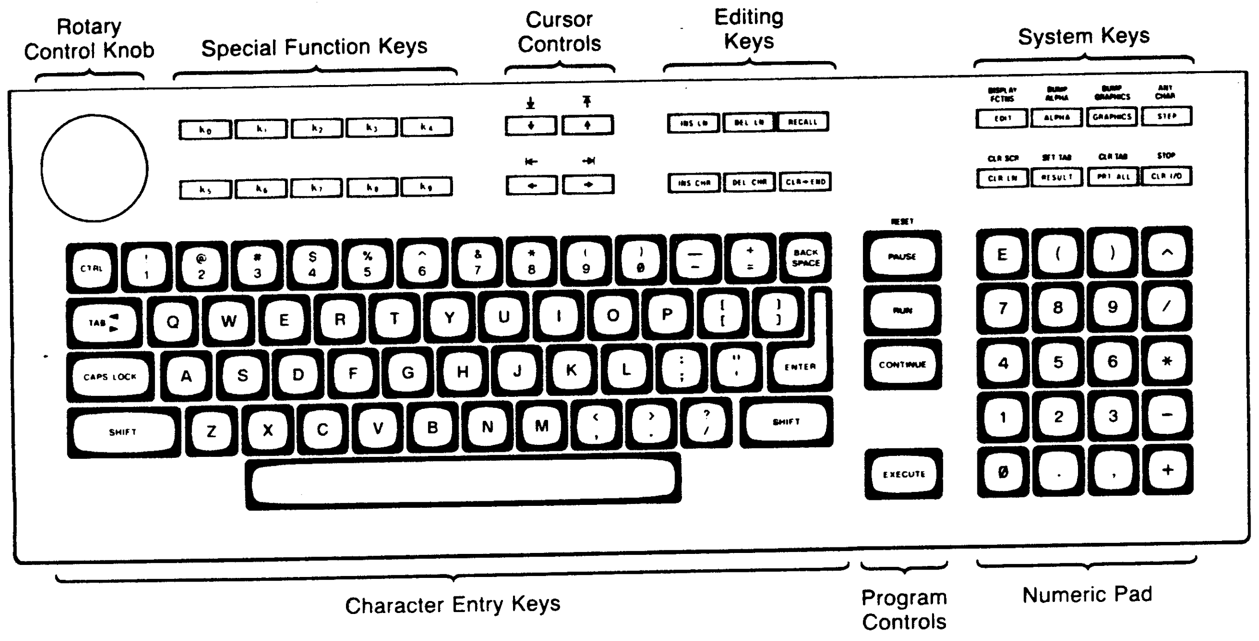
The **arrow** in the left-hand corner indicates the currently set direction of the cursor wheel, either up-and-down or left-and-right. Use the wheel with the cursor-control keys to rapidly position the display cursor.

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 graphics mode can be automatically set when a program outputs graphic data on the display. You can switch back and forth, between graphics and alpha modes, by using the **GRAPHICS** and **ALPHA** keys.

Keyboard Operations

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



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.

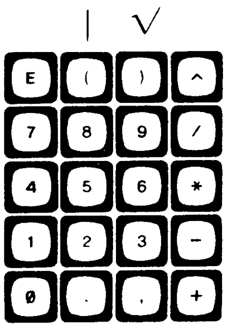
The **CAPS LOCK** key sets the unshifted keyboard to either upper case or lower case (for normal typewriter operation). The computer displays the mode set when you press the key.

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

The **TAB** key is the HPL assignment operator key, "→".

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

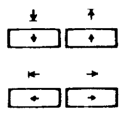
Numeric Pad



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

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

Cursor Controls

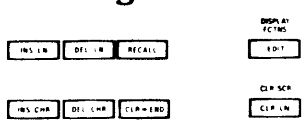


The cursor-control keys move the display cursor one space at a time. Press and ← or → to move the cursor to the end of the line. The ↑ and ↓ keys allow you to scroll information in the displayed output area up and down.

The 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.

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

DEL LN Deletes the line containing the cursor (edit mode only).

RECALL Recalls the last line entered or executed.

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

DEL CHR Deletes the character under the cursor.

CLR→END

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.

CLR SCR
CLR LN

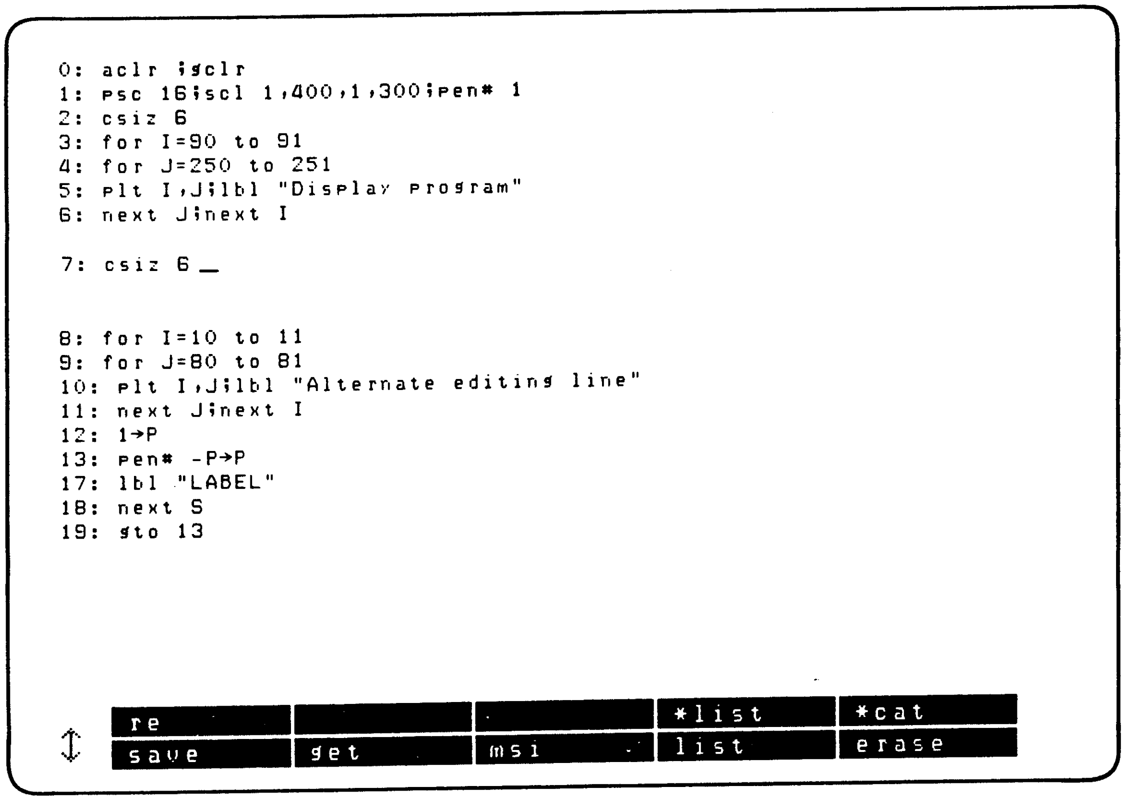
Clears the entire line. Press **shift-CLR LN** to clear the entire display screen.

EDIT

Enters the `edit` command, allowing the programmer to use an editing mode for entering and editing program lines. (This is the 9825 FETCH key).

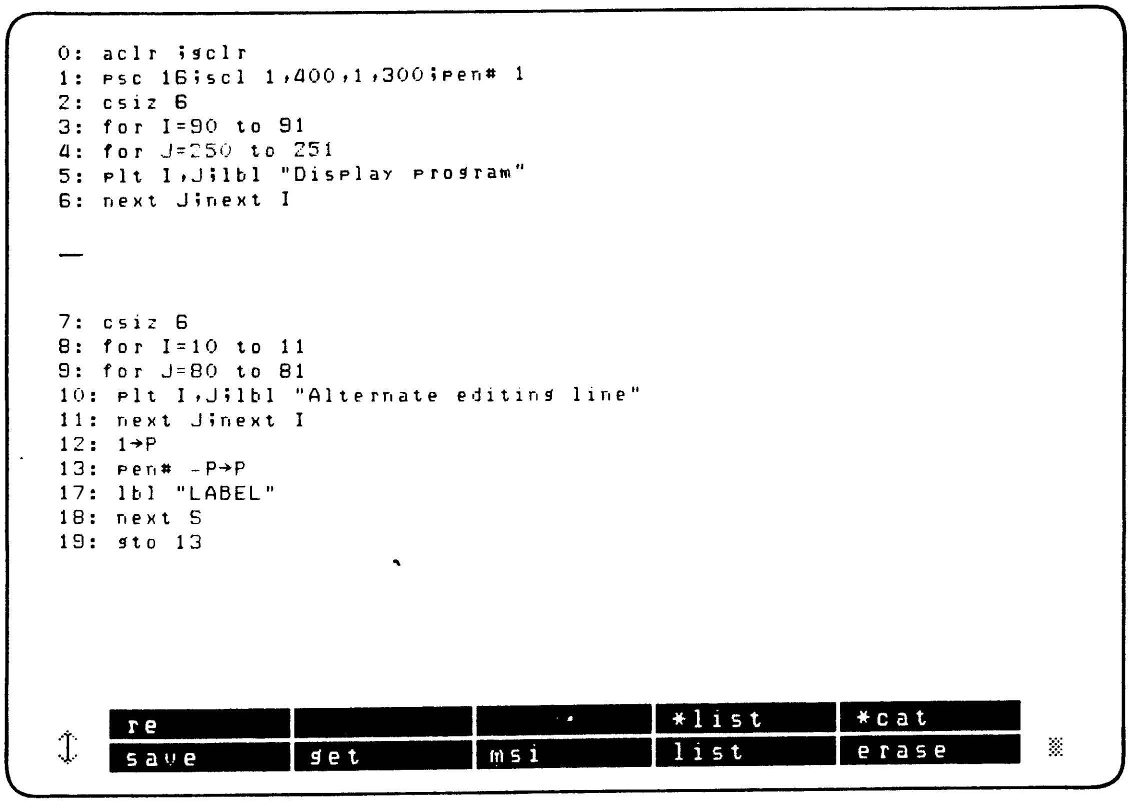
When a program is in computer memory, the program editing mode displays the program lines and waits for the programmer to scroll through, line by line, using the cursor wheel and cursor-control keys. Program changes are made by editing each line and pressing **ENTER**.

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



```
0: aclr ;gclr
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: 1→P
13: pen# -P→P
17: lbl "LABEL"
18: next S
19: gto 13
```

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

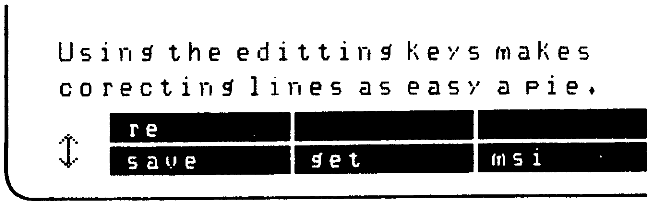


```
0: aclr ;gclr
1: psc 16;scl 1,400,1,300;pen# 1
2: csiz 6
3: for I=90 to 91
4: for J=250 to 251
5: plt I,J;lbl "Display program"
6: next J;next I

—


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: 1→P
13: pen# -P→P
17: lbl "LABEL"
18: next S
19: gto 13
```

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).

