Installing, Using, and Maintaining the BASIC 5.0 System

HP 9000 Series 200/300 Computers

HP Part Number 98613-90042

HEWLETT
PACKARD

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

NOTICE

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

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

WARRANTY

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend

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

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

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California.

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.

February 1987...Edition 1

July 1987...Update

August 1987...Edition 2. Update merged.

Table Of Contents

Chapter 1: Loading BASIC

Chapter 2: Verifying and Labeling Peripherals

Chapter 3: Mass Storage Concepts

Chapter 4: Preparing Flexible Discs

Chapter 5: Putting BASIC on a Hard Disc

Chapter 6: Putting BASIC on the SRM

Chapter 7: Language Extensions, Drivers, and Configuration

Chapter 8: Introduction to the System

Chapter 9: Loading and Running Programs

Chapter 10: Using Directories and Files

Chapter 11: Editing and Storing Programs

Chapter 12: Overview

Chapter 13: Using COPY for Back-Ups

Chapter 16: HFS File System Implementation

Chapter 17: Other Maintenance Tasks

Chapter 18: BASIC Utilities Library

Installing BASIC: Introduction

The "Installing" section of this manual consists of chapters which describe in detail the steps for installing your BASIC system. The white card you received with the BASIC discs and the READ ME FIRST pamphlet will outline the necessary steps for loading BASIC. This section will give you the details for both loading and installing your system. The following is a summary of the steps covered in this section:

Summary of Installation Steps

Task/Topic	Chapter
Set up your hardware (see the *Peripheral Installation Guide*)	n/a
1. Load BASIC (from flexible discs into memory)	1
2. Verify and label peripherals (optional)	2
3. Read about mass storage concepts (optional)	3
4. Prepare flexible disc (if you have only floppy disc drives)	4
5. Put BASIC on your hard disc (if you have a hard disc drive)	5
6. Put BASIC on your SRM system (if you have an SRM system)	6
7. Customize your system configuration	7

Loading BASIC 1

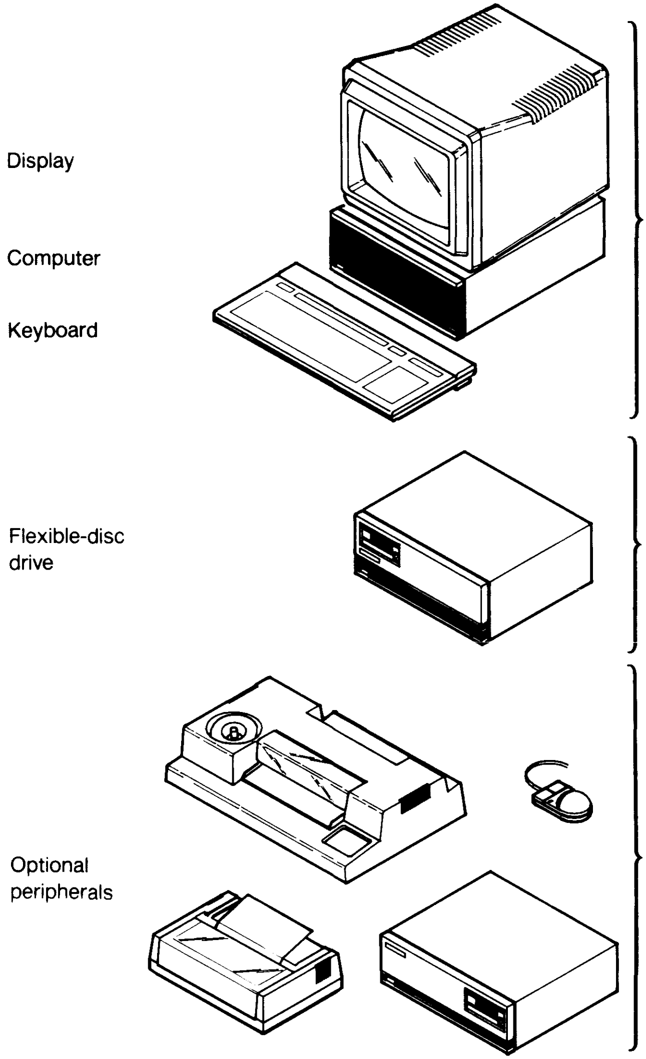
Before you can use the BASIC system, you will need to load it into memory from the system discs. This process is virtually automatic, involving little more than inserting a disc into a disc drive and turning on the computer. "Loading" and "booting" BASIC both mean moving the system from the flexible discs into the computer's memory.

Note

If you have a ROM-based BASIC system, then most of the system is ready to use nearly as soon as you power up your computer. However, you will still need to go through many of the same steps that you would in loading BASIC from discs.

Prerequisites

Before attempting to load and begin using BASIC, the **following equipment must already be operational**:



If these items are not set up and operational, refer to your computer's *Installation Card* or *Installation Reference.*

If at least one flexible-disc drive is not set up and operational, then refer to the *Peripheral Installation Guide.*

If you want to connect additional peripherals to your computer, then refer to the *Peripheral Installation Guide.*

Figure 1-1. Computer Must Be Installed (Peripherals are optional)

Using BASIC with HP-UX

In BASIC 5.0, the default mode of a file is assigned when the file is created (which can be changed with the HP-UX `chmod` command or the BASIC command, `PERMIT`). At that time, a user and group identifier are also assigned to the file. All BASIC files will not be created with the same user and group identifier values. However, you can change the values for particular directories and files in BASIC with `CHOWN` and `CHGRP`.

There will be a unique user identifier for BASIC, Pascal and a unique group identifier to be shared by BASIC and Pascal. These identifiers will be reserved in HP-UX by convention. These default values are:

```
user_id:          18
user_name:        pws
user_id:          17
group_name:       workstations
group_id:         9
file mode:        -rw-rw-rw-
dir mode:         drwxrwxrwx
```

An HP-UX system's `/etc/passwd` file could contain the following additional entries:

```
basic:*:18:9:#BASIC workstation user:/users/workstation/basic:/bin/false
pws:*:17:9:#Pascal workstation user:/user/workstation/pws:/bin/false
```

A new entry for the `/etc/group` file would be:

```
workstation::9:basic,pws
```

The `PERMIT` command provides a subset of the permission setting capabilities of HP-UX. It provides a "change mode" capability. `PERMIT` will not be concerned with all nine permission bits of the mode, but will provide a subset of the permission setting capabilities of HP-UX. **BASIC will not support execute permission for files.** You will be able to set the execute bit, but the BASIC operating system never checks for it on files. BASIC does support execute permission for directories (using the `SEARCH` secondary keyword). BASIC's `CHOWN` and `CHGRP` allow you to change the owner or group for a file. Thus, BASIC files can be shared with a particular HP-UX user or group, while restricting access to everyone else.

Before You Begin Loading BASIC

If you have an HP-UX system already running, and you wish to load BASIC, you will need to **first** follow these steps:

1. Login as superuser: `su` [Return]
2. Change directory to root: `cd /` [Return]
3. Write enable the root: `chmod 777 /` [Return]
4. Shutdown the system: `shutdown -h` [Return]
5. Wait for the prompt: `halted`
   then power down the system.
6. Continue with "Step 1: Turn the computer off".

Using Softkeys

Before you begin the loading procedure, you should become familiar with your keyboard. There are several types of keyboards available for use, and therefore several different kinds of keys. The ITF and 98203/C keyboards are most common. The following diagram shows how to determine which keys to press when you see softkey labels on the computer screen:

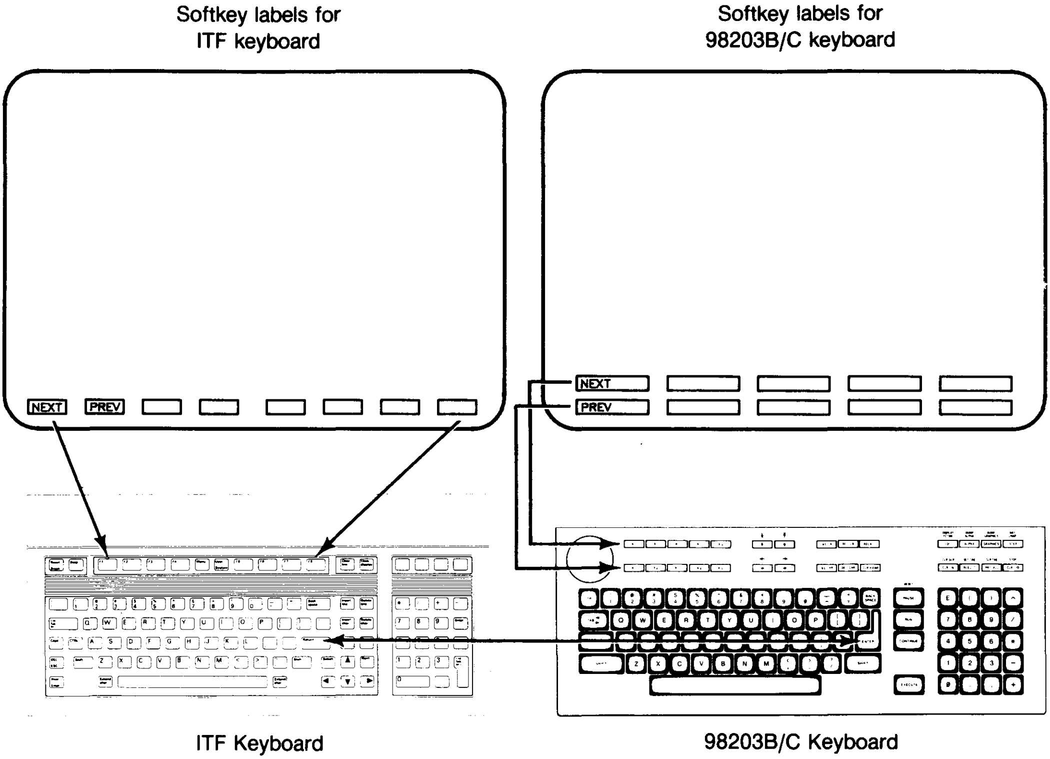

Softkey labels for ITF keyboard
Softkey labels for 98203B/C keyboard
NEXT
PREV
NEXT
PREV
ITF Keyboard
98203B/C Keyboard

Figure 1-2. Keyboard Softkey Comparison

Each label corresponds to a softkey on the top left of the keyboard (*softkey labels* are the key representation on the screen; *softkey caps* are the actual keys on the keyboard). With the ITF keyboard, the softkey caps are labeled: f1, f2 through f8. With the 98203 keyboard, they are labeled: k0 through k9.

If you need to use the [Next] softkey, look on the keyboard for the corresponding softkey cap. For example, [Next] corresponds to [f1] on the ITF keyboard and [k0] on the 98203 keyboard; [Prev] corresponds to [f2] and [k1], and so forth.

The diagram also shows that [Return] on the ITF keyboard is the same as the [ENTER] on the 98203 keyboard. When you see [Return] in this guide, it means press [ENTER] on the 98203 keyboard.

Steps in Loading BASIC

Step 1: Turn the Computer Off

Locate the power switch.

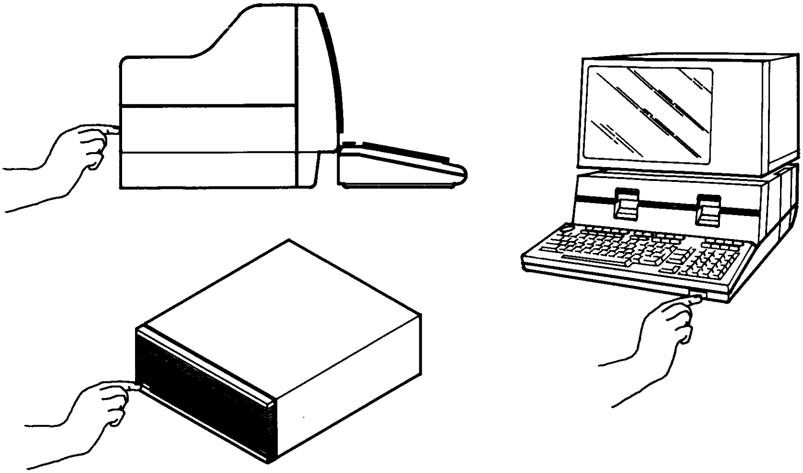


Figure 1-3. Locations of the Power Switch on Various Computers

If the computer is on, then press the switch to turn it off.

If the monitor is separate from the computer, then you should make sure it is turned **on** (to warm it up for subsequent use).

Step 2: Turn Your Disc Drive On

If you have BASIC in ROM, skip this step and go directly to **Step 5**. Ignore this step if loading from Model 226/236 built-in drives.

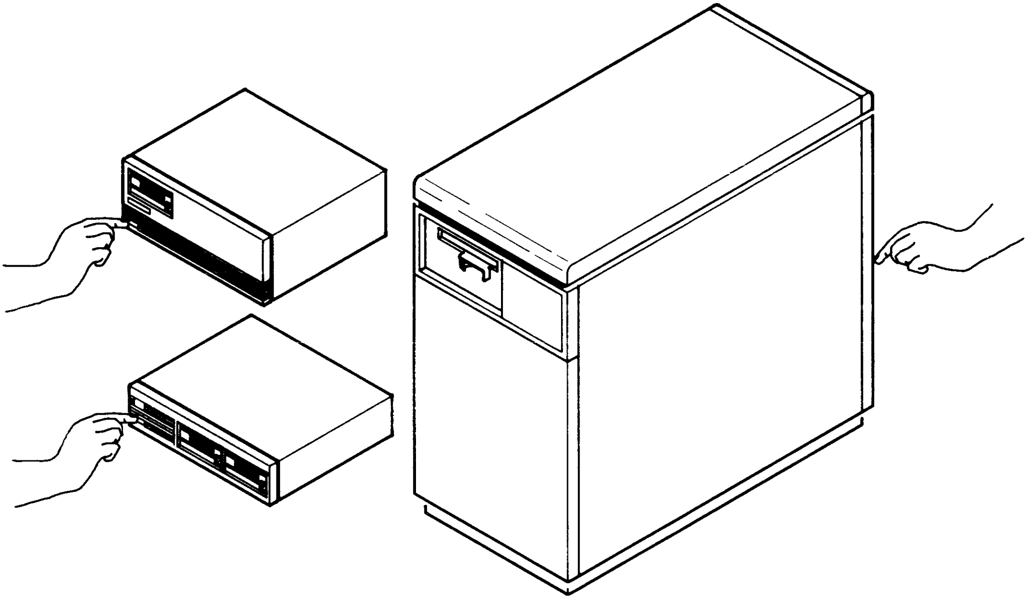


Figure 1-4. Turning On Your Disc

If you aren't sure where the power switch is on your disc drive, please refer to your drive's manual.

Step 3: Find the System "DISC ONE" and White Card

The BASIC System DISC ONE and white card are located in the plastic disc caddy located in the large box stamped **98616**; the box

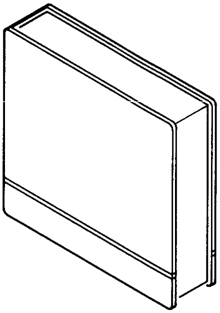


Figure 1-5. The System Disc and White Card Are in the Disc Box

Locate and remove this disc, as well as the white card and READ ME FIRST. You can follow the steps outlined in the READ ME FIRST and white card, or continue with the detailed steps in this chapter.

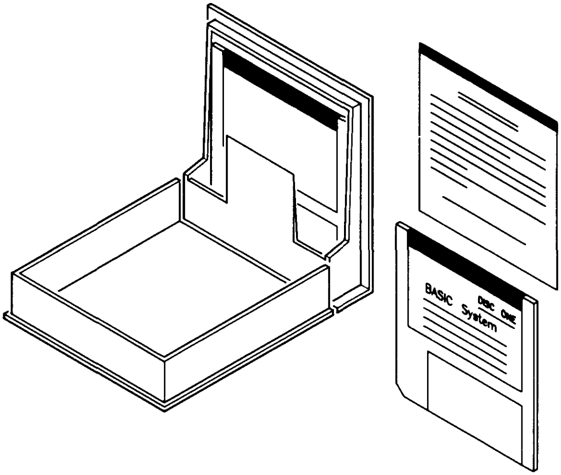


Figure 1-6. Remove the System Disc and White Card

Step 4: Insert the "DISC ONE" into a Drive

Slide the BASIC System DISC ONE into the disc drive as shown below.