Here's our first try:



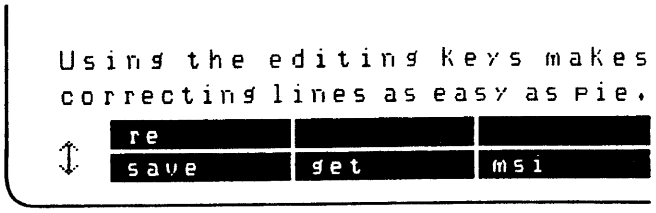
```
Using the editting keys makes
corecting lines as easy a pie.
```

Original Line

Not bad, we only misspelled one word in the first line.

Move the cursor to the extra "t" in "editting" and press (DEL CHR) once.

The second line needs some characters inserted. First move the cursor to the "r" in "corecting", press (INS CHR) and insert the missing "r". While the insert mode is still set, insert an "s" after "a" to spell "as". Then press (INS CHR) again to cancel the insert mode.



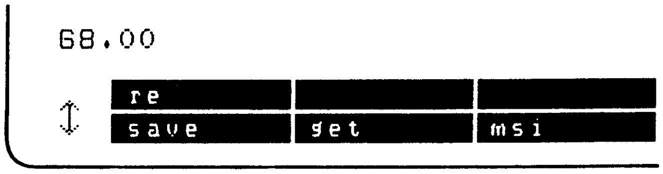
```
Using the editing keys makes
correcting lines as easy as pie.
```

Corrected Line

SET TAB
shift- RESULT

Enters res, for the result function. When executed, returns the result of the last expression executed. For example, press (EXECUTE). Then enter:

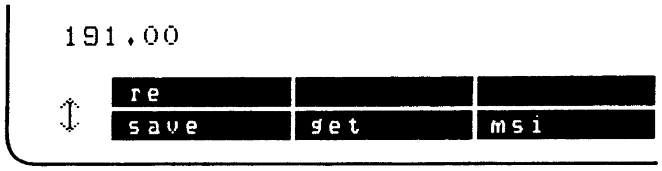
23 + 45 (EXECUTE)



```
68.00

re
save      get      msi
```

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

(RESULT) + 123 (EXECUTE)



```
191.00

re
save      get      msi
```

PRT ALL

Sets 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 chapter 4.

STOP
shift- CLR I/O

Halts program execution after the current line is executed, as is the case for the **PAUSE** key. To re-start the program, press **RUN**.

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 ten special function keys (without displayed labels) can be defined at the same time and accessed by the **SHIFT** key. These shifted keys are identified as **k10** through **k19**.

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) within quotes, followed by a colon, followed by the remainder of the key definition. Some examples will help to clarify this:

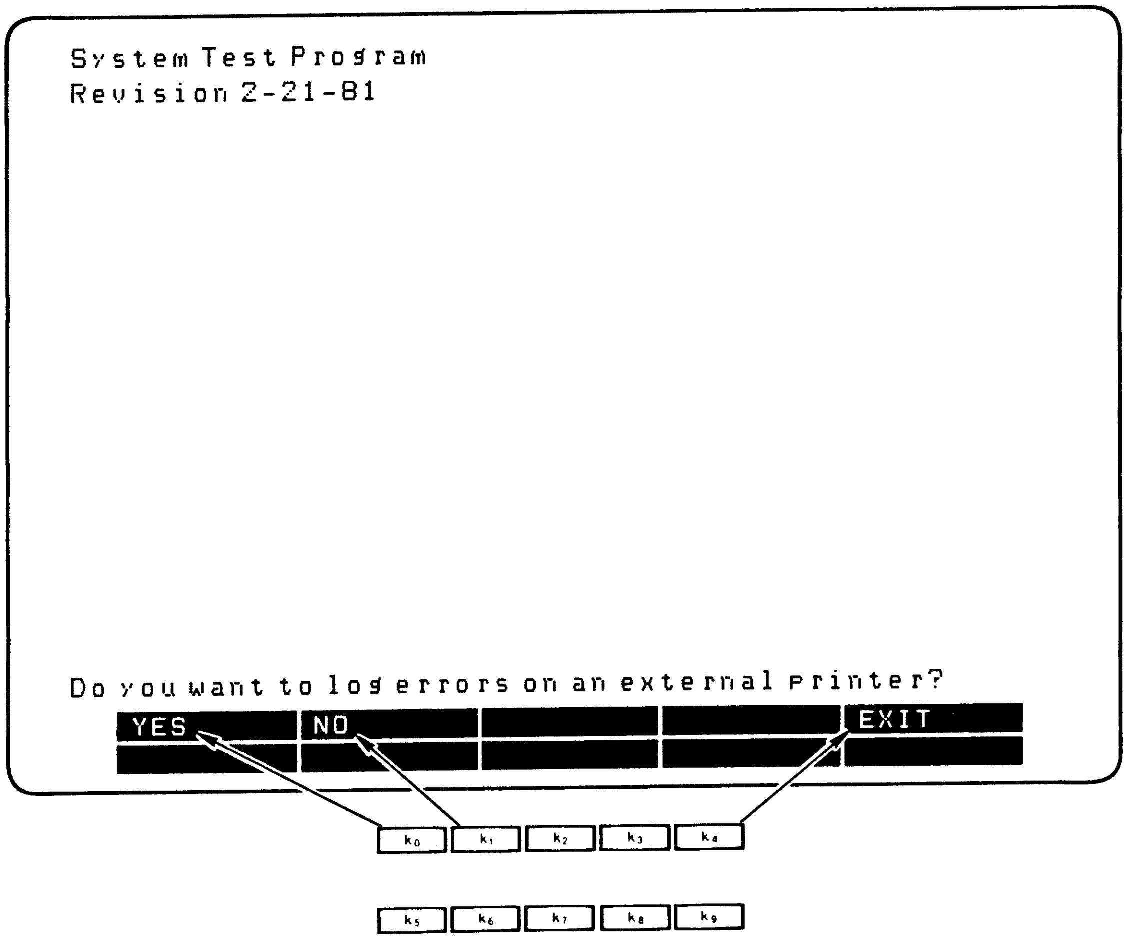
Press: (EDIT) (k1)
Type: "Label":definition
Press: (ENTER)

Notice that Label is now the "soft" key label assiciated with **k1**.

Other labelled special function key examples:

```
"Recover" :*cont "Restart"
"Abort" :*sfg 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:



```
System Test Program
Revision 2-21-81








Do you want to log errors on an external printer?
```

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: `*prt"  ",`	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.

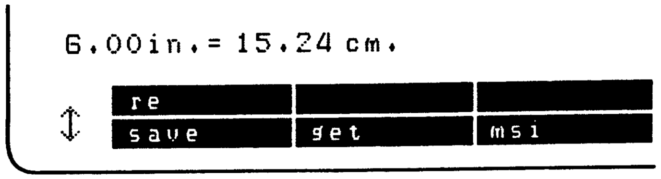
Keys with Multiple Statements
By separating statements with semicolons, several statements can be stored under one special function key. As an example, suppose you want to convert inches to centimetres. The following line is stored under special function key k2.

Press: EDIT k2

Type: `* R;dsp R,"in.=",2.54R,"cm."`

Press: ENTER

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



```
6.00in.= 15.24 cm.
```

Extended Control Special Function Keys
If a special function key definition is preceded by an EOL character (decimal 127: use the ANY CHAR 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, 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
Press: SHIFT STEP	Any char of 127 is the EOL char
Type: 127	
Press:: CTRL SHIFT →	Home right
Press: CTRL CLR→END	Take out of insert-character mode.
Press: CTRL SHIFT ←	Home left key
Press: CTRL INS CHR	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 "`Comment`".

Default Special Function Keys
There are several special function keys predefined by the 9826A HPL Language system. 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 the **EDIT** key 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"`

`*list` — k3 This is an immediate execute key that list 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** is pressed, the program in memory is saved to the current system disc drive (set by msi or drive).

`get` — k6 This is a typing-aid key that displays `get""` 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** is pressed, the specified program is loaded into memory (if one exists on the disc).

`msi`	k7 This is a typing-aid key that displays `msi":"` 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 **EXECUTE** is pressed, the disc specified becomes the system mass storage device.
`list`	k8 This is a typing-aid key that displays `list` on the keyboard line. Its use is essentially the same as the **LIST** key was on the 9825A.
`erase`	k9 This is a typing-aid key that displays `erase` on the keyboard line. Its use is essentially the same as the **ERASE** key was on the 9825A.

Program Control Keys

The **PAUSE**, **RUN**, **CONTINUE** and **CLR I/O** keys allow you to control execution of the program stored in the computer's memory.

RUN	Starts program execution from the beginning.
PAUSE	Pauses program execution after the current line, returning computer control to the keyboard.
CONTINUE	Resumes program execution from where it was paused, or enters data for an active ent or enp statement.
shift- RESET PAUSE	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
shift- STOP CLR I/O	Stops program execution after the current line, as by pressing the **PAUSE** key.

Arithmetic Operations

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

√	Square root
ˆ	Exponeniation
no operator	Implied multiplication
*/	Multiplication and division
+ −	Addition and subtraction

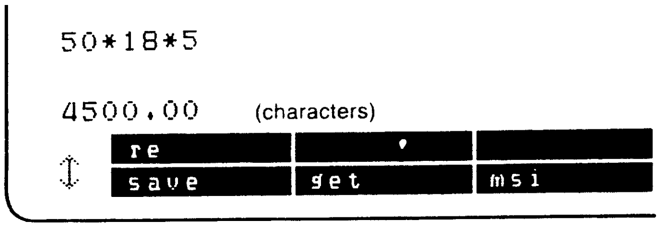
If you prefer, you can also use the same characters found within the typewriter keyboard.

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

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

Enter: 50*18*5

Press: (EXECUTE)

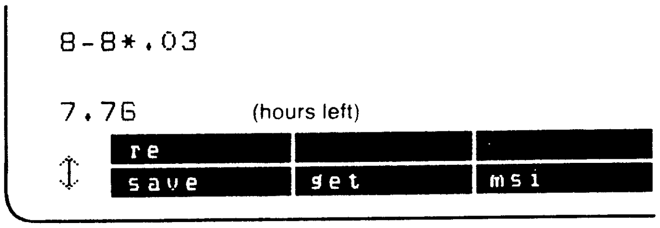


```
50*18*5

4500.00     (characters)
```

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

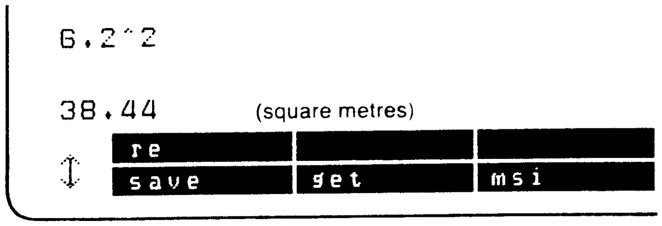
Press: (EXECUTE)



```
8-8*.03

7.76        (hours left)
```

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 2):

Enter: 6.2ˆ2 (EXECUTE)



```
6.2ˆ2

38.44       (square metres)
```

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
↕  save        get         msi
```

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

DISPLAY FCTNS · DUMP ALPHA · DUMP GRAPHICS · ANY CHAR
EDIT · ALPHA · GRAPHICS · STEP
CLR SCR · SET TAB · CLR TAB · STOP
CLR LN · RESULT · PRT ALL · CLR I/O

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.

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

DISPLAY FCTNS shift-EDIT — 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.

ALPHA
GRAPHICS — 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.

DUMP ALPHA shift-ALPHA — Enters the `adump` command. Pressing **EXECUTE** causes the complete alpha display to be output to the current prtsc device. See chapter 4 for details.

DUMP GRAPHICS shift-GRAPHICS — Enters the `gdump` command. Pressing **EXECUTE** causes the graphics display to be output to the current prtsc device. See chapter 4 for details.

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

ANY CHAR shift-STEP — Shows each available display character. First press **shift-STEP**. 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 at the back of this manual.

CLR LN — Clears the display input line and the message/result line.

CLR SCR shift-CLR LN — Clears the entire alpha display, including the input line and the scrolling buffer, and turns on key labels.

Chapter 3
Program Transfer

Introduction

This section ddresses a major concern of 9825 owners and programmers, namely, how to get 9825 programs and data into the 9826. 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 9826 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 internal to the 9826 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 9826 (and vice-versa, if desired). In addition, disc file transfers offer the only method of transferring key files from the 9825 to the 9826. Once the disc drive is installed on the 9826, the same commands that were used on the 9825 can be used on the 9826. For example, to load a 9825 program "Prog1" into the 9826 (from a 9885 disc, unit number 0 at select code 8), the following statements could be used:

```
drive 0,8
get "Prog1"
```

The above sequence can be shortened by using the new mass storage unit specifier (msus) available on the 9826:

```
get "Prog1:F8,0"
```

Disc data files can be accessed from the 9826 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. On the 9826, to access an external disc drive (9885 or 9895), a drive or msi statement must be executed explicitly stating the unit number and select code of the disc drive. (The default drive on the 9826 is the internal minifloppy.)

There are additional disc programing capabilities offered by the 9826: you may wish to refer to Chapter 4 of this manual for specific details of the 9826 disc programming enhancements.

Interface Transfer

Interface transfers offer a means of getting programs and data from a tape-based 9825 system into the 9826. 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 9826 that reads in the program and stores it into 9826 memory. You then save the program on the internal minifloppy disc drive of the 9826.

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 9826. 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 9826 execute `wtc 7,30` to change the bus address of the 9826 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;ldf8`).
the 9826, RUN the following program:

```
0: "Loader":dim A$[85];nal→N;721→S
1: "input":red S,A$;if len(A$)<=2;gto "input"
2: if A$[1,1]="*";gto "done"
3: "store":on err "error";store A$,nal
4: gto "input"
5: "error":"%"&A$→A$;gto "store"
6: "done":red S,A$,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 8,N,X'. Then delete the"
10: prt "old program and re-run this program for the"
11: prt "next program to be transferred.";fxd 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 to the 9826.

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 9826 internal disc. The string files will be saved as tape files on the 9826 disc 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
```

9826 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 9826; the 9826 is reading in the data from the 9825 and saving it on the disc.

Chapter 4
9826 HPL Programming

Introduction

This section summarizes the enhancements and differences between the 9826 HPL operating system and the 9825. Organization is presented by 9825 ROM function so that you can easily understand new material in relationship to the 9825.

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
```

```
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)

```
pbeep [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).

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

The Clear Alpha Statement

```
aclr[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). Each page is 18 lines of 50 characters and uses 900 bytes of read-write 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!

```
aclr int(avm/900)-2
```

The graphics screen is not affected by aclr, key labels are caused to be displayed, the Run screen format of the CRT is selected (as opposed to the Edit screen), and Display Functions is turned off when aclr is executed.

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).

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. The row parameter should be within the range of 0 through 17. Specifying row greater than 17 causes the row parameter to be truncated to 17 (you cannot tab below the bottom of the screen).

The Read CRT Statements

`red 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. Power-up default for `prtsc` is to the 9826 CRT, width = 50. (No, executing `prtsc 16,80` will not cause 80 character lines to be displayed on the 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 $\sqrt{\ }$ 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.

Tape Cartridge Operations (O&P chapter 5)

A program utilizing tape cartridge mass storage operations will in general be able to run unmodified on a 9826 HPL 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 9826 HPL: 9885, 9895, 8290x, and the internal minifloppy drive. 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:

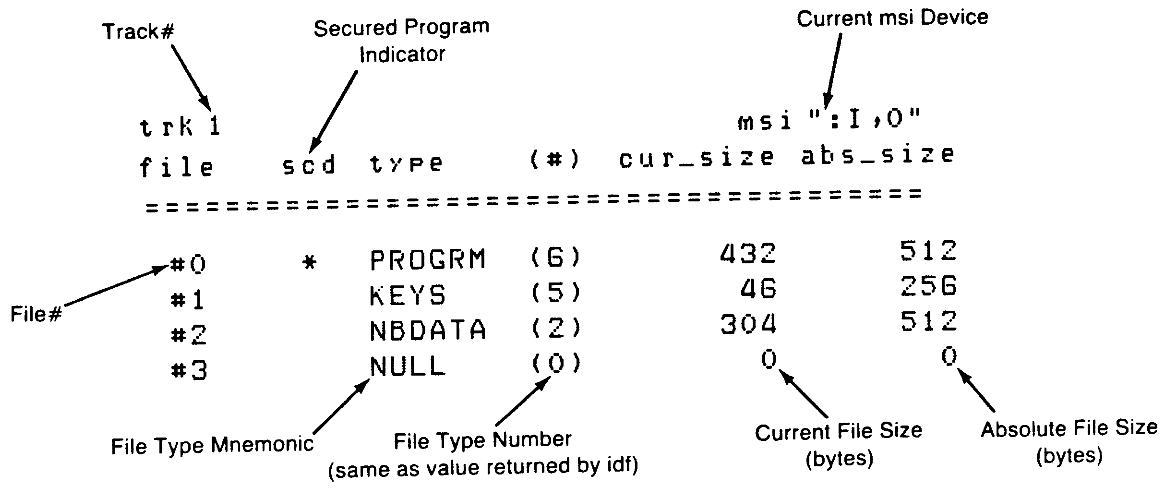
  ```
  TxFyyy
  ```

  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 `init` statement) before the tape files can be marked.
- The format for a tlist printout has been modified to take advantage of the new 50 character line width of the 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.



```
trk 1                               msi ":I,0"
file   scd  type      (#)  cur_size abs_size
==========================================

  #0    *   PROGRM    (6)      432       512
  #1        KEYS      (5)       46       256
  #2        NBDATA    (2)      304       512
  #3        NULL      (0)        0         0
```

Track#
Secured Program Indicator
Current msi Device
File#
File Type Mnemonic
File Type Number (same as value returned by idf)
Current File Size (bytes)
Absolute File Size (bytes)

- 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 9826, 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 9826, because all files on the 9826 are rounded up to a multiple of 256 bytes.
- A `tlist` 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 `cat` (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 on the 9826. (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

`rcb` 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 ommitted file 0 is assumed.

String Variables

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

`val (`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)

str `(`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: `val("FF",16)`$\longrightarrow$`A` A = 255
`str(10,2)`$\longrightarrow$`A$` A$ = "1010"

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

The general I/O `red` statement now allows substrings as data destination variables.

Example: `red3,B$[15,250]`

Systems Programming

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

Additional capabilities for program emulation of intelligent terminals are provided by the 9826 HPL Language system. 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]]
      .
      .
      .
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 Knob Function

```
knob
```

The `knob` function returns the accumulated RPG, 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 clockwise rotation. Positive values for `knob` indicate counter-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 9826 hardware key codes, but the `asc` function can be used to obtain the ASCII codes for keys. These ASCII keycodes are identical for both computers. For example, the hardware keycode for "5" on the 9825 numeric keypad is 83 while on the 9826 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 9826 keycodes. Refer to the Keycode Conversion Table for the corresponding values of 9825 and 9826 keys, or you can use the keycode conversion utility function "9825 key" to achieve program compatibility.

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

The Remote Keyboard Statement

`rkbd` select code [,type]

The 9826 `rkbd` statement operates in an identical fashion to the 9825 `rkbd` statement except that the type 0 remote keyboard now requires 9826 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 appendix to this manual for the location of the remote keyboard jumper.

The Press Keyboard Statement

`pkbd`[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 the **EXECUTE** key, and has the effect of executing previous characers in the string. The statement

`pkbd "5*2LF"` results in a display of `10.00`.

The `pkbd` 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 `pkbd` those ASCII keycodes that need to be executed by the system. For example, to emulate a full live keyboard with a running program:

```
0: on key "svc"                              Set up service routine
1: gto +0                                    Loop
2: "svc":if not (asc key→K);kret             Test for and exit if empty buffer
3: if K#1;pkbd char(K);kret                  Else "push" key and exit unless
4: end                                       PAUSE key was pressed: stop.
*3629
```

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 default settings
`sfk 8, "wrt 705,"`	defines key #8 to be the string "wrt 705,"
`sfk3,"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 160 (k19), 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**. With this example, an ASCII "$^{S}{}_{H}$" character is displayed, just as if there were no program running.

```
0: on key "svc"
1: gto 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 "BNU"` (null)	Boot BASIC ROM language system (if BASIC soft system is not present)
`sysboot "SYSTEM_B"`	Boot BASIC soft system

Matrix Programming

Array Output (Matrix p.8)

```
aprt 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, `aprt` 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 on the 9826 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 9826 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 9826 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
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
J	9895	LIF[1]	1-15	0-7	707	0-3

1 LIF. Hewlett-Packard Logical Interchange Format: File type ASCII is compatible with 9826 BASIC language ASCII files and with 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 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:

`:F8,0`	Specifies 9885 disc, 9825-compatible format, select code 8, unit number 0.
`:I`	Specifies 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:

```
asgn "Data1:H707,3",1
asgn "Data2:I",2
kill "Oldprog:F8,1"
```

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 a 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:

`msi`	Restore power-up default (":I" for internal drive)

`fxd0;":J"→M$; 707→S; 2→U`
`msi M$&str(S)&","&str(U)`
	Set mass storage device to 9895 disc select code 707, unit number two, LIF format.

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

asgn 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:

```
asgn "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.