- If using a **3½-inch "micro" disc**, the metal end should enter the drive first.

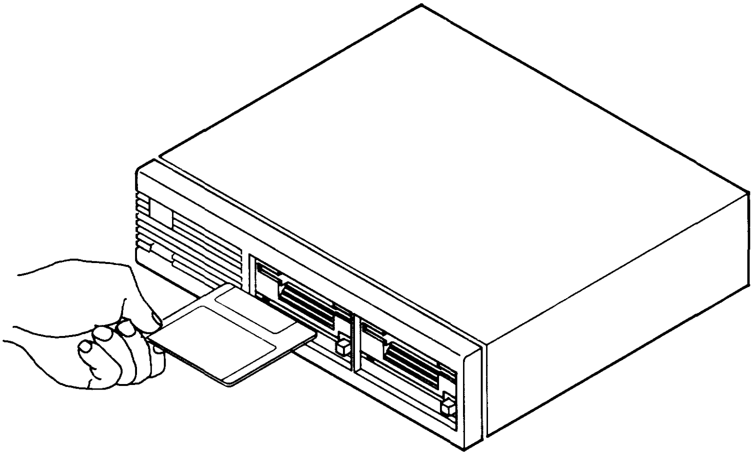


Figure 1-7. Inserting a 3½-inch "Micro" Disc

- If using a 5¼-inch **"mini" disc**, the media window should enter the drive first.

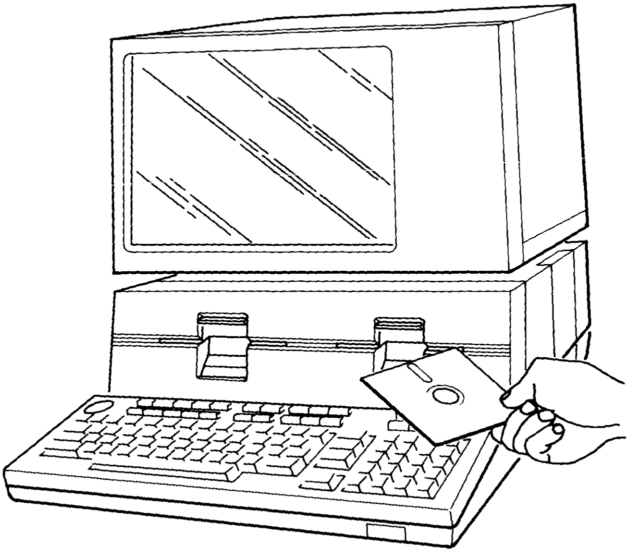


Figure 1-8. Inserting a 5¼-inch "Mini" Disc

- If using a **5¼-inch "mini" disc**, close the disc drive door.

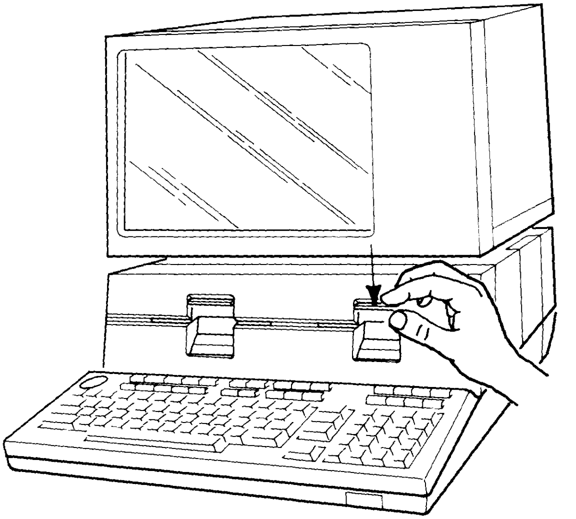


Figure 1-9. Closing the "Mini" Disc Drive Door

Step 5: Turn on the Monitor (If Separate from the Computer)

If the computer monitor is separate from the computer, and you didn't turn it on in Step 1, then turn it on now. This will give it a few moments to warm up before turning on the computer.

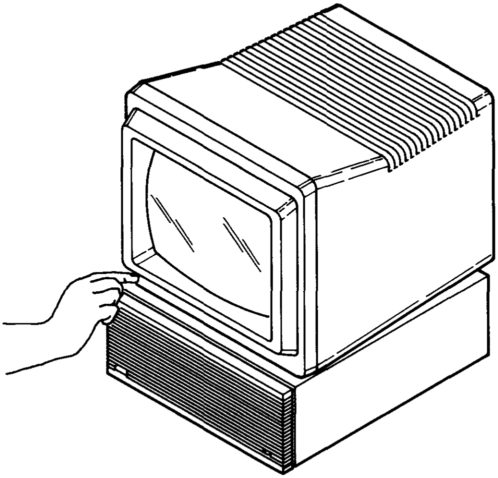


Figure 1-10. Turn Monitor On

(If the monitor is not separate from the computer, then continue on with the next step.)

Step 6: Turn on the Computer, and Hold Down the Space Bar (on the Keyboard)

If you have an empty hard disc, or there are **no** other on-line systems, you **need not** hold down the space bar. You can skip ahead to **Step 8**.

After pressing the computer's power switch (the same switch as in step 1), press the space bar on the keyboard and hold it down.

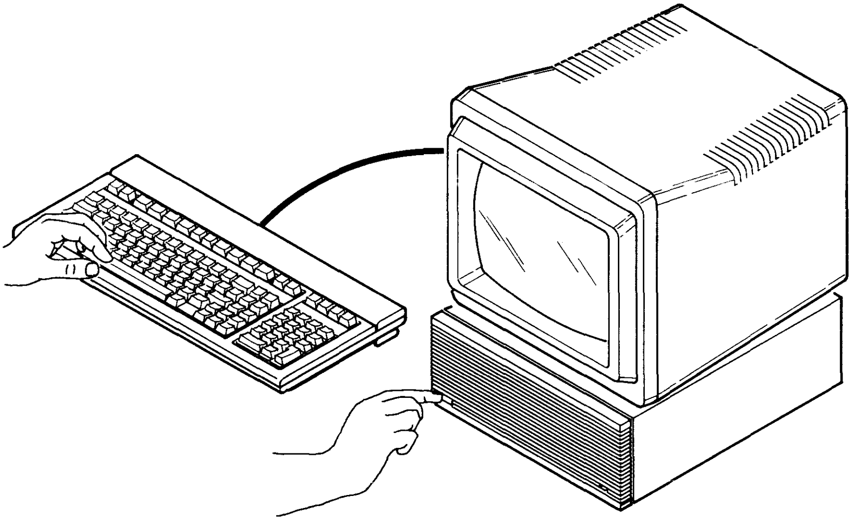
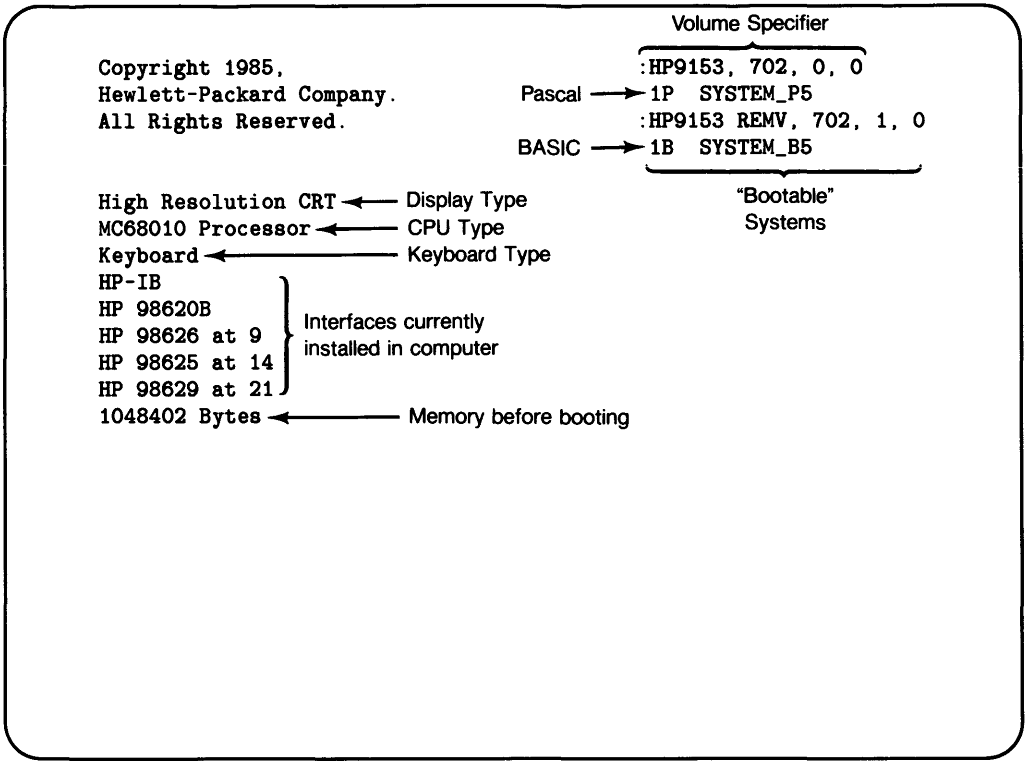


Figure 1-11. Turn Computer On and Hold Down the Space Bar

Holding down the space bar does two things:

- Tells the computer to wait until you are ready to load a system (instead of automatically loading the first one it finds).
- Allows you to watch the computer's display as it:
  - Performs its self-test (checks memory, etc.)
  - Searches for interfaces (such as HP-IB, RS-232C, etc.), and *on-line systems* (operating systems stored on discs, tapes, and ROM cards). For a list of the order in which the devices are search, see the last section in this chapter, "Order of Devices Searched by the Boot ROM".

The resultant display should look something like this (if not, see the next sections):



```
Copyright 1985,                              :HP9153, 702, 0, 0
Hewlett-Packard Company.                     1P  SYSTEM_P5
All Rights Reserved.                         :HP9153 REMV, 702, 1, 0
                                             1B  SYSTEM_B5

High Resolution CRT
MC68010 Processor
Keyboard
HP-IB
HP 98620B
HP 98626 at 9
HP 98625 at 14
HP 98629 at 21
1048402 Bytes
```

Figure 1-12. Display of Boot ROM 3.0 and Later Versions

If You Have an Earlier Boot ROM (Models 226 and 236 Only)

If your display looks like this, you have Boot ROM version 1.0 or 2.0. Please read the description below, and then skip to step 8.

```
        nnnnnn AVAILABLE BYTES
```

Figure 1-13. Display of Boot ROM 1.0 and 2.0

This is the amount of memory that the Boot ROM has found in the computer. (If it is less than you expected, you may need to check the switch settings on your memory cards. See the *Peripheral Installation Guide* for instructions.)

Unlike later Boot ROM's, these earlier Boot ROM versions will not display the results of the search for interfaces and mass storage devices.

Write Down the Memory Before Booting

Write the number you see at the lower left column on the boot screen. This is a number you may need to use later when checking memory requirements. It will help you greatly to write this number on the line below at this time:

____

If You See Nothing on the Monitor Screen

Here are some possible explanations:

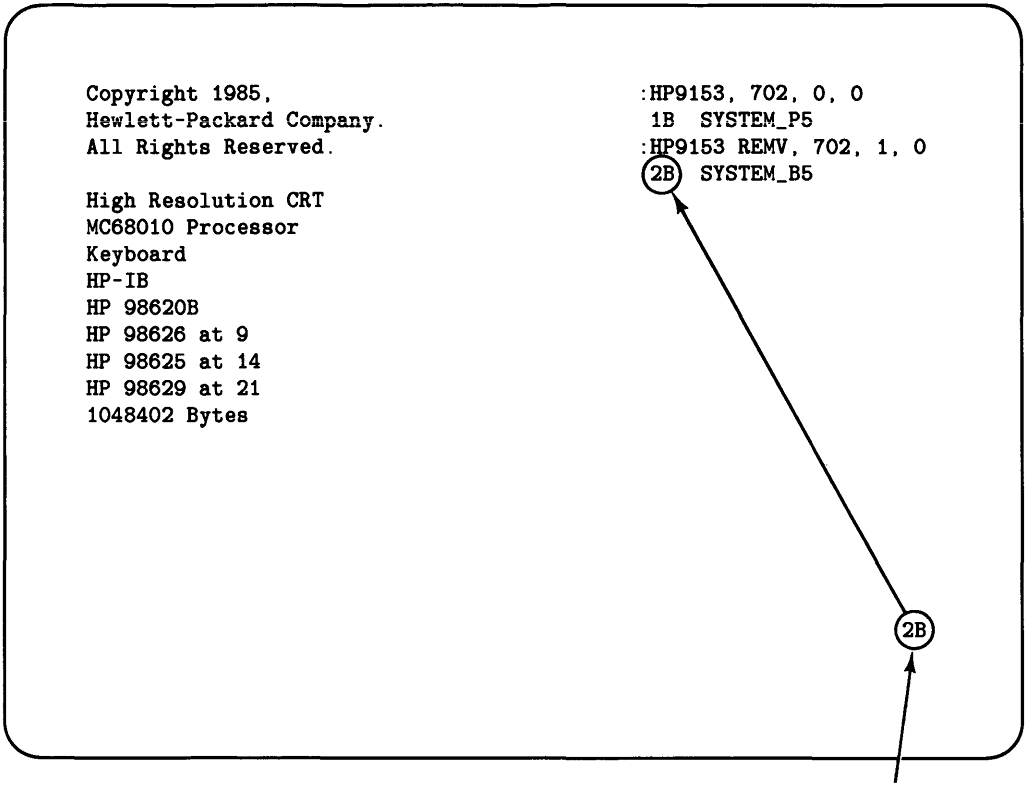
- The monitor's brightness is not turned all the way up.
- The monitor is not plugged in.
- The computer is not plugged in.
- Your eyes are not pointed in the right direction, or are obscured by your eyelids.

If You Still Have Problems

If you have verified that the above problems do not exist, but you are still not getting a display, see your computer's *Installation Reference* for troubleshooting advice.

Step 7: Choose a System to Load

You should now have a display something like this:



```
Copyright 1985,                            :HP9153, 702, 0, 0
Hewlett-Packard Company.                    1B  SYSTEM_P5
All Rights Reserved.                       :HP9153 REMV, 702, 1, 0
                                            2B  SYSTEM_B5
High Resolution CRT
MC68010 Processor
Keyboard
HP-IB
HP 98620B
HP 98626 at 9
HP 98625 at 14
HP 98629 at 21
1048402 Bytes
```

Figure 1-14. Choosing a System

Letters you type
will appear here

Example

To select a system, type in the letters to the left of the system file name. For instance, to select SYSTEM_B5 in the above example, type 2 B. (Don't be alarmed if the computer takes a few seconds to respond to your selection.)

The System's Response

The lower-left corner of the display will show the following message to indicate that it is loading the desired system:

```
BOOTING A SYSTEM
RESET To Power-up
```

After you see this message, you may move on to the next step.

Step 8: Follow the Instructions on the Screen

The computer should now be loading the BASIC system. You should follow the instructions displayed on the lower left of the screen.

Switching Discs, and Pressing the CONTINUE Key

The computer will prompt you to switch discs and press the CONTINUE key to signify that you have switched the disc.

If you have an ITF keyboard or keyboard that does not have a CONTINUE key, press the softkey which corresponds to the softkey label "Continue" (f2 on ITF keyboards). If you have trouble, see "Using the Softkeys" section at the beginning of this chapter or the "Introduction to the System" chapter.