```
asgn "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 9826 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 1-15 for 5-1/4" minifloppy discs, and 1-29 for 8" 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
9885	2	2	1
9895 no DMA	4	4	3
9895 with DMA	4	3	2

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 minifloppies (5¼ inch) is 14, or 112 entries. The default for 8 inch floppies 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

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 (Disc Programming, p. 1-16)

9826 HPL 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 9826 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 & LIF catalog listing formats are somewhat different. An example of a LIF catalog listing from the 9826 internal minifloppy is shown below:

```
MSI ":I,0"
AVAILABLE RECORDS 1002
FILE NAME   SCD TYPE        #BYTES  #RCRDS  ADDRESS
===================================================

grades          TDATA                   10       16
Class           PROGRM          96       1       26
Count           PROGRM          62       1       27
Name            PROGRM         242       1       28
give            TDATA                   10       29
Keys            KEYS           200       1       39
null_file       NULL             0      10       50
secured_pr  *   PROGRM         242       1       50
ascii_file      ASCII                    1       51
numeric_bd      NBDATA          24       1       52
string_bd       SBDATA         134       1       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 Programming p. 1-18)

killall unit number , select code

or

killall complete msus

The killall statement can be used as a strictly 9825-compatible statement, but only discs of device format types F and H can be accessed when this syntax is used (first syntax).

The killall statement can also be used with the full range of 9826 HPL supported discs and formats by specifying the second syntax, which allows the msus specifier.

The Open Statement (Disc Programming 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 in 9826 HPL 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 9826 HPL 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 9826 HPL and other 9826 programming languages. "ASCII" data files are also used to provide transportability between the 9826 and 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 Programming 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 in 9826 HPL 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 9826 HPL 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.

CAUTION

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 ":F8,0","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 Programming p.4-7)

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

In general, almost all 9826 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 9826 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 a 9826 HPL-created 9825-compatible disc to a LIF disc, but with 9825-compatible files created by other mainframes, it might.

As with all other 9826 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",0,8,"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 9826 binaries are in no way compatible. Binaries written for the 9825 are processor-dependent and as such would have to be completely re-written to run on the 9826. Because the file types for 9825 and 9826 binaries are identical, a user cannot merely look at a catalog listing and determine whether an unknown binary is for a 9825 or for a 9826; both show up as type "B" on 9825-compatible discs and "BINARY" on LIF discs. However, 9826 HPL sets a flag in the directory to enable it to distinguish between binaries, so a 9826 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 9826 binary if directed to do so by the user. This will certainly generate unpredictable operation of the 9825. Don't do it!

Unimplemented Disc Programming Statements

Four disc statements have not been implemented in 9826 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:

```
dump
load
savem
getm
```

Unimplemented 98217 ROM System Cartridge Binary Statements

9826 HPL, 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:

```
dtrk
tinit
ltrk
dirc
boot
vfyb
ptrn tst
ckrd
init      (implemented with different syntax)
killall   (implemented with different syntax)
```

More I/O

Interface Control Operations (I/O p.4-14)

In general, 9826 HPL Language System 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 the 9826 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 9826 HPL emulation of those operations.

Select Code Summary Table

Select Code	9825 Source/Destination	9826 Source/Destination
0 READ	Keyboard (KDP)	Keyboard
WRITE	Single line display	Display line of CRT
1	Reserved for tape cartridge	External
2	External	.
		.
.	.	.
.	.	.
.	.	
7	.	Internal HP-IB
8	.	External
.	.	.
.	.	.
.	.	.
16 READ	Keyboard (KDP)	Read from CRT
WRITE	Internal printer	Write to CRT

NOTE.
- Select code 1 of the 9826 is available for external devices
- Select code 7 of the 9826 is reserved for internal HP-IB
- Read from select code 16 does not return keycodes on the 9826, but instead reads data from the CRT print area
- Read from select code 0 of the 9826 returns 9826 hardware keycodes, completely unlike 9825 hardware keycodes

9826 Select Codes 0 and 16

Operation	Select code 0	Select code 16
rdb	Wait for keypress, then return 9826 keycode	Read a byte from CRT at current print position
red	Not allowed	Formatted read from CRT at current print position
wtb	Write bytes to display line	Write bytes to CRT at current print position
wrt	Formatted write to display line	Formatted write to CRT at current print position
rds	Always returns value of 8	Always returns value of 8
wtc	Select language jumper (wtc 0,5 selects Katakana)	Select language jumper (wtc 16,5 selects Katakana)
wti4	No operation	No operation
wti5	No operation	No operation
wti6	No operation	No operation
wti7	Select language jumper	Select language jumper
rdi(4)	Always returns value of 0	Always returns value of 0
rdi(5)	Always returns value of 8	Always returns value of 8
rdi(6)	Always returns value of 0	Always returns value of 0
rdi(7)	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 the 9826.
- Read and write operations have different effects of 9826 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 9826.
- wti4 and wti6 on the 9825 wrote bytes to the display and printer buffers. There is no equivalent operation on the 9826.

98623A BCD I/O Operations

For BCD input, operation of the 9826 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.

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 `tfr` to or from the interface.

9825/9826 HP-IB Operations

There are some slight differences between register operations with the 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 the 9826.

HP-IB Operations

Operation	9825 Result	9826 Result
`rds(S,A,B,C)`→`D`	HP-IB Extended Status	HP-IB Extended Status
A	0 | 0 | 0 | 0 | 0 | DCL | 0 | Error	0 | 0 | REM LOC | LLO | GET | DCL | IFC | Error
B	1 | 1 | 0 | - HPIB Address -	1 | 1 | 0 | - HPIB Address -
C	EOI | REN | SRQ | ATN | IFC | NDAC | NRFD | DAV	EOI | REN | SRQ | ATN | IFC | NDAC | NRFD | DAV - REN, IFC may be incorrect[1] if System Controller - ATN indicates attempted drive status, not actual state of line if Active Controller - SRQ may be incorrect[1] if Active Controller - EOI may be incorrect[1] if Active Talker - NDAC, NRFD may be incorrect[1] if not Active Talker or Active Listener
D	SRQ | CA | TA | LA | SC | 1 | SPL | EOI	SRQ | CA | TA | LA | SC | 1 | 0 | EOI - 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
`rds(s)`	Return fourth HP-IB status byte (D, above)	Return fourth HPIB status byte (D, above)
`wtc S,X`		If X<=31, then software reset and terminate transfer. If X<31 then HPIB 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
`rdi4`	Immediate read of R4	Always 0
`rdi5`	Immediate read of R5	Always 0
`rdi6`	Immediate read of R6	Always 0
`rdi7`	Immediate read of R7	Always 0
`wti4` or `6`	Immediate write to R4/R6	Wait for FLG then write to R4/R6
`wti5`	Immediate write to R5 (Not recommended)	No operation
`wti7,X`	Immediate write to R7	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
`ios`	Return 98034 STS line.	Identical operation to 9825.
`iof`	Return 98034 FLG line	If the 9826 HP-IB is addressing or talking to the bus then iof=1 when ready for next data item, else iof=0.

1 These lines are driven from the 9826 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 HPIB analyzer or a defective instrument forcing a line to a state different from the attempted drive state of the 98624A HPIB interface card.)

HP-IB Operations (Cont'd)

Operation	9825 Result	9826 Result
`eir`	SRQ / CA / TA / LA / Internal Use Only / Err DCL SDC / EOI	SRQ / CA / TA / LA / Internal Use Only / IFC GET DCL / EOI 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.
`tfr` (buffer type 5)	Not allowed	Byte DMA transfer if DMA card is present.
`tfr701.1,"buf"`	Format ignored	Terminate on EOI.
`tfr701.2,"buf"`	Format ignored	Do not terminate on EOI.
`tfr"buf,"701.1`	Format ignored	Send EOI with last byte.
`tfr"buf,"701.2`	Format ignored	Do not send EOI.
Output to non-existent device	Completes.	Hangs. Card reset is required (wtc7,31 or SHIFT PAUSE)

HP-IB Capability Comparison
(from the 98034A Installation and Service Manual p. 4)

98034A	9826 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
–	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

The 9826 has a built-in real-time clock that is accessible from 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 the year 1900 (1900 is time zero) taking into account leap-years:

```
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
29: if Y>1900;Y-1900→Y
30: if M>2;M-3→M;gto 32
31: M+9→M;Y-1→Y
32: stime (int(1461*Y/4)+int((153*M+2)/5)+D)*24*60*60+H*3600+N*60+S
33: ret
```

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

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:

```
38: "Read-time":
39: data MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC,JAN,FEB
40: rstr "Read-time";for I=1 to 12;read M$[I];next I
41: int(rtime)→S;int(S/(24*60*60))→D}Z;int(D*4/1461)→Y
42: D-int(1461*Y/4)→D;int((D*5-2)/153)→M;D-int((153*M+2)/5)→D
43: if M>9;Y+1→Y
44: S-Z*24*60*60→S
45: S-(int(S/3600)→H)*3600→S
46: S-(int(S/60)→N)*60→S;fxd 0
47: str(H)&":"&str(N)&":"&str(S)&"    "&M$[M+1]&str(D)&","&str(Y+1900)→T$
48: ret
```

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

The On Interrupt Statement

```
oni select code [, label [ , abort byte]]
```

The syntax of the `oni` 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)

The 9826 HPL operating system, like the 9825, has certain critical operations that lock out interrupts for short periods. Unlike the 9825, however, the 9826 does not lock out interrupts for long periods of time during tape drive accesses. (There is no tape drive.) The 9826 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 of the 9826. 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 an on-cycle statement was executed. (The on-cycle statement zeroes the clock-cycle count.) The cycle function can be executed by the user program at any time to read the current clock-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 of the 9826. 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 of the 9826. 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: `wtc` select code `, 32`
HP-IB type transfer: `wtc` 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 `wtc` statement can be used to set values of buffer parameters "fill", "empty", and "type". You cannot execute `wtc` to a busy buffer, nor can the buffer ever be changed from a word-type to a byte-type buffer (or vice-versa). Syntax:

`wtc` 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 <= empty <= fill <= dimension

The `wtc` 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";rds("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;gto -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):

```
rds ("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:

```
rds("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:

```
rds("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

The 9826 Computer has a built-in graphics screen capable of displaying 300 vertical by 400 horizontal dots, or pixels. Operation of the internal graphics is much the same as operation of an external plotter, 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 device, such as pen selection, line resolution, and device size.

There are two capabilities of external plotters not offered by the internal graphics device: digitize (`dig`) 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 `pclr` is executed.

The Plotter Select Code Statement

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

```
psc 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.

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 (>), 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.

Graphics Screen Control Statements

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

```
goff                (turn off graphics screen)
gon                 (turn on graphics screen)
```

These statements have no effect on the contents of the graphics memory. To clear the graphics screen use the clear graphics statement:

```
gclr                          (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:

`gdump` [select code]

The select code must include the bus address of the printer if an HP-IB printer is used, for example, "`gdump 705`". If the select code parameter is omitted, the system printer (prtsc) device is the default. 300 lines of 400 dots are always sent, unless terminated by pressing the **PAUSE** key.

The Graphics Pointer (Cursor) Statement

Interactive graphics can be achieved with the use of either an external digitizer (or data tablet) or the 9826 keyboard and knob. A graphics cursor can be drawn at point x,y using the statement:

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

The x and y coordinates must be specified, and are measured in the units set by the `scl` 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.

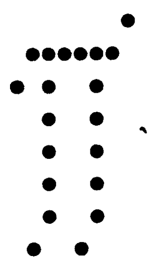
Binary Graphics

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 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. 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 bplot it:

```
0: dim A$[80];psc 16;gclr;pen;plt 3,3
1:
2:
3: char(1)&char(126)&char(164)&char(36)&char(36)→A$
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 bplot statement does not affect the pen coordinates (x,y) or the pen position (up or down).

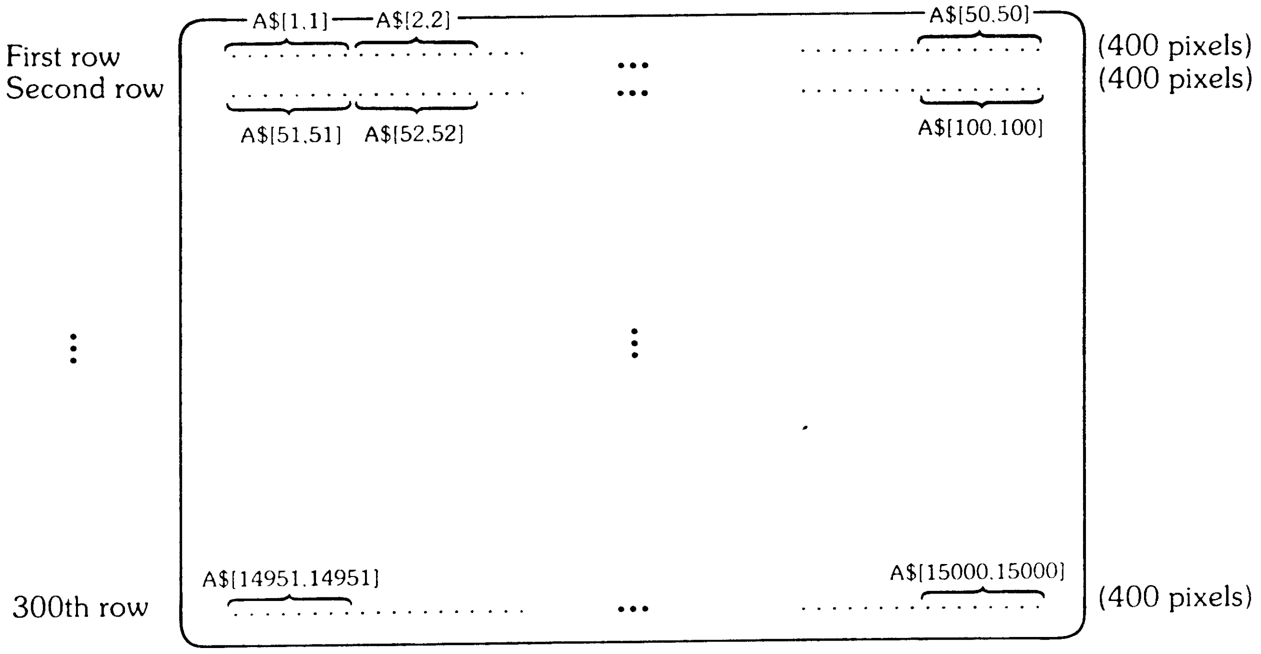
The Graphics Load and Graphics Store Statements

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:



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.

Appendix A Disc Programming Technical Appendix

Supported Device/Format Combinations

9826 HPL will support 4 distinct disc drives and 2 different mass storage directory formats. The supported drives are:

9826 internal minifloppy (5¼ inch double-sided discs)
82900 Series minifloppy (5¼ inch double-sided discs)
9885 floppy (8 inch single-sided discs)
9895 floppy (8 inch double or single-sided discs)

The supported directory formats are:

HP Corporate Logical Interchange Format (LIF)
9825 format (also 9835/9845 etc.)

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 series 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 and 8290x series drives, enabling data interchange with 9826 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 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
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
J	9895	LIF[1]	1-15	0-7	707	0-3

1 LIF: Hewlett-Packard Logical Interchange Format: File type ASCII is compatible with 9826 BASIC language ASCII files and with HP 2642A Terminal files.

9826 HPL's 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.

With 9826 HPL, the default directory size is the number of records required to fill up the first track. For 5¼ inch minifloppies (16 sectors per track), this yields 14 directory records; for 8 inch floppies (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 alphabetics and the digits 0 through 9; furthermore, the first character is to be alphabetic.

9826 HPL's LIF implementation has relaxed the file name restrictions, allowing 9825 type file names to be specified. If you wish to interchange a 9826 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 9826 HPL creates a file name entry, not when 9826 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 PROGRM, 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 9826 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 9826 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 with 9826 HPL. "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 9826 HPL and 9826 BASIC.

LIF ASCII File Support

Defined under the HP Corporate Logical Interchange Format Standard (LIF), ASCII files provide the main data interchange mechanism with 9826 BASIC. They are anticipated to provide the main data interchange mechanism with future mainframes. 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 asgn 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
asgn "ascii_file:M702,2",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,1"; 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 32767, 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; 9826 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 neither "end" or "ens" 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 "end" is specified, an attempt is made to write a logical end-of-file mark after the last data item. If not enough room exists, a file overflow error message is issued. (This is consistent with typed data file behavior.)
3. 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 under 9826 HPL's 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.

9826 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 `mrk` statement or the `open` statement. The `open` statement is the more flexible of the two, allowing arbitrary file names (and optional msus), whereas the `mrk` statement creates file names corresponding to pseudo track and file numbers.

Examples:

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

rew;trk 0;mrk 5,2000
```

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

Binary data files are assigned with either the files or the `asgn` 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
asgn "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:

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

Assigned File Pointers on 8290x Series Minifloppies

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).

The hardware of the current 8290x Series Minifloppies, however, is incapable of detecting 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 Present

The 9885 driver on the 9825 used the 9825's built-in DMA channel to access the 9885 disc. The 9885 driver on the 9826 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 HPIB Programming Considerations mentioned in the "9825 Disc Programming Manual", part # 09825-90220, are valid for the 9826. However, due to hardware and low-level driver differences, items 1, 8, & 9 are not valid for 9826 HPL.

9826 HPL Disc Programming Error Messages

When 9826 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 9826 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 9826 command and cursor-control keys. This table is useful when developing programs that use the pkbd statement, and also when controlling the 9826 from a remote ASCII keyboard (such as a terminal).

The Keycode Conversion Table on pages B-3, B-4 is useful for designing a 9826-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 9826 8-bit keycode, and the **character** at that position corresponding to the 9825 keycode.

9826A ASCII Control Codes

CTRL of	ASCII Value	ASCII Character	9826A 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	🔔
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	$^{E}{}_{C}$
\\[2]	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 9826A 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

9826 Hardware Keycodes *			ASCII		9825 Hardware Keycodes		
DEC	OCT	HEX	CHAR	DEC	DEC	OCT	HEX
000	000	00	N_U	000	000	000	00
001	001	01		032	000	000	00
002	002	02		032	000	000	00
003	003	03		032	000	000	00
004	004	04		032	000	000	00
005	005	05		032	000	000	00
006	006	06		032	000	000	00
007	007	07		032	000	000	00
008	010	08		032	000	000	00
009	011	09		032	000	000	00
010	012	0A		032	000	000	00
011	013	0B		032	000	000	00
012	014	0C		032	000	000	00
013	015	0D		032	000	000	00
014	016	0E		032	000	000	00
015	017	0F		032	000	000	00
016	020	10		032	000	000	00
017	021	11		032	000	000	00
018	022	12		032	000	000	00
019	023	13		032	000	000	00
020	024	14		032	000	000	00
021	025	15		032	000	000	00
022	026	16		032	000	000	00
023	027	17		032	000	000	00
024	030	18	G_S	029	000	000	00
025	031	19	→	125	125	175	7D
026	032	1A	¥	128	065	101	41
027	033	1B	^	129	066	102	42
028	034	1C	r	130	067	103	43
029	035	1D	·	133	070	106	46
030	036	1E	∋	134	071	107	47
031	037	1F	⊽	135	072	110	48
032	040	20	ᴊ	131	068	104	44
033	041	21	\	132	069	105	45
034	042	22	D_L	016	016	020	10
035	043	23	D_1	017	017	021	11
036	044	24	◂	136	073	111	49
037	045	25	⊤	137	074	112	4A
038	046	26	D_4	020	020	024	14
039	047	27	N_K	021	021	025	15
040	050	28	B_S	008	008	010	08
041	051	29	H_T	009	009	011	09
042	052	2A	V_T	011	011	013	0B
043	053	2B	S_Y	022	022	026	16
044	054	2C	E_B	023	023	027	17
045	055	2D	S_O	014	000	000	00
046	056	2E	D_4	020	020	024	14
047	057	2F	F_F	012	012	014	0C
048	060	30	F_S	028	028	034	1C
049	061	31	R_S	030	000	000	00
050	062	32	U_S	031	000	000	00
051	063	33	C_N	024	024	030	18
052	064	34	D_2	018	018	022	12
053	065	35	🔔	007	007	007	07
054	066	36	D_3	019	019	023	13
055	067	37	◀	252	000	000	00
056	070	38	S_H	001	001	001	01
057	071	39	C_R	013	013	015	0D
058	072	3A	E_M	025	025	031	19
059	073	3B	L_F	010	010	012	0A
060	074	3C	0	048	078	116	4E
061	075	3D	.	046	088	130	58
062	076	3E	,	044	089	131	59
063	077	3F	+	043	043	053	2B
064	100	40	1	049	079	117	4F
065	101	41	2	050	080	120	50
066	102	42	3	051	081	121	51
067	103	43	-	045	045	055	2D
068	104	44	4	052	082	122	52
069	105	45	5	053	083	123	53
070	106	46	6	054	084	124	54
071	107	47	*	042	042	052	2A
072	110	48	7	055	085	125	55
073	111	49	8	056	086	126	56
074	112	4A	9	057	087	127	57
075	113	4B	/	047	047	057	2F
076	114	4C	e	101	096	140	60
077	115	4D	(	040	040	050	28
078	116	4E	)	041	041	051	29
079	117	4F	^	094	094	136	5E
080	120	50	1	049	049	061	31
081	121	51	2	050	050	062	32
082	122	52	3	051	051	063	33
083	123	53	4	052	052	064	34
084	124	54	5	053	053	065	35
085	125	55	6	054	054	066	36
086	126	56	7	055	055	067	37
087	127	57	8	056	056	070	38
088	130	58	9	057	057	071	39
089	131	59	0	048	048	060	30
090	132	5A	-	045	045	055	2D
091	133	5B	=	061	061	075	3D
092	134	5C	[	091	184	270	B8
093	135	5D	]	093	185	271	B9
094	136	5E	;	059	059	073	3B
095	137	5F	´	039	176	260	B0
096	140	60	,	044	044	054	2C
097	141	61	.	046	046	056	2E
098	142	62	/	047	047	057	2F
099	143	63		032	032	040	20
100	144	64	o	111	111	157	6F
101	145	65	p	112	112	160	70
102	146	66	k	107	107	153	6B
103	147	67	l	108	108	154	6C
104	150	68	q	113	113	161	71
105	151	69	w	119	119	167	77
106	152	6A	e	101	101	145	65
107	153	6B	r	114	114	162	72
108	154	6C	t	116	116	164	74
109	155	6D	y	121	121	171	79
110	156	6E	u	117	117	165	75
111	157	6F	i	105	105	151	69
112	160	70	a	097	097	141	61
113	161	71	s	115	115	163	73
114	162	72	d	100	100	144	64
115	163	73	f	102	102	146	66
116	164	74	g	103	103	147	67
117	165	75	h	104	104	150	68
118	166	76	j	106	106	152	6A
119	167	77	m	109	109	155	6D
120	170	78	z	122	122	172	7A
121	171	79	x	120	120	170	78
122	172	7A	c	099	099	143	63
123	173	7B	v	118	119	166	76
124	174	7C	b	098	098	142	62
125	175	7D	n	110	110	156	6E
126	176	7E		032	000	000	00
127	177	7F		032	000	000	00
128	200	80		032	000	000	00
129	201	81		032	000	000	00
130	202	82		032	000	000	00
131	203	83		032	000	000	00
132	204	84		032	000	000	00
133	205	85		032	000	000	00
134	206	86		032	000	000	00
135	207	87		032	000	000	00
136	210	88		032	000	000	00
137	211	89		032	000	000	00
138	212	8A		032	000	000	00
139	213	8B		032	000	000	00
140	214	8C		032	000	000	00
141	215	8D		032	000	000	00
142	216	8E		032	000	000	00
143	217	8F		032	000	000	00
144	220	90		032	000	000	00
145	221	91		032	000	000	00
146	222	92		032	000	000	00
147	223	93		032	000	000	00
148	224	94		032	000	000	00
149	225	95		032	000	000	00
150	226	96		032	000	000	00
151	227	97		032	000	000	00

* Lower 8 bits of 9826 keycode only.