When Prompted by the Computer, Write Your Volume Specifier on a Sticker

You may choose to write the volume specifier on the line below (ignore the quotes):

____

Then write the volume specifier on a sticker and affix it to the front of the default drive. You can have easy access to the volume specifier this way.

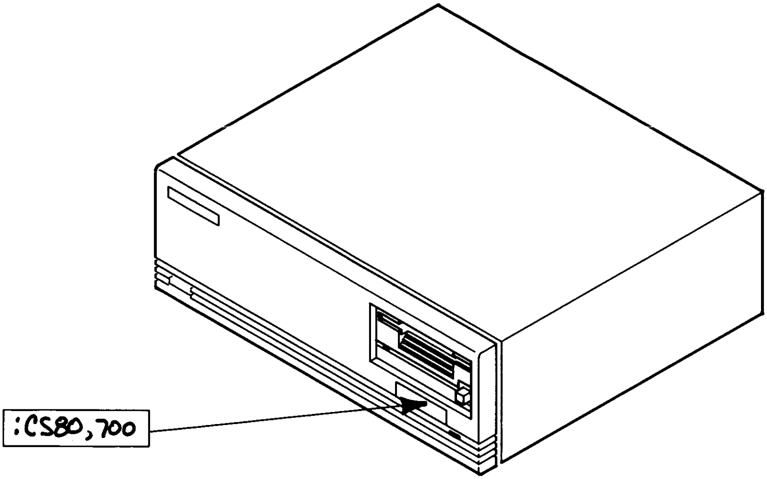


Figure 1-15. Place the Sticker on the Drive

If you have **only** one disc drive (or a dual-drive disc), then here is all you need to do:

a. If the VERIFY utility is currently running, stop the program by pressing [Reset] ([Shift]-[Break] or [SHIFT]-[PAUSE]).

b. Use the SYSTEM$ function to determine your current Mass Storage (MSI) device:

```
SYSTEM$("MSI")
```
[Return] or [ENTER]

The result displayed by the system is the volume specifier of your default drive. For instance:

```
:CS80,700,0
```

c. Write this on one of the stickers supplied with the system.

d. Peel the backing off of the sticker, and place it on the front of the disc cabinet near the drive it identifies.

e. If you have a dual drive, the other drive's volume specifier will have:
   - the same device type (such as CS80)
   - the same device selector (such as 700)
   - a *different* unit number (1 instead of 0).

   Here are the volume specifiers of a typical HP 9122D dual drive:

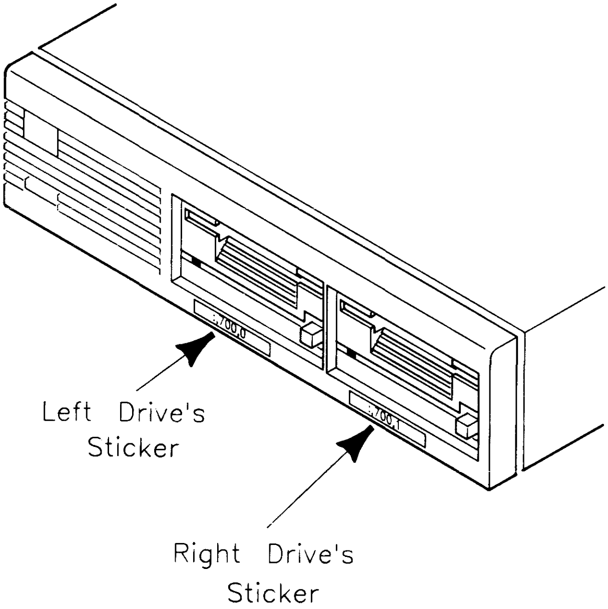


Figure 1-16. Stickers for an HP 9122D Dual Drive

Note you can always determine which device is the current *default volume* by executing this function:

```
SYSTEM$("MSI")  Return
```

This function is also available on a typing-aid softkey. See "Introduction to the System" chapter for details.

Is BASIC Loaded and Ready for Use?

If the following message appears at the bottom of the monitor screen, then BASIC has been successfully loaded:

```
The BASIC system is now loaded for your use.
```

If not, please turn the page for instructions about potential problems and solutions.

HP-UX Clock Compatibility

If you are sharing BASIC with an HP-UX system, you should execute the TIMEZONE command to set the correct time zone. For example, this is the command for Mountain Standard Time Zone:

```
TIMEZONE IS -7*60*60 Return
```

The value used for your local time zone may be different. For example: Pacific Standard Time Zone is -8*60*60, and Central Standard Time Zone is -6*60*60.

Now, execute this command to check the real-time clock:

```
TIME$(TIMEDATE), DATE$(TIMEDATE) Return
```

The result might look like:

```
09:00:37  16 Mar 1987
```

If the time is incorrect, set the time: see the *BASIC Language Reference* manual. For example:

```
SET TIMEDATE TIME("09:00:00")+ DATE("Feb 28 1987") Return
```

If Loading Fails

If the computer did **not** load the BASIC system and display this message:

```
The BASIC system is loaded for your use.
```

then you should check for the following potential problems.

Problem	Solution
The Boot ROM does not find the BASIC system.	Make sure that the disc drive is plugged in and turned on. Make sure that the drive is connected to the HP-IB interface of the computer. Make sure the *System* disc is properly inserted in the drive.
`NOT ENOUGH MEMORY` is displayed.	Your computer does not have sufficient memory, or memory board(s) have been improperly installed. Refer to the *Peripheral Installation Guide* for instructions on how to configure and install memory cards.

What To Do Next?

Refer back to the table in the Introduction to this section "Installing BASIC: Introduction".

Order of Devices Searched by the Boot ROM

The following table lists the order in which the Boot ROM searches mass storage devices for system files.

Priority	Type of Mass Storage Device	Comments
1	Internal disc drive (9826 and 9836 only)	Select code 4; unit number 0 (right drive)
2	External disc drives	Select codes 7 through 31; primary address 0; unit number 0; volume number 0
3	Shared Resource Manager (SRM)	Select code 21; node number 0; volume number 8; /SYSTEMS directory
4	Bubble memory card (HP 98259)	Select code 30 (Bubble memory is organized as files and treated the sameas other mass storage devices.)
5	EPROM card (HP 98255)	Unit 0 (EPROM is also organized and treated like other mass storage devices: if a system is found in EPROM, then it is first transferred to computer memory and executed from there—the CPU does not execute the system directly from EPROM)
6	ROM-based operating systems	Execute directly from ROM and do not have to be loaded into computer memory
7	Remaining external disc drives	Select codes 7 through 31; addresses 0 through 7; unit numbers 0 through 7; volume numbers 0 through 7
8	Remaining SRM systems	Select codes 8 through 31
9	Remaining bubble memory cards	Select codes 0 through 31, except 30
10	Remaining EPROM units	Units 1 through last one found

Notes

Verifying and Labeling Peripherals 2

Once you have successfully loaded your BASIC system, you may want to see if it can communicate with all of its peripherals. This chapter describes the tools provided for this task.

The most important task in this chapter is **labeling all of your mass storage devices**. All other tasks are optional for installing BASIC, and you may want to skip the other sections unless you encounter an error while trying to use one of your peripherals.

Prerequisites

You should have loaded ("booted") the BASIC system. If not, please refer to the "Loading BASIC" chapter.

Overview of Using the VERIFY Utility

The "VERIFY" Utility, located on the *BASIC Utilities* disc[1], allows you to programatically verify the operation of several common peripheral devices. The utility will also help you to label your devices, along with the explanations in this chapter.

Capabilities of this Utility

Using this utility will help you to perform the following tasks:

- List and label all **mass storage devices** currently connected to the computer
- List all **HP-HIL devices** currently connected to the computer
- Determine which **interface cards** are currently installed in the computer; determine whether or not the corresponding **driver** is also loaded; optionally load the driver, if not already loaded
- Verify and label a **printer**
- Verify and label a **plotter**
- Verify and label an **HP-IB graphics tablet**.

Loading and Running the Utility

After your BASIC system has completely booted, you can load and run the VERIFY utility. Follow this procedure:

1. Insert the *BASIC Utilities* 1 (*Utilities* with the double-sided 3½ inch discs) disc into the *default* drive (the drive you used when loading the BASIC system).
2. Run a `LOAD` statement. For instance, this statement loads the utility from the disc in the default drive:

   `LOAD "VERIFY"` [Return]

   or if you need to use the volume specifier:

   `LOAD "VERIFY:,702,1"` [Return]

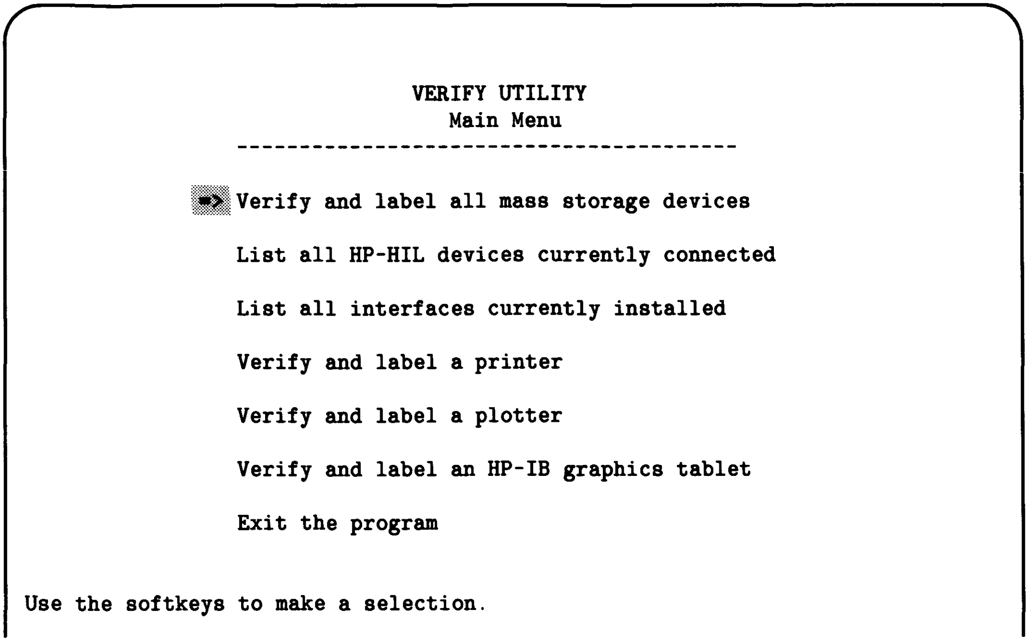
3. Run the program. Either type:

   `RUN` [Return]

   or press the [RUN] key ([f3] in the System menu of an ITF keyboard).

[1] With *single*-sided 3½-inch discs (option 042) and 5¼-inch discs (Option 044), the disc is labeled *BASIC Utilities 1*.

The utility shows the following display:



```
                          VERIFY UTILITY
                            Main Menu
                ----------------------------------------

              => Verify and label all mass storage devices

                 List all HP-HIL devices currently connected

                 List all interfaces currently installed

                 Verify and label a printer

                 Verify and label a plotter

                 Verify and label an HP-IB graphics tablet

                 Exit the program

Use the softkeys to make a selection.
```

Choosing Options in the Utility

You can select an option in the utility by taking these two steps:

1. Move the => cursor to point to the desired option by pressing the [▼] or [▲] cursor arrow keys (labeled [↓] or [↑] on some keyboards).
2. Press [Return] to select the option.

(Alternatively, you may also use the softkeys labeled **Next** or **Previous** to move the => cursor, followed by the softkey labeled `Select` to choose an option.)

Subsequent sections explain what each option does and what to do after you select the option.

Verifying and Labeling Mass Storage Devices

When you want to identify a particular disc drive, or any other mass storage device on the BASIC system, you will need to know its mass storage **volume specifier**. This section shows how to determine the volume specifier of each mass storage device in your system, and how to label and verify each. (If you want further explanation of mass storage devices, or have questions about the terms or concepts in this section, then please refer to the "Mass Storage Concepts" chapter.)

Example Volume Specifier

The volume specifier of any device is a combination of several pieces of information, which identify: the *interface* through which the device is connected to the computer, the *device's address* (if connected through an HP-IB interface), the number of *units* or *volumes* on the device (device-dependent).

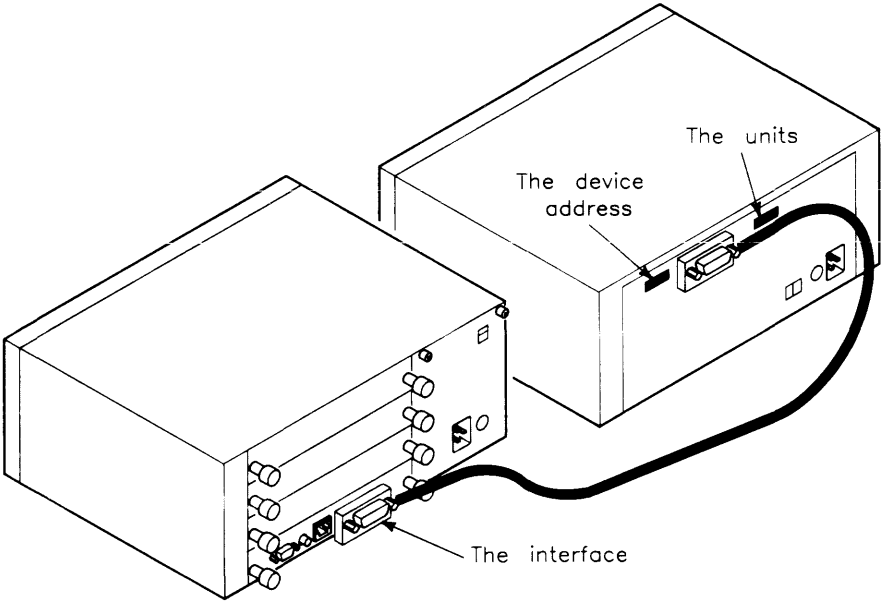


Figure 2-1. Volume Specifier Components

For instance, this is the volume specifier of the disc connected through the HP-IB interface at select code 7, and which has an address of 0 (the unit number of 1 usually specifies the right-hand drive):

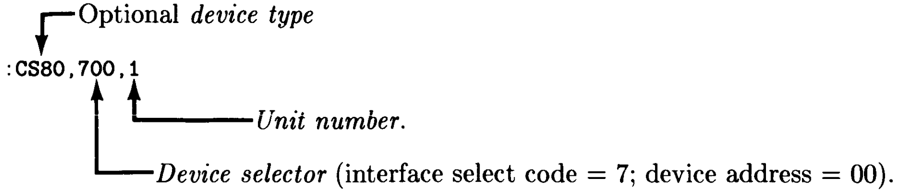
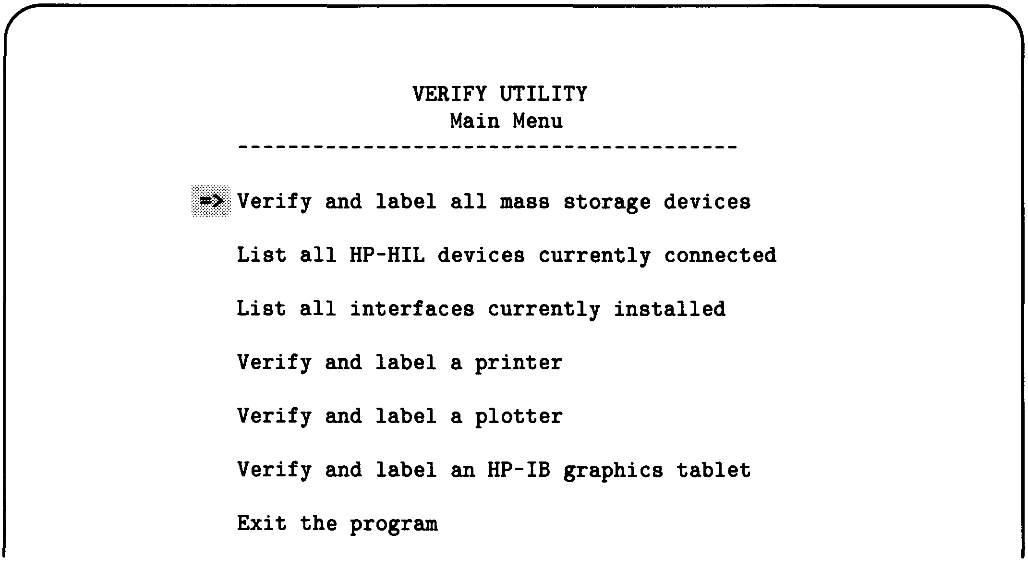

Optional *device type*
:CS80,700,1
Unit number.
Device selector (interface select code = 7; device address = 00).