9826 Hardware Keycodes *			ASCII		9825 Hardware Keycodes		
DEC	OCT	HEX	CHAR	DEC	DEC	OCT	HEX
152	230	98	G_S	029	000	000	00
153	231	99	→	125	125	175	7D
154	232	9A	ɣ	138	075	113	4B
155	233	9B	×	139	076	114	4C
156	234	9C	Þ	140	193	301	C1
157	235	9D	‥	143	196	304	C4
158	236	9E	·	144	197	305	C5
159	237	9F	ʔ	145	198	306	C6
160	240	A0	ɹ	141	194	302	C2
161	241	A1	э	142	195	303	C3
162	242	A2	E_Q	005	144	220	90
163	243	A3	A_K	006	145	221	91
164	244	A4	ɾ	146	199	307	C7
165	245	A5	ɂ	147	200	310	C8
166	246	A6	E_X	003	148	224	94
167	247	A7	E_T	004	149	225	95
168	250	A8	B_S	008	136	210	88
169	251	A9	H_T	009	137	211	89
170	252	AA	V_T	011	139	213	8B
171	253	AB	S_Y	022	150	226	96
172	254	AC	E_B	023	151	227	97
173	255	AD	█	251	000	000	00
174	256	AE	D_4	020	148	224	94
175	257	AF	F_F	012	140	214	8C
176	260	B0	E_C	027	156	234	9C
177	261	B1	▯	254	000	000	00
178	262	B2	S_B	026	000	000	00
179	263	B3	▣	253	152	230	98
180	264	B4	S_I	015	146	222	92
181	265	B5	◀	252	135	207	87
182	266	B6	◀	252	147	223	93
183	267	B7	S_H	001	001	001	01
184	270	B8		032	129	201	81
185	271	B9	C_R	013	141	215	8D
186	272	BA	E_M	025	153	231	99
187	273	BB	L_F	010	138	212	8A
188	274	BC	0	048	206	316	CE
189	275	BD	.	046	216	330	D8
190	276	BE	,	044	217	331	D9
191	277	BF	+	043	171	253	AB
192	300	C0	1	049	207	317	CF
193	301	C1	2	050	208	320	D0
194	302	C2	3	051	209	321	D1
195	303	C3	-	045	173	255	AD
196	304	C4	4	052	210	322	D2
197	305	C5	5	053	211	323	D3
198	306	C6	6	054	212	324	D4
199	307	C7	*	042	170	252	AA
200	310	C8	7	055	213	325	D5
201	311	C9	8	056	214	326	D6
202	312	CA	9	057	215	327	D7
203	313	CB		047	175	257	AF
204	314	CC		096	000	000	00
205	315	CD	|	124	251	373	FB
206	316	CE	\	092	222	336	DE
207	317	CF	~	126	000	000	00
208	320	D0	!	033	177	261	B1
209	321	D1	@	064	183	267	B7
210	322	D2	#	035	179	263	B3
211	323	D3	$	036	180	264	B4
212	324	D4	%	037	181	265	B5
213	325	D5		094	094	136	5E
214	326	D6	&	038	182	266	B6
215	327	D7	*	042	042	052	2A
216	330	D8	(	040	168	250	A8
217	331	D9	)	041	169	251	A9
218	332	DA	_	095	000	000	00
219	333	DB	+	043	171	253	AB
220	334	DC	π	123	123	173	7B
221	335	DD	→	125	125	175	7D
222	336	DE	:	058	178	262	B2
223	337	DF	"	034	191	277	BF
224	340	E0	<	060	172	254	AC
225	341	E1	>	062	174	256	AE
226	342	E2	?	063	063	077	3F
227	343	E3		032	160	240	A0
228	344	E4	O	079	239	357	EF
229	345	E5	P	080	240	360	F0

9826 Hardware Keycodes *			ASCII		9825 Hardware Keycodes		
DEC	OCT	HEX	CHAR	DEC	DEC	OCT	HEX
230	346	E6	K	075	235	353	EB
231	347	E7	L	076	236	354	EC
232	350	E8	Q	081	241	361	F1
233	351	E9	W	087	247	367	F7
234	352	EA	E	069	229	345	E5
235	353	EB	R	082	242	362	F2
236	354	EC	T	084	244	364	F4
237	355	ED	Y	089	249	371	F9
238	356	EE	U	085	245	365	F5
239	357	EF	I	073	233	351	E9
240	360	F0	A	065	225	341	E1
241	361	F1	S	083	243	363	F3
242	362	F2	D	068	228	344	E4
243	363	F3	F	070	230	346	E6
244	364	F4	G	071	231	347	E7
245	365	F5	H	072	232	350	E8
246	366	F6	J	074	234	352	EA
247	367	F7	M	077	237	355	ED
248	370	F8	Z	090	250	372	FA
249	371	F9	X	088	248	370	F8
250	372	FA	C	067	227	343	E3
251	373	FB	V	086	246	366	F6
252	374	FC	B	066	226	342	E2
253	375	FD	N	078	238	356	EE
254	376	FE		032	000	000	00
255	377	FF		032	000	000	00

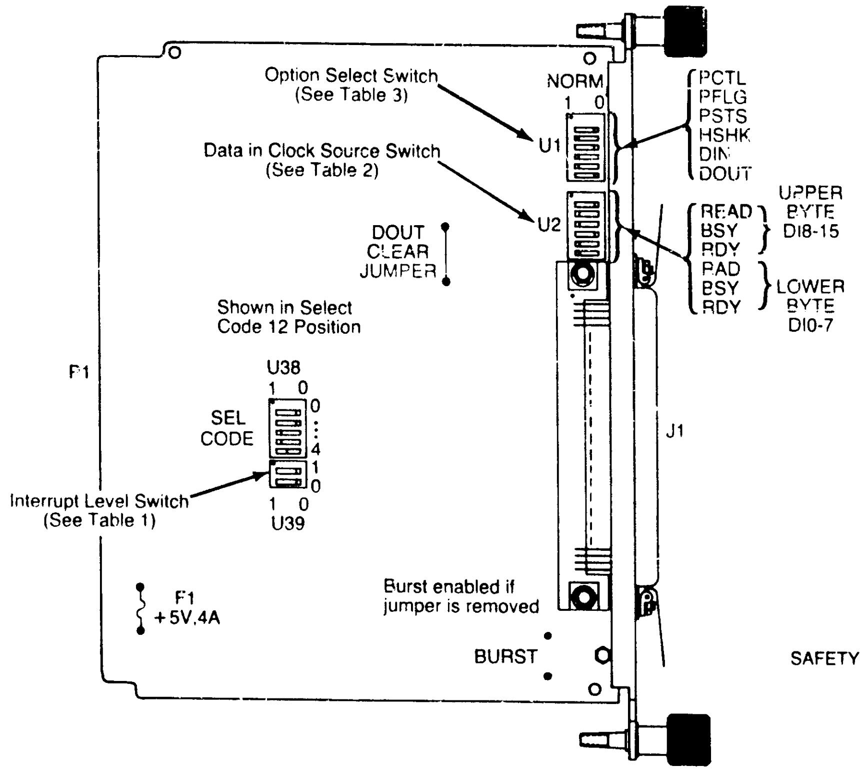
* Lower 8 bits of 9826 keycode only.

Appendix C
Interface Board Diagrams

The following diagrams document the interface board switch positions and meanings of the 98622A, 98623A, 98624A, and 98626A Interfaces. In particular, note the Remote Keyboard Jumper on the RS232 Interface Board of page C-4. Cutting this jumper enables the interface as a power-up remote keyboard, described in the Systems Programming chapter of the 9825 Operating and Programming Reference (p.7-19).

These diagrams are included here as quick-reference information only. For more explicit information regarding initial set-up requirements, refer to the appropriate interface installation and service manual.

GPIO Interface Board


Option Select Switch (See Table 3)
Data in Clock Source Switch (See Table 2)
NORM
1 0
U1
U2
PCTL
PFLG
PSTS
HSHK
DIN
DOUT
READ
BSY
RDY
UPPER BYTE DI8-15
READ
BSY
RDY
LOWER BYTE DI0-7
DOUT CLEAR JUMPER
Shown in Select Code 12 Position
F1
U38
1 0
SEL CODE
0
4
1
0
1 0
U39
Interrupt Level Switch (See Table 1)
J1
F1 +5V,4A
Burst enabled if jumper is removed
BURST

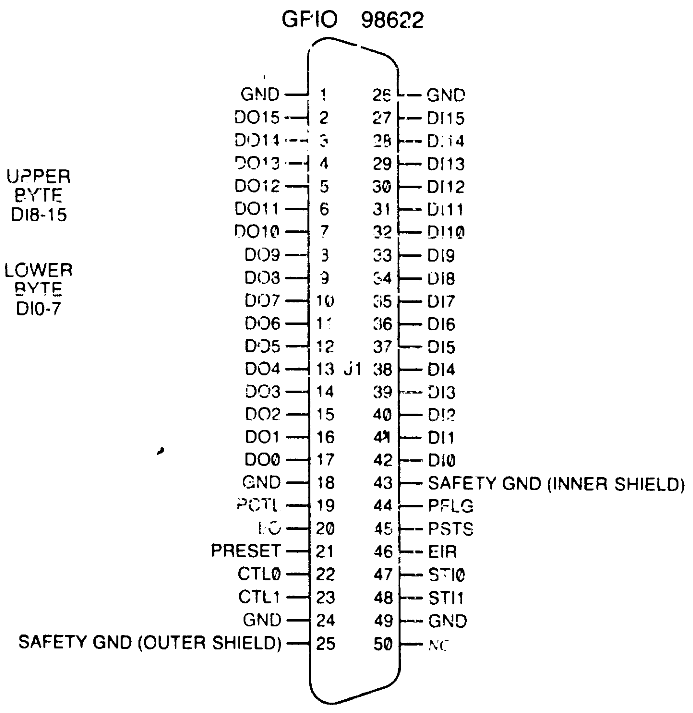


Table 1: Interrupt Level Switch

Interrupt Level	SW. Settings	
	1	0
3	0	0
4	0	1
5	1	0
6	1	1

Table 2: Data in Clock Source Switch

Switch Name	RDY	BSY	READ	RDY	BSY	READ
Switch Position 0 or Closed	PFLG busy to ready transition	PFLG ready to busy transition	READ of upper data input register	PFLG busy to ready transition	PFLG ready to busy transition	READ of lower data input register
	DI8-15 Upper Byte Under no circumstances should more than one switch from the upper byte clock source switch be in the closed or 0 position at the same time.			**DI0-7 Lower Byte** Under no circumstances should more than one switch from the lower byte clock source switch be in the closed or 0 position at the same time.		

Table 3: Option Select Switch

Switch Position Name	DOUT	DIN	HSHK	PSTS	PFLG	PCTL
Function	Invert Data Out	Invert Data In	Full/Pulse Handshake	Invert PSTS	Invert PFLG	Invert PCTL
Switch Position Open or 1	Low = 1 High = 0	Low = 1 High = 0	Full	Low = $\overline{\text{OK}}$ High = OK	Low = Ready High = Busy	Low = Set High = Clear
Switch Position Closed 0	Low = 0 High = 1	Low = 0 High = 1	Pulse	Low = OK High = $\overline{\text{OK}}$	Low = Busy High = Ready	Low = Clear High = Set

BCD Interface Board

Table 1: U24 Peripheral Status Switch

Description	Switch Positions	
	On	Off
Option Format	Std. Format	Option Format
Data	Pos. True, Active High	Inverted
Sign 1	Pos. True, Active High	Inverted
Sign 2	Pos.True, Active High	Inverted
Overload	Pos. True, Active High	Inverted

Table 2: U15 Handshake Switch

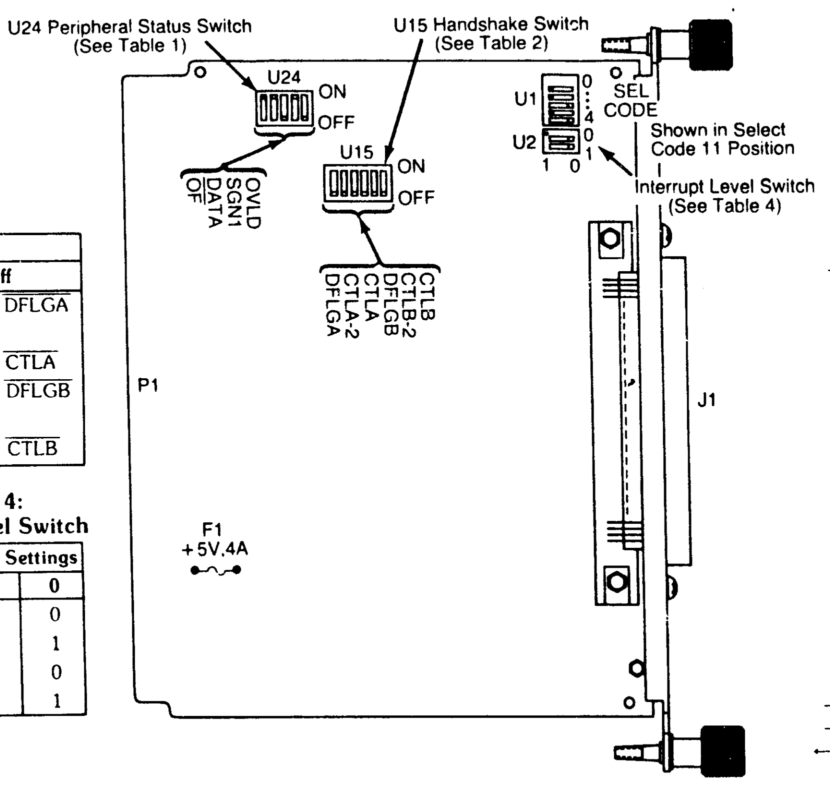
Description	Switch Positions	
	On	Off
Device Flag A	Active High, DFLGA	Inverted, $\overline{\text{DFLGA}}$
Control A 1 or 2 (Type)	CTLA-1	CTLA-2
Control A (Std. or Invert)	Active High, CTLA	Inverted, $\overline{\text{CTLA}}$
Device Flag B	Active High, DFLGB	Inverted, $\overline{\text{DFLGB}}$
Control B 1 or 2 (Type)	CTLB-1	CTLB-2
Control B (Std. or Invert)	Active High, CTLB	Inverted, $\overline{\text{CTLB}}$

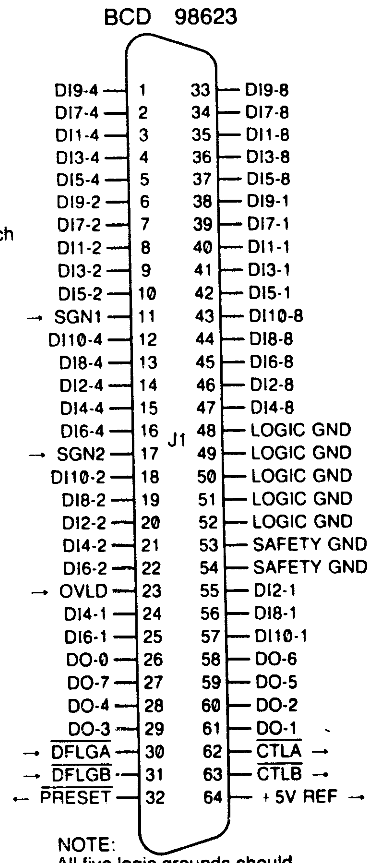
Table 3: BCD Code

Pos. True Logic	ASCII Character
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9
1 0 1 0	(L.F.) line feed
1 0 1 1	(+) plus
1 1 0 0	(,) comma
1 1 0 1	(–) minus
1 1 1 0	(E) exponent
1 1 1 1	(.) decimal point

Table 4: Interrupt Level Switch

Interrupt Level	SW. Settings	
	1	0
3	0	0
4	0	1
5	1	0
6	1	1


U24 Peripheral Status Switch (See Table 1)
U15 Handshake Switch (See Table 2)
U24
ON
OFF
OVLD
SGN2
SGN1
DATA
OF
U15
ON
OFF
CTLB
CTLB-2
DFLGB
CTLA
CTLA-2
DFLGA
U1
U2
0
4
0
1
1 0
SEL CODE
Shown in Select Code 11 Position
Interrupt Level Switch (See Table 4)
P1
J1
F1
+5V,4A



NOTE:
All five logic grounds should be paralleled at peripheral.

Table 5: Standard Format

Name	SGN1	DI1	DI2	DI3	DI4	DI5	DI6	DI7	DI8	Exponent	SGN2	DI9	Comma	OVLD	DI10	Line Feed
Char.	+ –	X	X	X	X	X	X	X	X	E	+ –	X		0 or 8	X	L.F.
Info.		MSD	←					→	LSD					0 = OVLD 8 = $\overline{\text{OVLD}}$		
Code (Pos. True)	+ 1011 – 1101									1110	+ 1011 – 1101		1100	0000 1000		1010

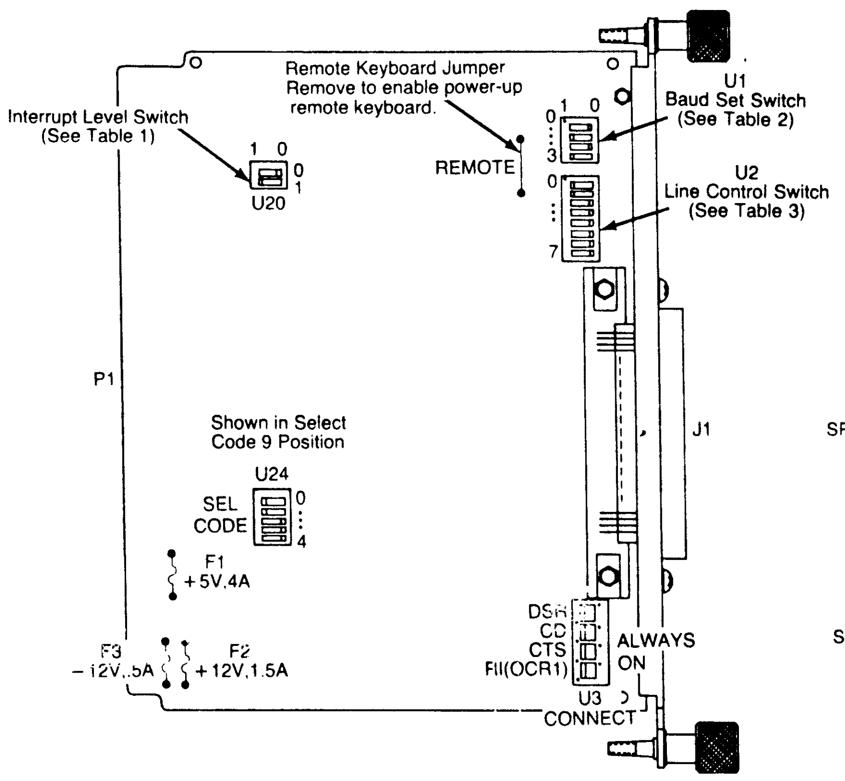
Table 6: Overload Table

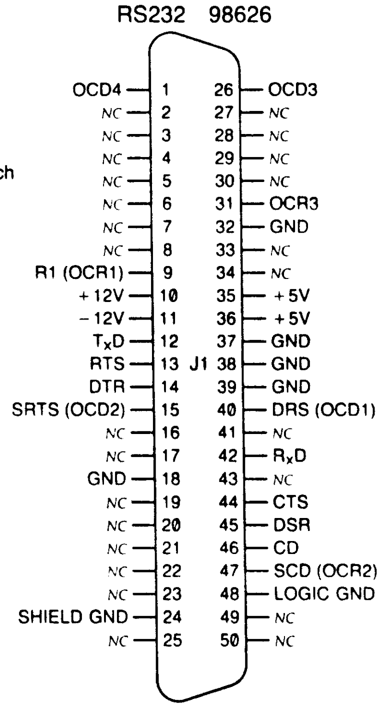
Description	OVLD	DI9
No Overload	0	0
1st Device Overload	8	0
2nd Device Overload	0	8
Both Overload	8	8

Table 7: Optional Format (Read Two BCD Devices)

	First Device (FD)						Second Device (SD)									
Name	SGN1	DI4	DI2	DI6	DI8	Comma	SGN2	DI10	DI1	DI5	DI3	DI7	Letter E	OVLD	DI9	Line Feed
Char	+ –	X	X	X	X		+ –	X	X	X	X	X	E	0 or 8	0 or 8	L.F.
Info.		MSD	←	→	LSD		MSD	←			→	LSD		FD (see table)	SD	
Code (Pos. True)	+ 1011 – 1101					1100	+ 1011 – 1101						1110			1010