Figure 2-2. Volume Specifier

Labeling Discs (Multiple Disc Drives)

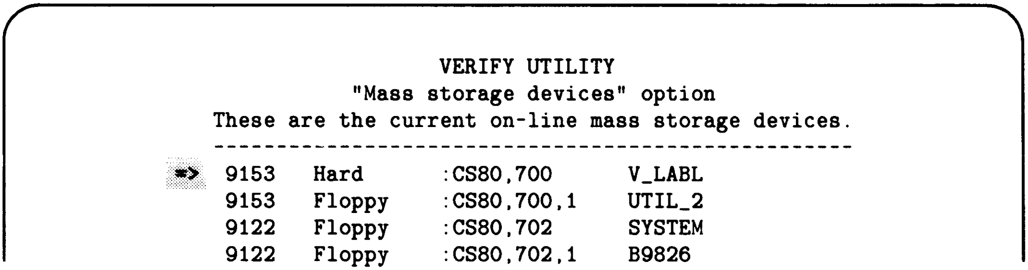
Here is a list of the steps you will take in labeling your discs.

1. From the main menu of the VERIFY utility, select the "`Verify and label all mass storage devices`" option. (If the utility is not currently running, press RUN to start it.)



```
                VERIFY UTILITY
                   Main Menu
   ----------------------------------------

=> Verify and label all mass storage devices

   List all HP-HIL devices currently connected

   List all interfaces currently installed

   Verify and label a printer

   Verify and label a plotter

   Verify and label an HP-IB graphics tablet

   Exit the program
```

This option of the VERIFY utility instructs the system to search all currently installed interfaces for the presence of mass storage devices. The resultant display lists the mass storage devices that it finds "on-line" (currently connected to your system, and operational). Here is a typical display produced by this option:



```
                            VERIFY UTILITY
                     "Mass storage devices" option
          These are the current on-line mass storage devices.
          ---------------------------------------------------
       => 9153   Hard       :CS80,700      V_LABL
          9153   Floppy     :CS80,700,1    UTIL_2
          9122   Floppy     :CS80,702      SYSTEM
          9122   Floppy     :CS80,702,1    B9826
```

The first column is the HP product number of each mass storage device. The second column shows the type of storage media used by the device (such as flexible or hard disc, tape, and magnetic-bubble memory). The third column identifies the hardware location of the device; you must use this to tell BASIC which device you want to use (without them, you cannot tell BASIC to use anything except the "default drive"). The last column is the "name" recorded on the volume (in the volume's directory). You can read it with the READ LABEL statement, and change it with the PRINT LABEL statement; see the "Using and Managing Files" chapter for details.

For further information about each of these parameters, see the "Mass Storage Concepts" chapter of this manual.

2. For each device listed on the screen:

    a. Determine which device the volume specifier identifies.

    b. Write the device's volume specifier on a sticker.

    HP ships several stickers with the BASIC system; they are in the box in which your BASIC system discs are packaged. You will be writing the volume specifiers of your mass storage devices on these stickers and then affixing them to the front of your devices' cabinets. Having this information readily available for each device will greatly simplify its subsequent use (you won't have to derive this information by looking at looking at interfaces and switches).

Copy the volume specifier onto the sticker as on the display. You may omit the "CS80" for it is optional. If you do, be sure to write everything else on the sticker, for example:

```
:,700,0
```

c. Peel off the backing on the sticker, and then place the sticker on the front panel of the device near the drive it specifies.

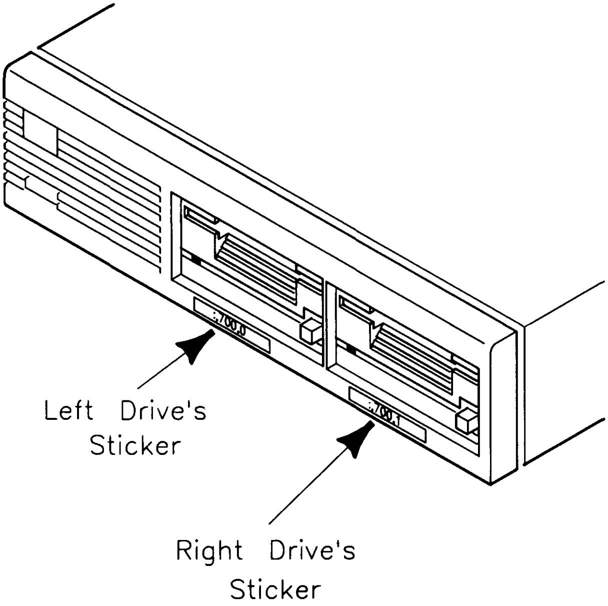


Figure 2-3. Putting the Sticker on the Mass Storage Device

With devices that contain both a flexible disc and a hard disc, place the flexible disc's volume specifier close to its drive door, and the hard disc's volume specifier on the other side.

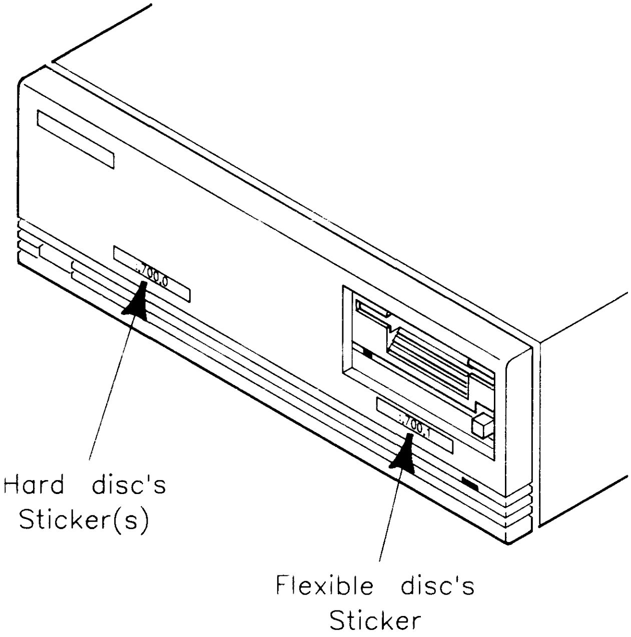
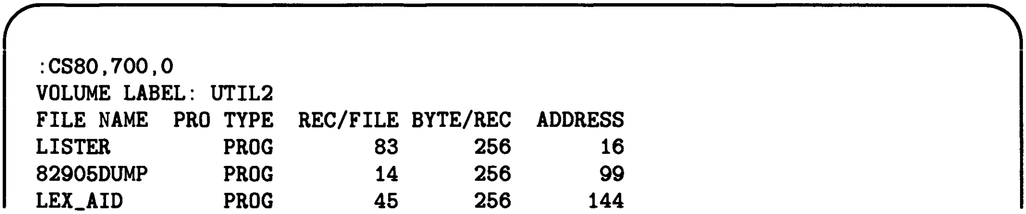


Figure 2-4. Dual Disc Drives

3. Once you have labeled all discs, you can verify the volume specifier by using each one in a CAT (disc catalog) statement. First, press Reset (Shift-Break on ITF keyboards), then type (for the default mass storage device):

   ```
   CAT Return
   ```



```
:CS80,700,0
VOLUME LABEL: UTIL2
FILE NAME  PRO TYPE  REC/FILE BYTE/REC  ADDRESS
LISTER         PROG        83      256       16
82905DUMP      PROG        14      256       99
LEX_AID        PROG        45      256      144
```

   To get a catalog of other mass storage devices, use the volume specifier following the CAT statement; for example for volume specifier ":CS80,700,1":

```
CAT ":,700,1"
```

   The "CS80" is omitted in this example to save you typing (remember it is optional). For a complete description of disc catalogs, see the "Using Files and Directories" chapter of this manual.

Verifying HP-HIL Input Devices

The "`List all HP-HIL devices currently connected`" option of the VERIFY utility lists all of the HP-HIL (Hewlett-Packard Human Interface Link) devices currently connected to your computer. Since you do not normally need to know each device's address[1], you will not need to label these devices.

Installing and Removing HP-HIL Devices

HP-HIL (Human Interface Link) devices connect to the computer through the HP-HIL interface. Normally you will be connecting these devices to the computer **before** booting the BASIC system. As BASIC is booted, the system recognizes which devices are installed.

You can also, however, install HP-HIL devices when the computer is running. However, in order for BASIC to recognize that it has been connected, you must execute this statement:

```
SCRATCH A  [Return]
```

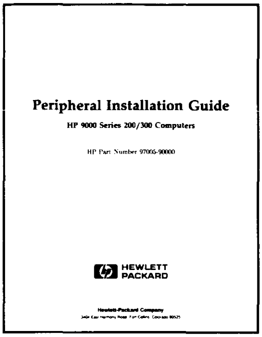
`SCRATCH A` performs a "re-configuration" of the link, after which the BASIC system recognizes (up to 7) HP-HIL devices.

Note

`SCRATCH A` will also delete the "VERIFY" program from memory, if currently loaded. If you want to use this program to list the HP-HIL devices currently linked to your system, then you will need to re-LOAD it before attempting to RUN it.

[1] The address of each HP-HIL device is determined by its relative position in the link. See the "HIL Devices" chapter of *BASIC Interfacing Techniques* for information on addressing and low-level access of each type of HP-HIL device.

1. Make sure that you have properly connected all HP-HIL devices to the computer. The installation instructions for each device describe the steps required; this information is also listed in the *Peripheral Installation Guide.*



Peripheral Installation Guide

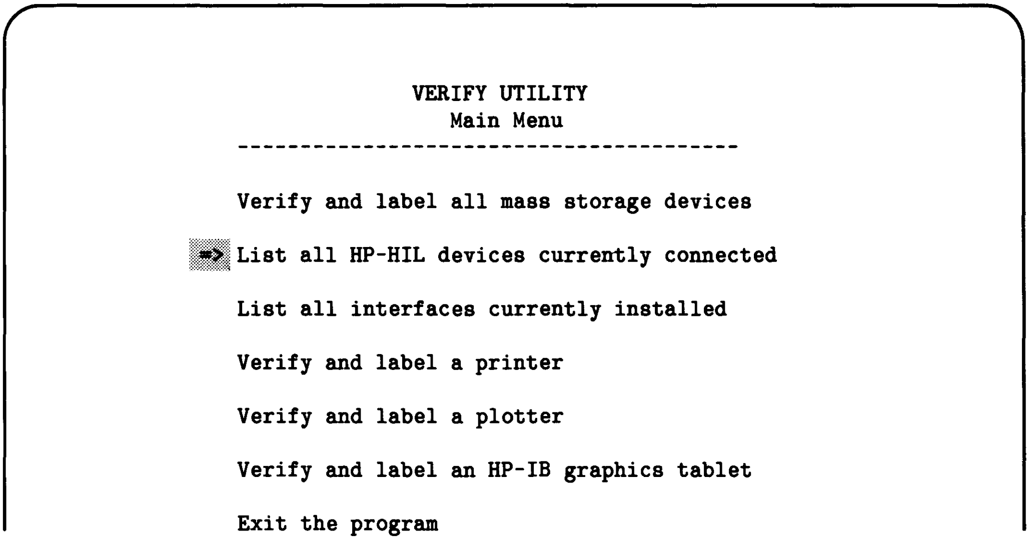
HP 9000 Series 200/300 Computers

HP Part Number 97005-90000

HEWLETT PACKARD

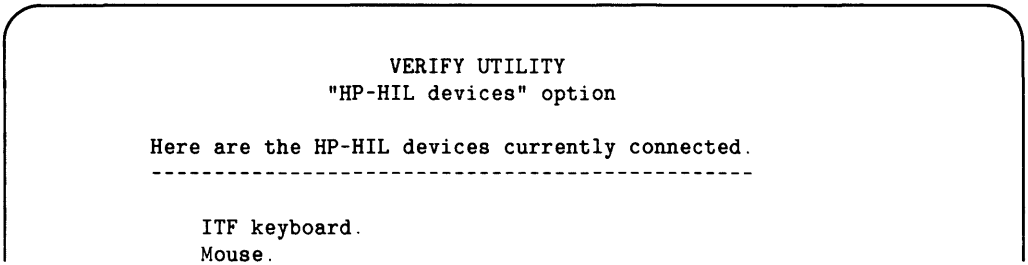
Hewlett-Packard Company
3404 East Harmony Road Fort Collins, Colorado 80525

2. Select the "`List all HP-HIL devices currently connected`" option from the main menu of the Verify utility.



```
                    VERIFY UTILITY
                       Main Menu
          ----------------------------------------

          Verify and label all mass storage devices

       => List all HP-HIL devices currently connected

          List all interfaces currently installed

          Verify and label a printer

          Verify and label a plotter

          Verify and label an HP-IB graphics tablet

          Exit the program
```

3. The program should then display something like the following list of HP-HIL devices:



```
                         VERIFY UTILITY
                      "HP-HIL devices" option

          Here are the HP-HIL devices currently connected.
          ------------------------------------------------

             ITF keyboard.
             Mouse.
```

In the above example, there is a keyboard at address 1, and a mouse at address 2. (Note most uses of these devices, you will not need to know the device's address.)

Select "Main Menu" to return to the Main Menu.

If the Verification Fails

Verify that you have:

- Loaded the KBD language extension file (execute a LIST BIN statement to check).
- Correctly connected all HIL devices to the computer. Make sure that you did not forget to plug in a device, or incorrectly plug one of the connectors with two dots (● ●) into a connector with one dot (●). See the *Peripheral Installation Guide* or your particular HIL device's installation instructions for further information.

Verifying Interfaces

The "`List interfaces currently installed`" option of the VERIFY utility lists the interfaces currently installed in your computer. It also displays whether or not the driver[1] for the interface is currently installed, and gives you the option of loading the driver for each interface.

Interfaces Recognized by the Utility

This utility recognizes and lists the following HP products:

HP Product Number	Interface Type
98253	EPROM Programmer Card
98259	Bubble Memory Card
98620	Direct-Memory Access (DMA) Card
98622	General-Purpose Input and Output (GPIO) Interface
98623	Binary-Coded Decimal (BCD) Interface
Built-in and 98624	Hewlett-Packard Interface Bus (HP-IB) Interface (IEEE 488-1978)
98625	High-Speed (Fast) Disc HP-IB Interface
Built-in and 98626	RS-232C Serial Interface
98627	Color Video Output (RGB) Interface
98628	Datacomm Interface
98629	Shared Resource Manager (SRM) Interface
98635	Floating-Point Math Card
98640	Analog-To-Digital Converter Card
98643	Local-Area-Network (LAN) Interface (not supported by the BASIC operating system)
Built-in and 98644	Low-Cost RS-232C Serial Interface
98646	VME card (not supported by the BASIC operating system; looks line two 'non-HP' cards at select codes 15 and 16)

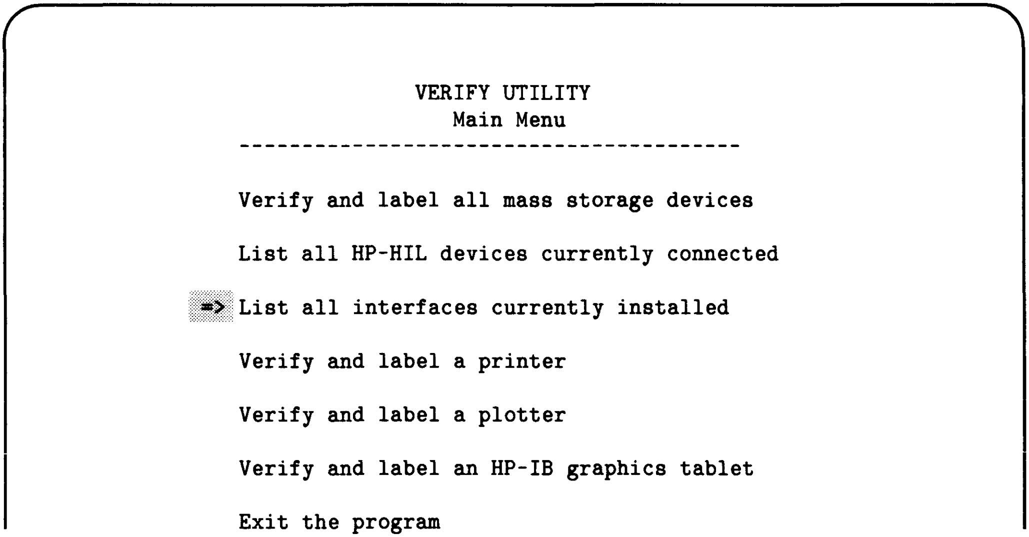
[1] A driver is a routine used by the BASIC system to "talk to" interfaces and devices. Thus, if you will be using an interface to talk to a device connected to the computer, then you will need to load the corresponding driver.

Listing All Interfaces and Interface Drivers Currently Installed

Use the following steps to list all interface cards (and corresponding interface drivers) currently installed in your computer.

1. Make sure that all cards are properly configured and installed in your computer. (Make sure that the power is turned **off** before you do this!) Instructions for configuring and installing cards are given in the "Accessory Cards" and "Interface Cards" chapters of the *Peripheral Installation Guide.*

2. From the main menu of the VERIFY utility, select the "`List interfaces currently installed`" option.



```
                    VERIFY UTILITY
                       Main Menu
          ----------------------------------------

          Verify and label all mass storage devices

          List all HP-HIL devices currently connected

       => List all interfaces currently installed

          Verify and label a printer

          Verify and label a plotter

          Verify and label an HP-IB graphics tablet

          Exit the program
```

The program then search the backplane for interface cards currently installed. Here is a typical display produced by selecting this option:

```
                         VERIFY UTILITY
                       "Interfaces" option

                Here are the interfaces currently installed.
                --------------------------------------------

Interface       Product         Select      Binary      Currently
  Type          Number           Code       Name         Loaded?
-------      -------------      -------     -------     ----------
HPIB         Built-in              7        HPIB           Yes
Serial       Built-in              9        SERIAL         Yes
GPIO         98622                12        GPIO           Yes
```

3. For each of the interfaces listed in the preceding display, the utility shows whether or not the corresponding driver is "currently loaded."

   If there are any missing binaries (that is, if the driver for an interface which requires a driver is not present), then the program will display the following question:

```
  Want to load missing binaries? (Type 'Y' or 'N')
```

You have two answers to this question:

- If your answer is N, the program terminates.
- Any other answer will be interpreted as "Yes", and the utility will prompt:

```
  Insert the 'BASIC Drivers' disc (or enter volume specifier) & press 'Return'
```

You have three choices in responding to this prompt:

- If the binaries **are** on the *current default volume*, just press Return. The program will then load the binaries from this volume.
- If the binaries are **not** on the *current default volume*, insert the 'BASIC Drivers' disc into the default drive and press Return. The program will then load the binaries from this volume.
- If the binaries are in a directory **other than** the current working directory (of an HFS or SRM volume), then direct the program to the directory and/or volume containing the binaries by specifying the directory and/or volume on which they reside. For example:

  ```
  /WORKSTATIONS/BIN5.0 :,700
  ```

  The program will attempt to load the driver binaries from this directory and volume.

At this point, the program will go through all the list of "missing binaries" and ask if you want to load each one.

Note

If you answer "Yes" to the prompt for the SRM binary, the program will automatically load the DCOMM binary (because the SRM binary is not usable without the DCOMM binary).

Other Interface Utilities

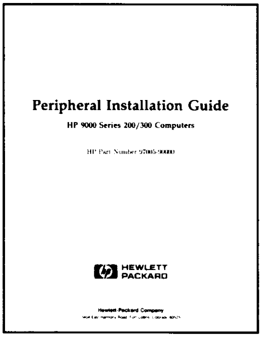
There are also additional utility programs, described in the *BASIC Utilities* manual, that allow you to programatically show the configuration of the following interfaces:

- HP-IB
- GPIO
- Serial

Verifying and Labeling Printers

The default *device selector* assumed by the BASIC system is 701. This value indicates that the system expects the printer to be connected to the built-in HP-IB interface, and expects the printer's address to be set to 01. This verification procedure shows you how to verify this assumption, or change it if necessary, so that you can print a message on the printer to verify that it is working properly.

1. Connect the printer to your computer, and make sure that its power is turned on. (Instructions for configuring and connecting printers are given in the "Printers" section of the *Peripheral Installation Guide.*)



Peripheral Installation Guide

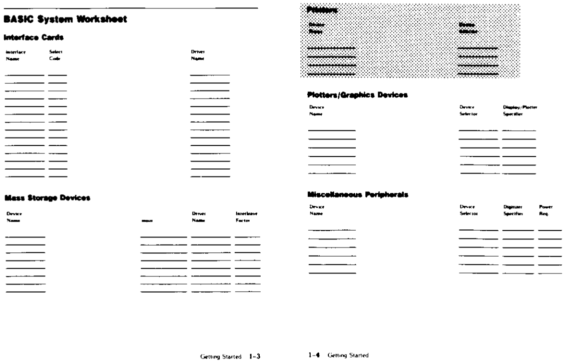
HP 9000 Series 200/300 Computers

HP Part Number 97005-90000

HEWLETT PACKARD

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

2. Look up the device selector of your printer in the "Printers" section of the BASIC System Worksheet (which is in the *Peripheral Installation Guide*).



BASIC System Worksheet

Interface Cards

Interface Name | Select Code | Driver Name

Mass Storage Devices

Device Name | msus | Driver Name | Interleave Factor

Getting Started 1-3

Printers

Device Name | Device Selector

Plotters/Graphics Devices

Device Name | Device Selector | Display/Plotter Specifier

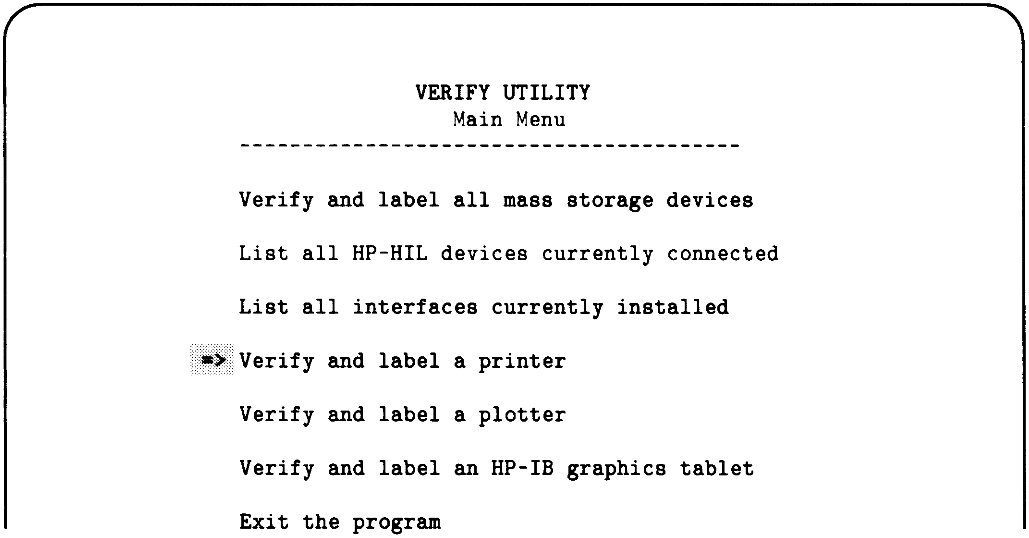
Miscellaneous Peripherals

Device Name | Device Selector | Digitizer Specifier | Power Req.

1-4 Getting Started

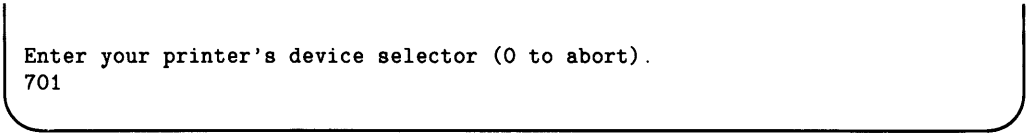
If you have not completed this worksheet, then you may attempt to use the default device selector (701) supplied by the VERIFY utility. (If that doesn't work, you should turn to the *Peripheral Installation Guide* for further instructions.)

3. From the main menu of the VERIFY utility, select the "**Verify and label a printer**" option.



```
                      VERIFY UTILITY
                        Main Menu
          ----------------------------------------

          Verify and label all mass storage devices

          List all HP-HIL devices currently connected

          List all interfaces currently installed

        => Verify and label a printer

          Verify and label a plotter

          Verify and label an HP-IB graphics tablet

          Exit the program
```

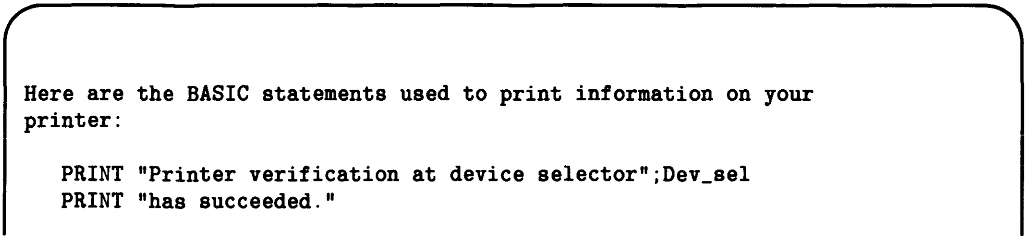
4. The program will prompt with a request for you to verify the *device selector* of your printer.



```
Enter your printer's device selector (0 to abort).
701
```

If your printer is at device selector 701 (or if you are not sure what it is), merely press Return. **If you wish to abort back to the main menu at this time, clear the** "701" **and enter** 0. Otherwise change the 701 to the appropriate value and press Return.

5. The utility then attempts to print the following message on your display (top message) and printer (bottom message):



```
Here are the BASIC statements used to print information on your
printer:

   PRINT "Printer verification at device selector";Dev_sel
   PRINT "has succeeded."
```

```
Printer verification at device selector 701
has succeeded.
```

If the Printer Verification Fails

Check that:

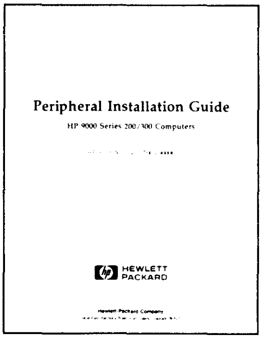
- You typed the correct device selector
- The printer is turned on and connected to the computer
- The printer's "on-line" light is on (if it has one)
- The HP-IB primary address is set correctly (if using an HP-IB plotter)
- The device selector is recorded correctly on the BASIC System Worksheet (see the *Peripheral Installation Guide*)
- You have loaded the drivers required to access the printer and its interface (type LIST BIN to check)

Verifying and Labeling Plotters

There is no "default plotter" assumed by the BASIC system. Therefore, you will need to use the PLOTTER IS statement to specify which device you want to use as a plotter.

The usual device selector used with the BASIC system is 705. This value indicates that the system expects the plotter to be connected to the built-in HP-IB interface, and expects the plotter's address to be set to 05 (00101 on the switch settings). This verification procedure shows you how to verify this assumption, or change it if necessary, so that you can send some graphics to your plotter to verify that it is working properly.

1. Connect the plotter to your computer, and make sure that its power is turned on. Instructions for configuring and connecting plotters are given in the "Plotters" section of the *Peripheral Installation Guide.*



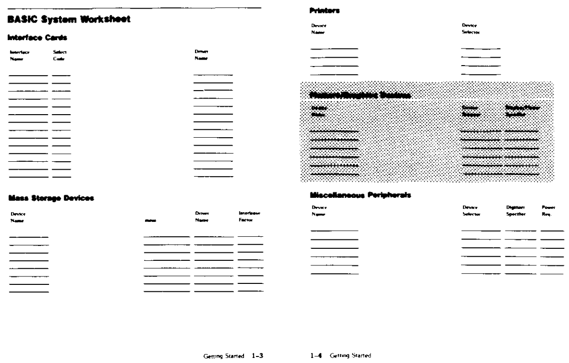
Peripheral Installation Guide

HP 9000 Series 200/300 Computers

HEWLETT PACKARD

Hewlett-Packard Company

2. Look up the device selector of your printer in the "Plotters/Graphics Devices" section of the BASIC System Worksheet (which is in the *Peripheral Installation Guide*).



BASIC System Worksheet

Interface Cards

Interface Name | Select Code | Driver Name

Mass Storage Devices

Device Name | msus | Driver Name | Interleave Factor

Getting Started 1-3

Printers

Device Name | Device Selector

Plotters/Graphics Devices

Device Name | [illegible] | [illegible]

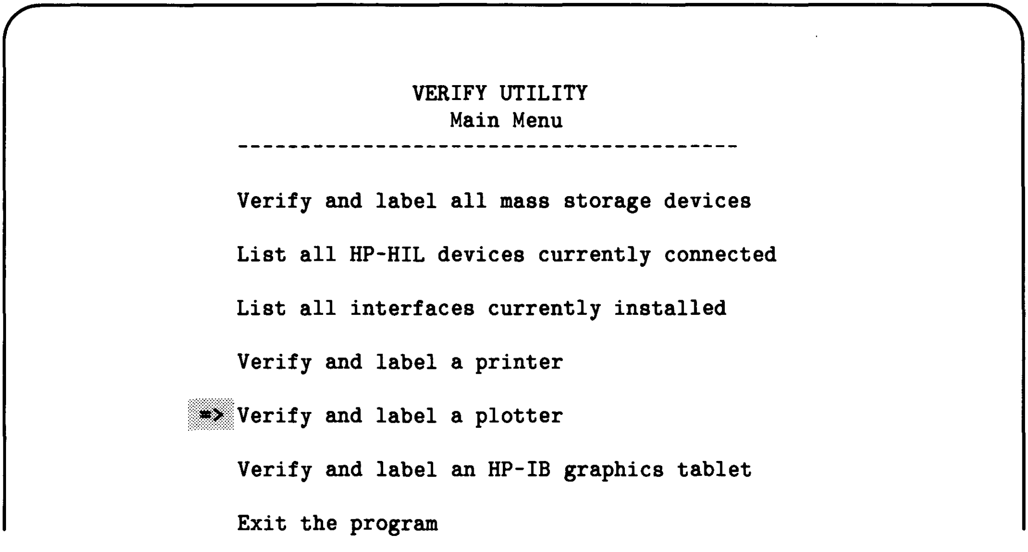
Miscellaneous Peripherals

Device Name | Device Selector | Digitizer Specifier | Power Req.

1-4 Getting Started

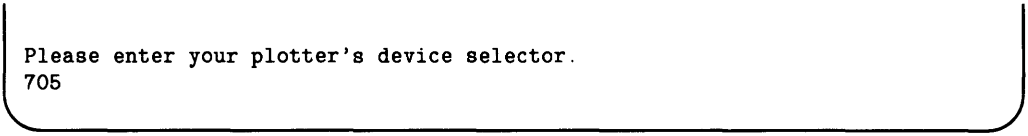
If you have not completed this worksheet, then you may attempt to use the default device selector (705) supplied by the VERIFY utility. (If that doesn't work, you should turn to the *Peripheral Installation Guide* for further instructions.)