RS232 Interface Board


Interrupt Level Switch (See Table 1)
1 0
0 1
U20
Remote Keyboard Jumper Remove to enable power-up remote keyboard.
REMOTE
0 1 0
3
U1 Baud Set Switch (See Table 2)
0
7
U2 Line Control Switch (See Table 3)
P1
J1
Shown in Select Code 9 Position
U24
SEL CODE
0
4
F1 +5V.4A
F3 −12V,.5A
F2 +12V,1.5A
DSR
CD
CTS
RI(OCR1)
ALWAYS ON
U3
CONNECT



Signal	Pin	Pin	Signal
OCD4	1	26	OCD3
NC	2	27	NC
NC	3	28	NC
NC	4	29	NC
NC	5	30	NC
NC	6	31	OCR3
NC	7	32	GND
NC	8	33	NC
R1 (OCR1)	9	34	NC
+12V	10	35	+5V
−12V	11	36	+5V
T_xD	12	37	GND
RTS	13	38	GND
DTR	14	39	GND
SRTS (OCD2)	15	40	DRS (OCD1)
NC	16	41	NC
NC	17	42	R_xD
GND	18	43	NC
NC	19	44	CTS
NC	20	45	DSR
NC	21	46	CD
NC	22	47	SCD (OCR2)
NC	23	48	LOGIC GND
SHIELD GND	24	49	NC
NC	25	50	NC

Table 1: Interrupt Level Switch

Interrupt Level	SW. Settings	
	1	0
3	0	0
4	0	1
5	1	0
6	1	1

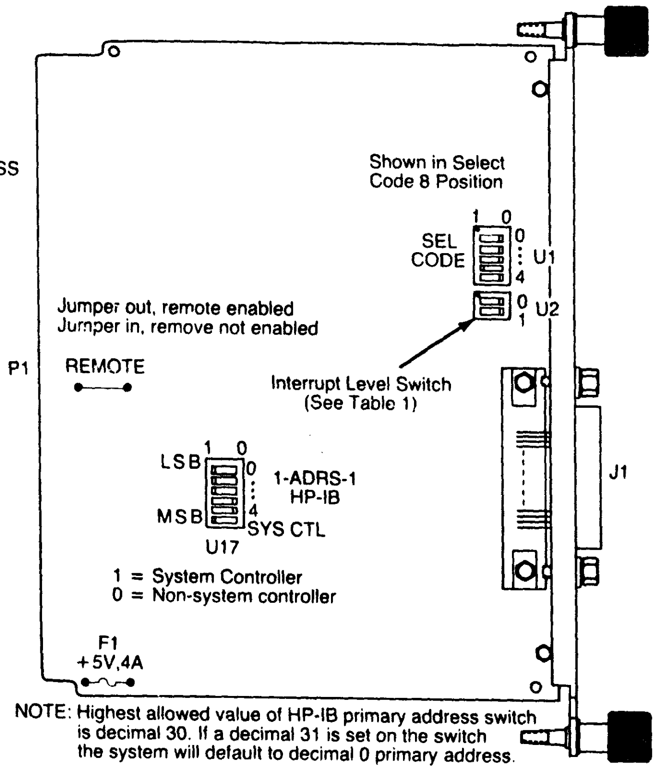
Table 2: Baud Rate Set Switch

Baud Rate	Switch Settings			
	3	2	1	0
50	0	0	0	0
75	0	0	0	1
110	0	0	1	0
134.5	0	0	1	1
150	0	1	0	0
200	0	1	0	1
300	0	1	1	0
600	0	1	1	1
1200	1	0	0	0
1800	1	0	0	1
2400	1	0	1	0
3600	1	0	1	1
4800	1	1	0	0
7200	1	1	0	1
9600	1	1	1	0
19200	1	1	1	1

Table 3: Line Control Switch Setting

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Handshake Type		Parity Type		Parity Enabled	Stop	Character Length	
Don't Care	Don't Care	0	0 = Odd	Parity Enable Generation and Checking	0 = Stop Bit Added	0	0 = 5 Bits/Char
Reserved For Future Use	Reserved For Future Use	0	1 = Even		1 = 1.5 Stop Bits @ 5 Bits/Char	0	1 = 6 Bits/Char
		1	0 = Parity Bit = 1			1	0 = 7 Bits/Char
		1	1 = Parity Bit = 0	1 = Enabled	1 = 2.0 Stop Bits @ 6, 7 or 8 Bits/Char	1	1 = 8 Bits/Char

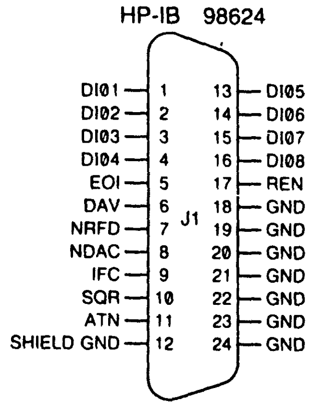
NOTE: INTERNAL POWER-UP ADDRESS
If controller, decimal 21
If not controller, decimal 20


Shown in Select Code 8 Position
1 0
SEL CODE
0
4
U1
0
1
U2
Jumper out, remote enabled
Jumper in, remove not enabled
P1
REMOTE
Interrupt Level Switch (See Table 1)
J1
1 0
LSB
MSB
0
4
1-ADRS-1
HP-IB
SYS CTL
U17
1 = System Controller
0 = Non-system controller
F1
+5V,4A

NOTE: Highest allowed value of HP-IB primary address switch is decimal 30. If a decimal 31 is set on the switch the system will default to decimal 0 primary address.

Table 1:
Interrupt Level Switch

Interrupt Level	SW. Settings	
	1	0
3	0	0
4	0	1
5	1	0
6	1	1



Appendix D
Alphabetical List of HPL Extensions/Revisions

`aclr` [number of pages]
  Clears screen, allocates specified number of scrolling pages

`adump` select code
  Dumps alpha screen to printer

`aoff`
  Alpha off

`aon`
  Alpha on

`bplt` string , # bytes per line [,function]
  Plots binary data in string to the graphics screen.
  Function: 0 = OR, 1 = AND, 2 = EOR, 3 = STORE

`cret`
  Returns to the program from an on-cycle routine.

`cycle`
  Returns the number of clock cycles since last executed.
  One cycle = 10 milliseconds.

`data` numeric or string constant [,numeric or string constant]...
  Provides constants for `read` statement variables.

`dret`
  Returns to the program from an on-delay routine.

`gclr`
  Clears the graphics screen.

`gdump` select code
  Dumps the graphics screen to a raster-scan standard printer.

`goff`
  Turns the graphics display off.

`gon`
  Turns the graphics display on.

`gptr` xcoord ,ycoord[,type]
  Draws a graphics cursor at the specified location. (type: 0 = off)

`gload` string expression
Loads the graphics screen from the specified string.

`gstore` string expression
Store graphics screen to the specified string.

`Kloff`
Turns special function key labels off.

`Klon`
Turns special function key labels on.

`Knob`
Returns the accumulated knob count. CCR rotation is negative valued, CR rotation is positive valued.

`Kret`
Returns to the program from an on-knob routine.

`Kstat`
Returns knob status: Bit: 6 = Knob, 5 = $\overline{\text{Control}}$, 4 = $\overline{\text{Shift}}$

`mret`
Returns to the program from an on-match routine.

`msi` [`:`device format [controller select code [`,`unit number]]]
Sets the current system disc drive and format.
Drive formats: I=internal disc, M=8290x minifloppy, F=9885(9825), G=9885(LIF), H=9895(9825), J=9895(LIF)

`on cycle` time [`,`label]
Sets up clock periodic interrupt service routine. `on cycle` with no label specified cancels clock-cycle interrupt service.

`on delay` time [`,`label]
Sets up clock delay interrupt service routine. `on delay` with no label specified cancels clock-delay interrupt service.

`on Knob` [label]
Sets up Knob interrupt service routine. `on Knob` with no label cancels Knob interrupt service.

`on match` time[`,`label]
Sets up clock match interrupt service routine. `on match` with no label cancels clock-match interrupt service.

`pbeep` [frequency [`,`duration]]
Programmable beep with frequency (0-5167 Hz) and duration (0-2.56 seconds).

`pi`
Returns value of pi.

`pKbd` [string]
Executes ASCII string (as if it were pushing keys).

`prtsc` select code [,width]
  Sets system printer select code (and width)

`rcb` [file number]
  Records the binary program in memory to the specified tape file.

`rds` (buffer name[, type [, empty [, fill [, dim]]]])$\longrightarrow$status
  Extended buffer status function.

`read` variable name [,variable name]
  Reads data statement constants into variables.

`rkbd` select code [,type]
  Enables remote keyboard: type 1 = ASCII, type 0 = 9826A

`rstr` [label]
  Resets data pointer either to line 0, or to "label" if specified.

`rtime`
  Returns internal clock value in elapsed seconds.

`saveb` file name
  Saves the binary program in memory to the specified file.

`sfk` key number [,definition string [,label string]]
  Defines SFK (0 to 32) and optional soft label.

`sqr` variable
  Returns the square root of the specified variable.

`stime` seconds
  Sets the internal clock to the specified number of seconds.

`sysboot` [system name]
  Boots language system specified from disc or ROM.

`tabxy` xcol ,yrow
  Moves print position to column x (0...49), row y (0...17).

`wtc` buffer name [,type[,empty[,fill]]]
  Write buffer pointers to specified buffer name.

`wtc` HP-IB select code, value
  Resets HP-IB interface. If "value" is <31, sets new bus address = "value". If "value" =31, no further action. If "value" >31, "value" configures Parallel Poll response.

9826A HPL Error Messages

There are some new error messages specific to the 9826. These are the "X", "f", and "P9" error messages.

HPL Language Errors

X0	No memory or I/O card present at specified address. This error should not be encountered when programming from HPL. Consult your HP field sales and service office for advice concerning this error should you receive it.
X1	A read statement was executed with no data remaining. Either a data statement must be added or a `rstr` statement must be added to reset the data pointer to the desired data statement in the program.

Disc Programming Errors

f0	Unable to access disc controller. This error has the same cause as the error which issued the "DISK IS DOWN" and "UNABLE TO ACCESS DISK CONTROLLER" messages, except the error is now trappable by on err.
f1	No DMA card present for 9885 disc controller.
f2	Invalid msus syntax. Probable illegal device/format specifier.
f3	Directory entry field overflow. Attempted file copy not possible.
f4	Illegal structure on LIF format disc. The disc cannot be repacked.
f5	Disc copy attempted to a significantly larger disc. Use file copy to back up contents of disc.
f6	Disc copy attempted from 9825-compatible disc to LIF disc, or vice-versa. Only file copy is allowed across media formats.
f7	System record is not valid for LIF disc.
f8	System record is not valid for 9825-compatible disc.
f9	Statement not implemented on 9826A. (See Disc Programming Technical Appendix for a list of disc statements that are not implemented.)

Plotter Programming Errors

P9	No graphics hardware present. If you have a 9826A, and `psc 16` has been executed, you should not experience this error. Consult your HP field sales and service office for advice.

HEWLETT
PACKARD

Part No. 09826-90040
Microfiche No. 09826-99040

Printed in U.S.A.
First Edition June 1981