3. From the main menu of the VERIFY utility, select the "`Verify and label a plotter`" option.



```
                    VERIFY UTILITY
                      Main Menu
        ----------------------------------------

        Verify and label all mass storage devices

        List all HP-HIL devices currently connected

        List all interfaces currently installed

        Verify and label a printer

     => Verify and label a plotter

        Verify and label an HP-IB graphics tablet

        Exit the program
```

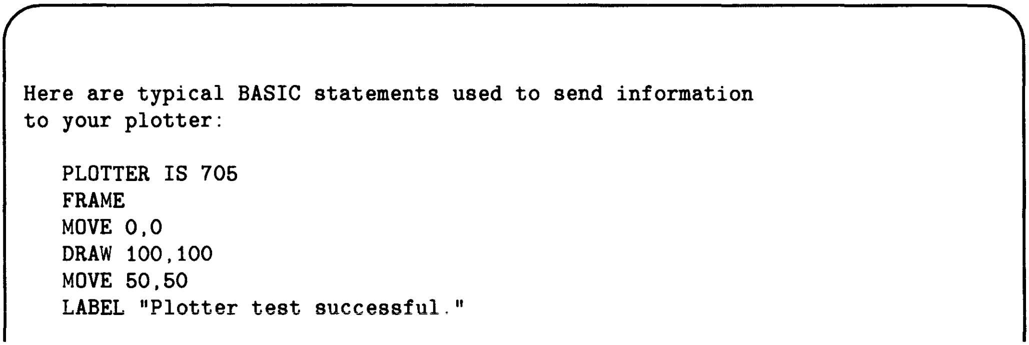
4. The program will prompt with a request for you to verify the *device selector* of your printer.



```
Please enter your plotter's device selector.
705
```

If your plotter is at device selector 705 (or if you are not sure what your plotter's device selector is), merely press [Return]. Otherwise enter the proper device selector and press [Return].

5. The utility then attempts to print the following message on your display (top message) and plotter (bottom message):



```
Here are typical BASIC statements used to send information
to your plotter:

   PLOTTER IS 705
   FRAME
   MOVE 0,0
   DRAW 100,100
   MOVE 50,50
   LABEL "Plotter test successful."
```

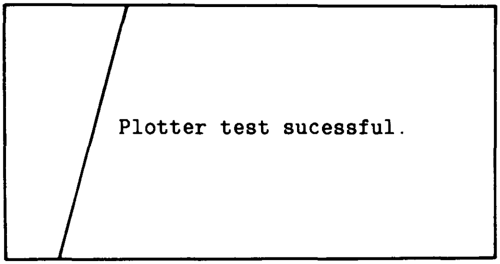


Figure 2-2. Plotter Verification

If the Plotter Verification Fails

Check that:

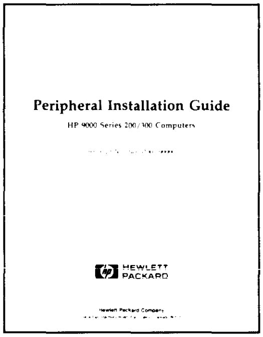
- You typed the correct device selector
- The plotter is turned on and connected to the computer
- The plotter's "Chart Load" light is **not** on (if it has one)
- The HP-IB primary address is set correctly
- The device selector is recorded correctly on the BASIC System Worksheet (see the *Peripheral Installation Guide*)
- You have loaded the drivers required to access the plotter and its interface (type LIST BIN to check)

Verifying and Labeling HP-IB Graphics Tablets

There is no "default graphics input device" assumed by the BASIC system. Therefore, you will need to use the GRAPHICS INPUT IS statement to specify which device you want to use for input.

The usual device selector used with the BASIC system is 706. This value indicates that the system expects the tablet to be connected to the built-in HP-IB interface, and expects the device's address to be set to 06 (00110 on the switch settings). This verification procedure shows you how to verify this assumption, or change it if necessary, so that you can send some graphics to your plotter to verify that it is working properly.

1. Connect the tablet to your computer, and make sure that its power is turned on. (Instructions for configuring and connecting HP-IB graphics tablets are given in the "Graphics Devices" section of the *Peripheral Installation Guide.*)



Peripheral Installation Guide

HP 9000 Series 200/300 Computers

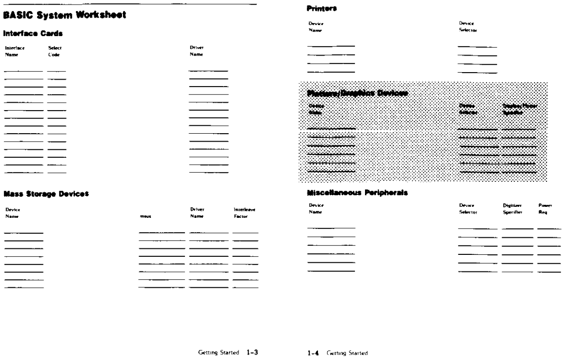
[illegible]

HEWLETT PACKARD

Hewlett Packard Company

[illegible]

2. Look up the device selector of your tablet in the "Plotters/Graphics Devices" section of the BASIC System Worksheet (which is in the *Peripheral Installation Guide*).



BASIC System Worksheet

Interface Cards

Interface Name	Select Code	Driver Name

Mass Storage Devices

Device Name	msus	Driver Name	Interleave Factor

Getting Started 1-3

Printers

Device Name	Device Selector

Plotters/Graphics Devices

Device Name	Device Selector	[illegible]

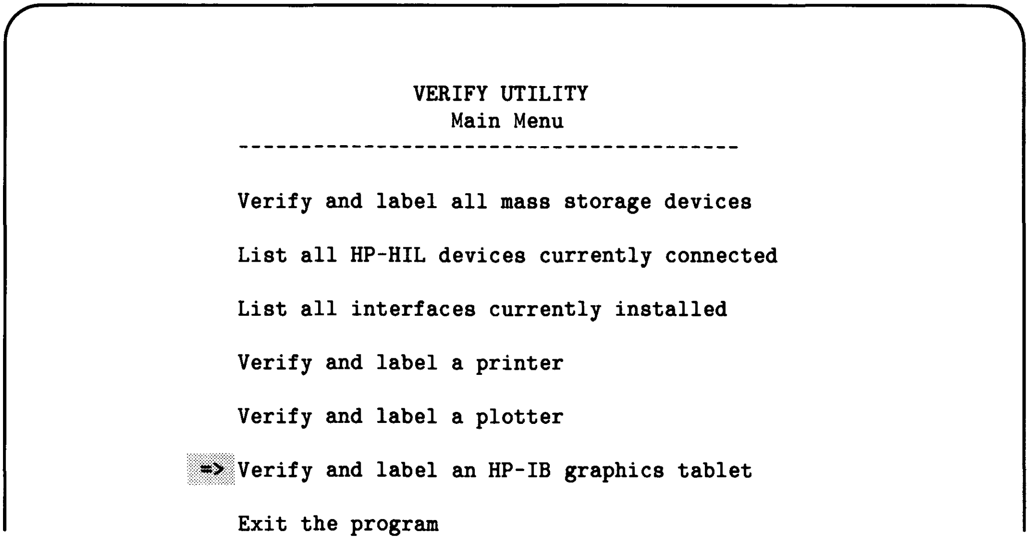
Miscellaneous Peripherals

Device Name	Device Selector	Digitizer Specifier	Power Req

1-4 Getting Started

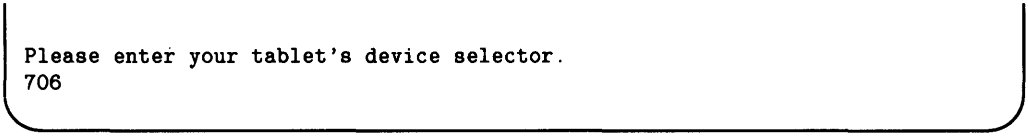
If you have not completed this worksheet, then you may attempt to use the default device selector (706) supplied by the VERIFY utility. (If that doesn't work, you should turn to the *Peripheral Installation Guide* for further instructions.)

3. From the main menu of the VERIFY utility, select the "**Verify and label an HP-IB graphics tablet**" option.



```
                    VERIFY UTILITY
                      Main Menu
        ----------------------------------------

        Verify and label all mass storage devices

        List all HP-HIL devices currently connected

        List all interfaces currently installed

        Verify and label a printer

        Verify and label a plotter

      => Verify and label an HP-IB graphics tablet

        Exit the program
```

4. The program will prompt with a request for you to verify the *device selector* of your tablet.



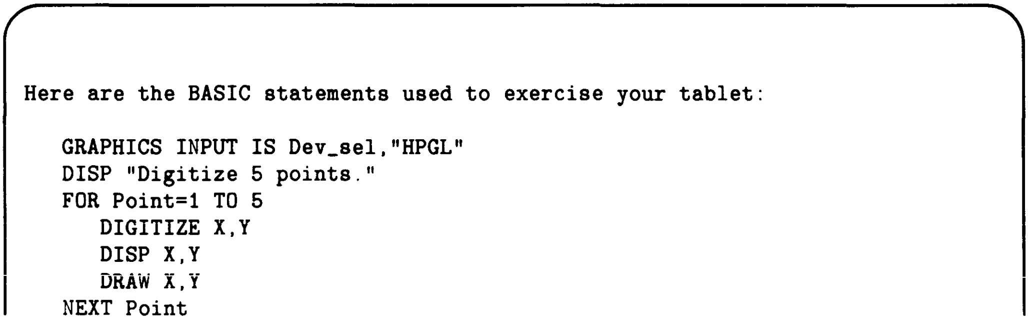
```
Please enter your tablet's device selector.
706
```

If your printer is at device selector 706 (or if you are not sure what your tablet's device selector is), merely press [Return]. Otherwise change the 706 to the appropriate value and press [Return].

5. The utility then prints:

```
Digitize 5 points.
```

You need to plot 5 points on the graphics tablet with your graphics stylus. You will see lines drawn on the screen and then (after the fifth point):



```
Here are the BASIC statements used to exercise your tablet:

   GRAPHICS INPUT IS Dev_sel,"HPGL"
   DISP "Digitize 5 points."
   FOR Point=1 TO 5
      DIGITIZE X,Y
      DISP X,Y
      DRAW X,Y
   NEXT Point
```

To return to the Main Menu, press **MainMenu**.

If the HP-IB Graphics Tablet Verification Fails

Check that:

- You typed the correct device selector
- The tablet is turned on and connected to the computer
- The tablet's "On" light is **not** on (if it has one)
- The HP-IB primary address is set correctly
- The device selector is recorded correctly on the BASIC System Worksheet (see the *Peripheral Installation Guide*)
- You have loaded the HPIB, GRAPH, and GRAPHX binaries (use LIST BIN to check)

What To Do Next

Read the next chapter, "Mass Storage Concepts", before continuing with the installation of BASIC (you can install BASIC without knowing all of the concepts in this chapter, but it is recommended you understand them before continuing with the "Formatting Flexible Discs," "Preparing Hard Discs," or "Putting BASIC on the SRM" chapters.) If you are in a hurry to install BASIC, go back to the table in "Installing BASIC: Introduction" to see what your next step is.

Mass Storage Concepts 3

This chapter describes the underlying concepts involved in mass storage operations on this BASIC system. **It is intended to support the decisions that you will be making while performing the tasks in the following chapters.** Therefore, you may choose to **skip** this chapter and move on to the following task-oriented chapters; those chapters will refer you to the relevant section(s) of this chapter when a detailed understanding of this mass storage system is required. For instance, while preparing a disc for use with BASIC you will need to choose between LIF and HFS directory structures. While formatting a disc, may wish to become familiar with the information in the section of this chapter called "Directory Structures Available".

Overview of Mass Storage Organization (or "What Are Files and Volumes?")

As the term "mass" suggests, mass storage devices are designed to store large quantities of data. Just how much data constitutes a large amount depends on the device itself. Most mass storage devices are capable of storing on the order of hundreds of thousands to several million items.

Mass storage devices are usually a "secondary" storage media, the "primary" media being computer memory. Because of this secondary nature, they are not generally expected to be as fast in accessing their data as primary storage.

Besides having the ability to store data, mass storage devices are capable of providing means for keeping data organized so that logical groups may be accessed systematically and efficiently. The information on mass storage is organized into files and volumes.

Files Are Named Collections of Data

One of the simplest ways to describe files and volumes is to use an analogy. A file is like a set papers on which information is written; we'll compare it to a tabbed chapter of this manual.

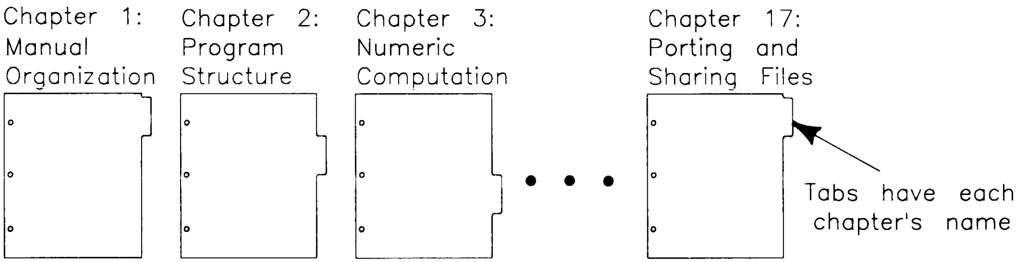

Chapter 1: Manual Organization
Chapter 2: Program Structure
Chapter 3: Numeric Computation
Chapter 17: Porting and Sharing Files
Tabs have each chapter's name

Figure 3-1. A File is Like a Tabbed Manual Chapter

Table 3-1. File Comparison

Attribute	Chapter	File
Type of information:	Printed characters, numbers, or graphics	Magnetic patterns (which represent characters, numbers, dots on screen, etc.)
Accessed method:	Use tab/chapter title	Use file name

Like pages in a chapter, files may contain BASIC programs or more general data—information to be used by a program, such as numbers and/or text. The name on the tab helps you locate the chapter, as well as helps you decide whether or not you want to look in the chapter.

Volumes Are Collections of Files

A volume is merely a collection of files. To continue our analogy, a volume is like a binder that contains several tabbed chapters:

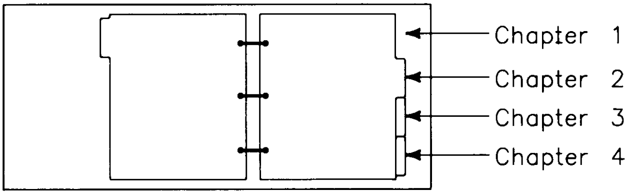


Figure 3-2. A Volume is Like a Binder Containing Tabbed Chapters

Table 3-2. Volume Comparison

Attribute	Book	Volume
Contains:	Chapters	Files
How contents listed:	Table of contents	Directory

Examples of Mass Storage Volumes

A flexible disc is typically one volume, as is a tape cartridge. Each of these types of *mass storage media* can hold several files.

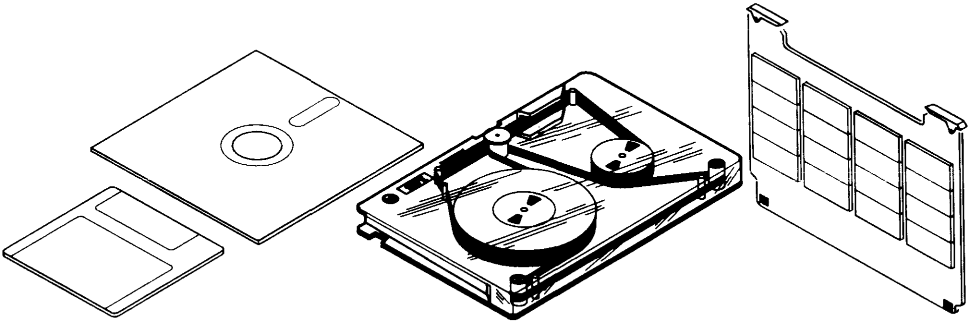


Figure 3-3. Examples of Mass Storage Volumes

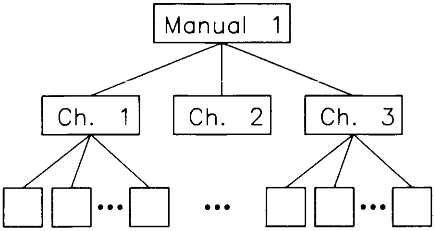
Hard discs, on the other hand, may be organized as one volume or as several independent volumes. We'll talk more about that subject later (in the section called "Hard Disc Partitioning").

Some volumes have *hierarchical* organizations. Here are two examples of hierarchies:

Textual Representation

```
------------------------------
BASIC  Programming  Techniques

    Chapter 1: Manual Overview
        Manual Organization
        What's in this Manual

    Chapter 2: Program Structure
        The Program Counter
        Sequence
              Linear Flow
              Halting Program Execution
              Simple Branching
        Selection
              :
              :
        Program-to-Program Communications
    Chapter 3: Numeric Computation
              :
              :
    Chapter 17: Porting and Sharing Files
```

Graphic Representation

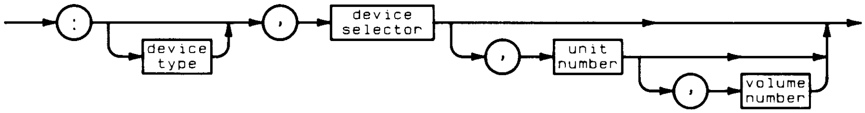


These two representations are two ways of showing the same type of hierarchical or superior/subordinate structure. BASIC also supports this hierarchical system, as described in the subsequent section called "Hierarchical Directories".

How to Specify Volumes

While it is easiest to use the default volume, there are times when you will want to use another mass storage device. In such cases, you will need to know how to specify the other volume. The way to specify mass storage devices is to use a *volume specifier*. If you read the "Verifying and Labeling Devices" chapter and followed the instructions, you should already have the volume specifier of each of your mass storage devices written on a label which you placed on the device's front panel.

Here is the syntax of a *volume specifier*:



Examples:

```
:CS80,700
:,700

:HP9122,702
:,702,1

:INTERNAL,4
:,4,1
```

This specifier consists of the following parts:

Table 3-3. Volume Specifier Components

Component	Explanation
A colon (:)	begins the volume specifier, separating the file name from the volume specifier (note that it is required even when you are performing an operation that only makes sense or a volume—for instance, INITIALIZE.)

Volume Specifier Components (Continued)

Component	Explanation
Device type	identifies the mass storage device's type. Once the BASIC system determines the device type, it can also determine the capacity of the device, its directory structure, and other information required to determine the access method for the device. Here are some examples: *DeviceType* — *DescriptionofMassStorageDevice* *omitted* — BASIC interrogates the device for its type. `HP9122` — HP 9122 3½-inch, flexible-disc drive. `HP9133` — HP 9133 Winchester (hard) disc drive (with optional 3½-inch, flexible-disc drive). `CS80` — any disc in the general class of "Command Set/80" devices (such as the HP 9133, HP 9122, and most other newer drives). `INTERNAL` — an internal, 5¼-inch, flexible-disc drive (Models 226 and 236 only). `HP8290x` — either an HP 82901 or 82902 5¼-inch, flexible-disc drive. `HP9895` — an HP 9895 8-inch, flexible-disc drive. For a list of all device types, see the *BASIC Language Reference* entry for MASS STORAGE IS.

Component	Explanation
Device selector	identifies the interface's select code (4, and 7 through 31) as well as the device's primary address (HP-IB devices only). Here are some examples of device selectors: 700 specifies select code 7 and address 0 (note that device selectors with HP-IB addressing must contain 3 or 4 digits). 702 specifies select code 7 and address 2. 800 specifies select code 8 and address 0. 1402 specifies select code 14 and address 2.
Unit number	tells the system additional information about the device's unit-number setting. Many devices have hard-wired unit numbers, while others use the unit number to identify different portions of one disc. For instance, the unit number of the right drive of a 9122 is 0, while the left drive is unit 1 (note that internal drives of Models 226 and 236 Computers are numbered in the opposite order).
Volume number	identifies the volume number (multi-volume hard discs only, such as HP 9133D, H, and L drives). For further information about volumes, see the subsequent section of this chapter called "Hard Disc Partitioning".

Examples

Here are several examples of volume specifiers:

```
:HP9122,700
```

or

```
:,700
```

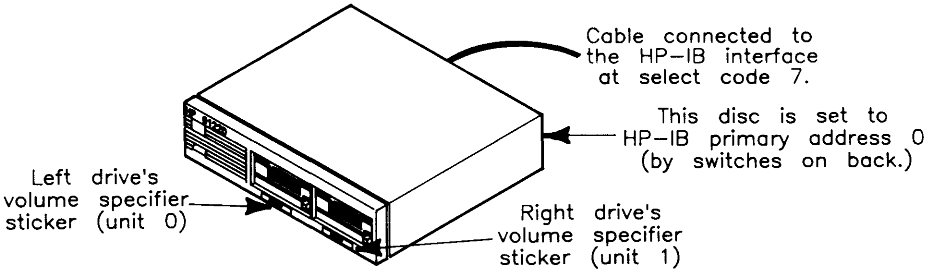

Cable connected to the HP-IB interface at select code 7.
This disc is set to HP-IB primary address 0 (by switches on back.)
Left drive's volume specifier sticker (unit 0)
Right drive's volume specifier sticker (unit 1)
9122D

Figure 3-5. HP 9122D Volume Specifier Example

```
:HP9121D,802,1
```

or

```
:HP8290x,802,1
```

or

```
:,802,1
```

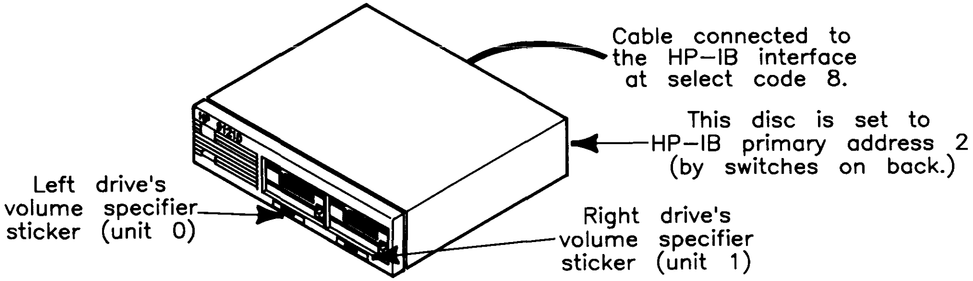

Cable connected to the HP-IB interface at select code 8.
This disc is set to HP-IB primary address 2 (by switches on back.)
Left drive's volume specifier sticker (unit 0)
Right drive's volume specifier sticker (unit 1)

Figure 3-6. HP 82901D Volume Specifier Example

```
:HP9133,1400,0,1
```

or

```
:,1400,0,1
```

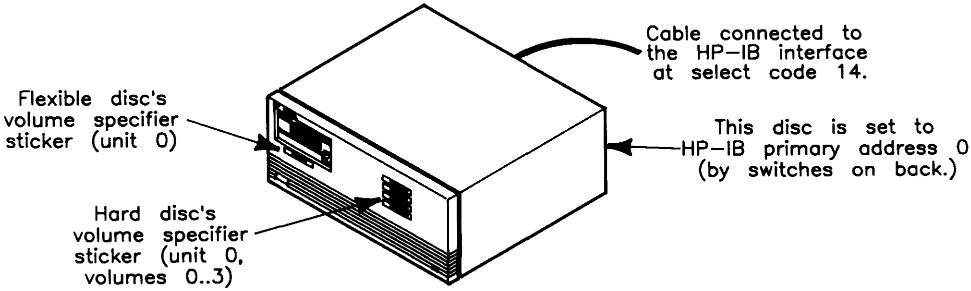

Cable connected to the HP-IB interface at select code 14.
This disc is set to HP-IB primary address 0 (by switches on back.)
Flexible disc's volume specifier sticker (unit 0)
Hard disc's volume specifier sticker (unit 0, volumes 0..3)

Figure 3-7. HP 9133L Volume Specifier Example

Checking a Disc's Format

To determine whether or not a disc is formatted, check to see if there is a directory on the disc. To do this, look at the catalog with the CAT command (you may need to specify the volume specifier if you loaded BASIC from a flexible disc):

CAT [Return] *Catalogs the default drive.*

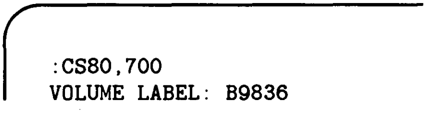
CAT ":,700" [Return] *Catalogs the volume* :,700,1

There are two classes of results for a CAT operation:

- A catalog is displayed (indicating the disc is formatted)
- An error is reported (indicating the disc is not formatted).

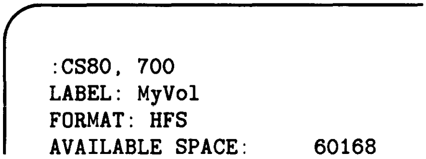
The Disc Is Formatted

LIF



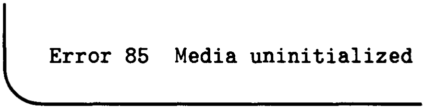
```
:CS80,700
VOLUME LABEL: B9836
```

HFS



```
:CS80, 700
LABEL: MyVol
FORMAT: HFS
AVAILABLE SPACE:      60168
```

The Disc Is Not Formatted



```
Error 85  Media uninitialized
```

The top catalog listing is shown for a LIF volume, and the bottom listing is for an HFS volume.

If you are not sure which format you would rather use, see the subsequent section called "Choosing a Directory Format".

Hierarchical Directories

A directory contains information about files, such as file name, size, type, etc. In hierarchical directory structures, a directory is itself a file, but it is used only to organize and control access to other files. This section describes the two BASIC directory formats which implement hierarchical directories:

- Hierarchical File System (HFS) format.
- Shared Resource Manager (SRM) format (the disc format is actually called Structured Directory Format, or SDF, on catalog listings of these directories).

What Is a Hierarchy?

As the word "hierarchy" suggests, hierarchical directories are arranged in "levels." Directories may contain either files or other directories.

- A directory is "superior" to the files and directories it contains.
- A file or directory within a directory is said to be "subordinate" to the containing directory.

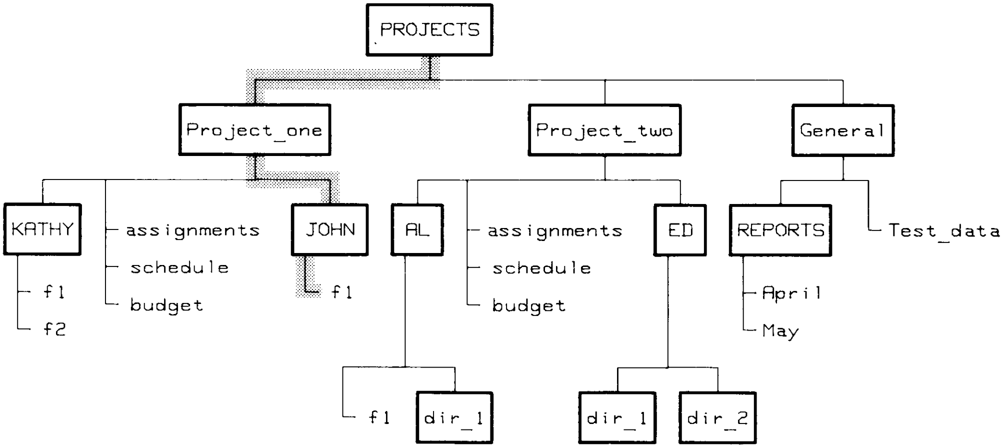


Figure 3-4. Hierarchy of Directories

In the Figure 3-4, the directory named `KATHY` is subordinate to the directory named `Project_one` because `Project_one` contains the information describing `KATHY`. The directory named `PROJECTS` is at level 1, the "root" level. You cannot create a directory at a higher level than the root level.

Each directory keeps information about each file or directory immediately subordinate to it, in fixed-format records. Each time a subordinate file or directory is added to a directory, one of these records is added to the directory.

Uses of the Hierarchy: An Example

Suppose you're managing several projects, each of which needs to access a shared disc. To organize the files for each project separately, you can create a directory for each project (as shown in the illustration). Within each project directory, you can have a subordinate directory for each person working on the project as well as files to be shared among all users. Each person may then construct a directory/file system for organizing their own files.

Because files at different locations in the directory structure can have the same file name, you can use generic file names to identify similar project functions in the different projects. At the same time, the division into separate directories isolates the projects, and thus their individual functions, from one another. For example, the file named `budget` in the `Project_one` directory is distinct from the file named `budget` in the `Project_two` directory.

Directories also limit the number of files users must deal with at any one time. For example, people working on `Project_one` (see illustration) need never see the files in `Project_two` and may, in fact, confine most of their activity to within their own directories.

To maintain security, BASIC provides the capability of protecting access to directories and files. For example, you may wish to allow only members of a project team to read that project's files. Or, you may wish to prevent other users from altering the contents of a personal file. See the "Protecting Directories and Files" section of the "Using Directories and Files" chapter.

Referring to Directories and Files in the Hierarchy

To access either a directory or a file, you must specify its location in the hierarchical directory structure. This location is specified by a list of directories, called a directory path, that you must follow to reach the desired file or directory. Directory names in the list are delimited by a slash (/).

For example, in the directory structure illustrated previously, the file specifier:

```
"/PROJECTS/Project_one/JOHN/f1"
```

defines the "directory path" to the file `f1` through its superior directories.

The directory path to a file begins at either:

- the root level, or
- the current working directory.

The current working directory is the directory specified by the most recent MASS STORAGE IS statement.

The *BASIC Language Reference* discusses the rules for specifying SRM and HFS files and directories.

Choosing a Directory Format

On the Series 200/300 BASIC system, there are three directory formats available for discs (and other mass storage media):

- Logical Interchange Format (LIF)
- Hierarchical File System (HFS)
- Structured Directory Format (SDF, used on Shared Resource Manager, or SRM, systems)

General Recommendation

The general recommendation is to:

- Use HFS format with hard discs.
- Use LIF format with flexible discs.

However, your values of the above criteria may suggest you make a different decision in your particular circumstances.

Characteristics of each format are described below.

Criteria for Choosing a Directory Format

Here are the criteria in choosing between LIF and HFS directory formats.

Feature	**HFS**	**LIF**	**For More Info**
Directory structure	Hierarchical (multi-directory) structure.	Single directory on each volume.	See preceding examples, and the "Using Files and Directories" chapter.
Multiple systems on same volume	HP-UX, BASIC, and Pascal systems can share a disc.	BASIC and Pascal can share a disc.	See the following section called "Hard Disc Partitioning and Multiple Systems".
File compatibility	"Text" files are the interchange method.	"ASCII" files are the interchange method.	See the "Porting and Sharing Files" chapter of *BASIC Programming Techniques.*
Extensible files	Files are extensible (when a file would otherwise overflow, the system automatically adds an incremental amount of space to it)	File length is fixed.	See "Data Storage and Retrieval" chapter of *BASIC Programming Techniques.*
Access times	Generally slower than LIF.	Generally faster than HFS.	For more exact data, you can use the benchmarking methods shown in the "Efficient Use of the Computer's Resources" chapter of *BASIC Programming Techniques.*
TRANSFER	Not really implemented as a background process.	True background process, and high data rates.	See the "Advanced Transfer Techniques" chapter of *BASIC Programming Techniques.*
Overhead required	Requires slightly more overhead than LIF.	Requires slightly less overhead than HFS.	See the next section for examples.

Examples of HFS Overhead

This section shows some figures of merit regarding how much of a disc is used as "overhead" on an HFS volume. (Overhead is the space on the disc required to store just the directory format, and not any data files.)

Disc Size	Approximate Overhead
single-sided 3½-inch, and 5¼-inch flexible discs	44% (256-byte sectors)
double-sided 3½-inch flexible discs	22% (256-byte sectors) 18% (1024-byte sectors)
55 Megabyte hard disc	6%
130 Megabyte hard disc	6%

In addition, you should keep the disc less than 90% full for optimum performance.

These figures show only the "extremes" of the spectrum; however, they do show that:

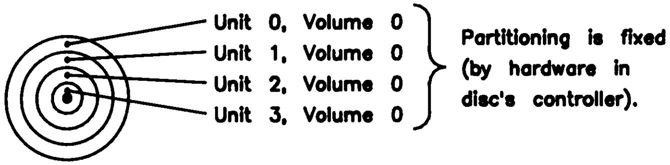
- The overhead for larger hard discs is fairly constant at about 6%.
- Overhead for flexible discs may be larger than expected (and more than tolerable).

Hard Disc Partitioning (Unit and Volume Numbers)

Most hard discs have their own microprocessor controller. Some controllers treat the entire disc as one single volume, while others partition the disc into several *volumes*. This section describes the types of partitioning available and how to access them from BASIC. In particular, it describes how this partitioning affects the disc's ability to have more than one operating system (such as BASIC, HP-UX, and Pascal) on the same disc.

Fixed Partitioning (BASIC Unit Numbers)

With fixed partitioning, the number of units on the disc cannot be changed. An example is the HP 9134B hard disc, which is divided into four separate entities.



The volume specifiers for these drives might be as follows (the partitions are indicated by specifying varying *unit numbers*):

:,700,0	*unit 0*
:,700,1	*unit 1*
:,700,2	*unit 2*
:,700,3	*unit 3*

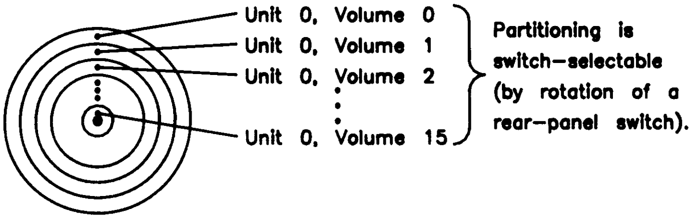
Implications for Systems Sharing the Disc

If Pascal is already on the disc and has logical partitioning, then BASIC can access only the first volume. If BASIC is to share with Pascal then: install Pascal first (with no logical partitioning) and use their "block size" and "frag size", etc. See the "Special Configurations" chapter of the *Pascal Workstation* Volume 2 manual.

Some versions of HP-UX support this type of disc. Check the *Configuration Reference* for details.

Switch-Selectable Partitioning (BASIC Volume Numbers)

With switch-selectable partitioning, the number of volumes on the disc is chosen by rotating a switch on the drive's back panel. An example is the HP 9133L hard disc, which can be configured as one volume or up to 16 volumes:


Unit 0, Volume 0
Unit 0, Volume 1
Unit 0, Volume 2
Unit 0, Volume 15
Partitioning is switch–selectable (by rotation of a rear–panel switch).

The volume specifiers for these drives might be as follows (the partitions are indicated by specifying varying *volume numbers*, with the unit number remaining constant):

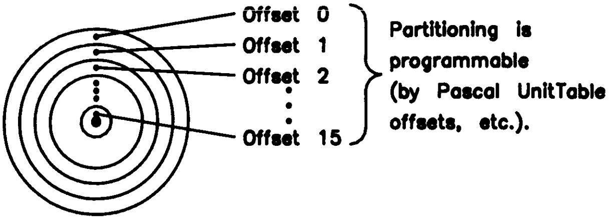
`:,700,0,0`	*volume 0*
`:,700,0,1`	*volume 1*
`:,700,0,2`	*volume 2*
. . .	
`:,700,0,15`	*volume 15*

Implications for Systems Sharing the Disc

If your disc has switch-selectable partitioning, then BASIC (and Pascal and/or HP-UX) can all access each of these volumes as independent entities. (Note the HP-UX unit must be HFS format, while the BASIC and Pascal units can be either LIF or HFS formats.) See the *Configuration Reference Manual* for more details.

Logical Partitioning (Pascal System "Unit Table")

The HP Series 200/300 Pascal Workstation System can perform its own logical partitioning (by the way that the directories are created and the "Unit Table" is set up[1]). An example could also be an HP 9133L hard disc, which is configured as one volume (on the rear-panel switches) but has 16 logical volumes (by Pascal partitioning):


Offset 0
Offset 1
Offset 2
Offset 15
Partitioning is programmable (by Pascal UnitTable offsets, etc.).

Pascal accesses these logical units by calculating the offset of each volume (from the beginning of the disc). Each logical unit then begins one byte past the sum of the offset plus the size of the preceding volume. **However, since BASIC has no structure like the Pascal Unit Table, BASIC can access only the first logical volume.**

Furthermore, the Pascal system can "overlay" its logical partitioning onto discs which are partitioned with either of the above two methods.

Implications for Operating Systems Sharing the Disc

If your disc has logical partitioning, then BASIC can access *only* the first volume[2]. Therefore, you might want to make this volume large enough for your BASIC needs, and then use the remainder of the volumes for Pascal needs[3]. See the "Special Configurations" chapter of the *Pascal Workstation System, Volume II* for details of logical partitioning.

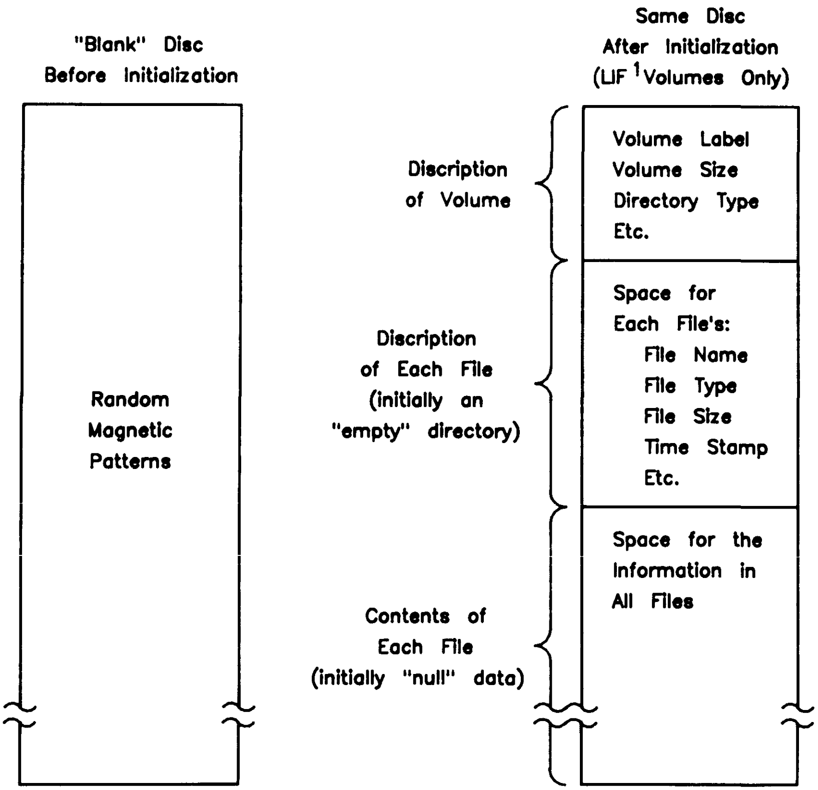
[1] See the "Special Configurations" chapter of the manual entitled *Pascal Workstation System, Volume II* for details of how this partitioning is performed.

[2] HP-UX cannot use the disc at all, since it has the LIF format.

[3] Note, however, that the bootable portion of the Pascal system must reside in this first volume (the SYSTEM_P, INITLIB, STARTUP, and TABLE files).

What Is Initialization?

When a "blank" disc is shipped from the factory, there are no files on it. In fact, there is not even an empty directory on it. You will therefore need to put one on before using it. (Some computer systems and documentation call this "formatting" a disc.) The "System Disc Utility" (DISC_UTIL) or the INITIALIZE statement will do this for you.



[1] HFS & SRM volumes have different structures, since they can have multiple directories on each volume.

Bad Sector Scan

Initialization writes “null” data onto the disc, and then reads the data to verify that it was written correctly. If the data was stored properly in a particular “sector” of the disc, then that sector is considered to be functional. However, if what was written onto the disc is not what is read back, then the sector is considered “bad” and is marked accordingly—or “spared” (and is not used subsequently).

Note

Note that initialization is also a good thing to do periodically to remove “bad” sectors from use. **(But remember to first make a back-up copy of the information on the disc, since all files will be overwritten!)**

Disc Sectors

Data on mass storage devices is written in "sectors", which are sets of contiguous bytes on the disc (on the same track). A sector is the smallest unit of data that can be written and read. Thus, if only one byte on a disc is to be changed, then the entire sector in which the byte resides must be read, the appropriate byte changed, and the entire sector re-written.

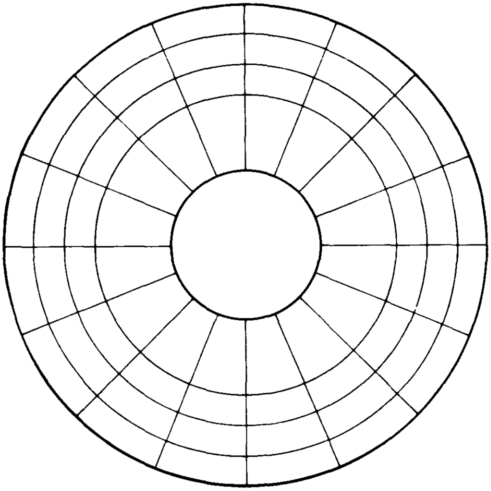


Figure 3-8. Discs are Accessed by Sectors

Sector/Volume Size Option

With this BASIC system, sector sizes of 256 and 1024 bytes are available. The "System Disc Utility" uses the default sector/volume-size option of 1, which specifies 256-byte sectors. If you want a non-default sector size, then you must use the INITIALIZE statement. See the descriptions of formatting discs in the following two chapters for details.

Disc Interleave

Interleaving a disc causes the sectors on the disc to be numbered according to a specific interval. For instance, an interleave factor of 1 causes sectors to be numbered consecutively. A factor of 2, on the other hand, causes the system to numbering to skip every other sector. The following drawing illustrates these two interleave factors.

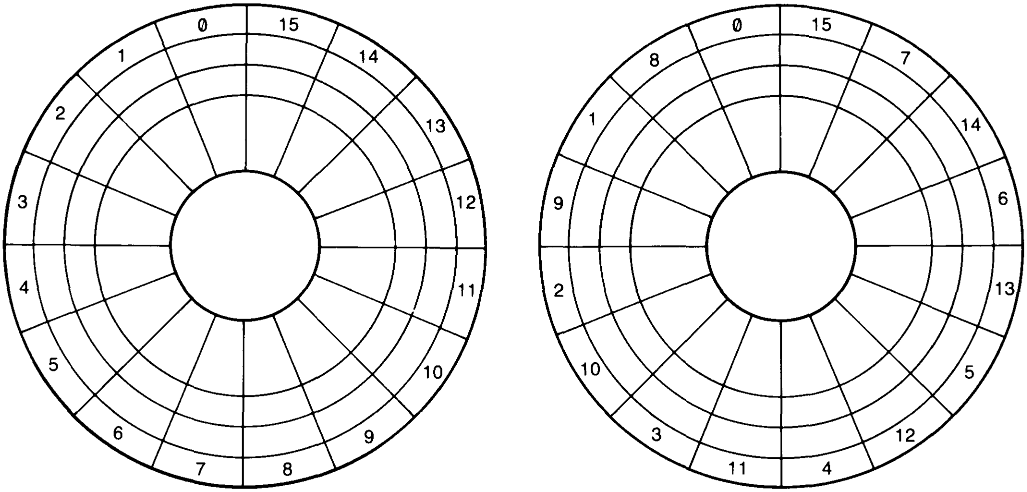


Figure 3-9. Examples of Disc Interleave

The BASIC system numbers each sector on the disc using the same interleave factor; that is, each file cannot have a separate interleave factor.

The purpose of disc interleave is to increase data-transfer rates, as demonstrated in the following example. Suppose that we are entering data from a spinning disc; suppose also that the data have formats which are different from the computer's internal data formats. Consequently, after each item is read, the computer must change the data from the disc's format to the computer's internal format. And in the meantime, the disc is still spinning.

If the processing of all items in a sector takes more time than it takes for the next sector to pass under the read/write head, then the system must wait one full revolution of the disc until the next sector again passes under the head. By interleaving a disc, you can provide time for this processing, thus not making the system wait as long until the next sector passes under the read/write head and can be read.

You can also choose the interleave factor when you initialize a disc using the INITIALIZE statement (but not when you initialize it using the "System Disc Utility"—the DISC_UTIL program). See the following two chapters for instructions on using these two tools.

What to Do Next

Task/Topic	Chapter
If you have only flexible disc drives, proceed to the next chapter, "Preparing Flexible Discs".	4
If you have a hard disc, skip ahead to the "Putting BASIC on a Hard Disc" chapter.	5
If you are installing BASIC onto an SRM system, skip ahead to the "Putting BASIC on the SRM" chapter.	6

Notes

Preparing Flexible Discs 4

If you have a hard disc, and you wish to install BASIC, skip this chapter and read Chapter 5, "Putting BASIC on a Hard Disc". You can return to this chapter when you need to format working flexible discs.

When a "blank" disc is shipped from the factory, there are no files on it. In fact, there is not even an empty directory on it. You will therefore need to put one on before using it. This process is called "initializing" the disc[1]. If you have a 3½-inch *double*-sided flexible disc, you may also want to store your entire BASIC system on this disc. This chapter describes these operations on BASIC system.

[1] On some systems, this is called "formatting" a disc. See "What Is Initialization?" in the "Mass Storage Concepts" chapter for a description of initialization.

Prerequisites

You should meet all of the following prerequisites:

- Configured and installed your computer, keyboard, and monitor. If not, please refer to your computer's *Installation Card* or *Installation Reference* for instructions.
- Configured and installed at least one flexible disc drive. If not, please refer to the *Peripheral Installation Guide* for instructions.
- Loaded ("booted") the BASIC system. If not, please turn to the tab labeled "Loading BASIC."
- Labeled your disc drive(s). If not, please turn to the tab labeled "Verifying and Labeling Peripherals."

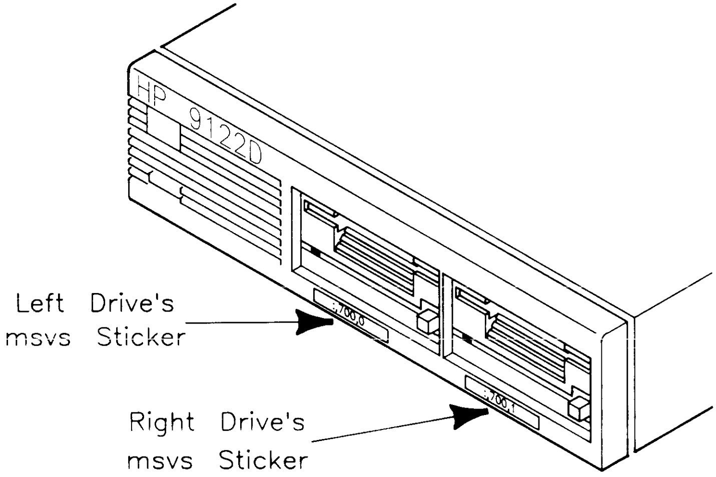


Figure 4-1. Labeling Disc Drives

Formatting Flexible Discs (Condensed Procedure)

Here is a list of the steps you will take in formatting flexible discs for use with the BASIC system. Each step is fully explained in a subsequent section.

1. Determine whether you will need to format a disc by looking at its catalog (using the CAT statement). You will get one of the following results:
    - If there is no directory on the disc: then it probably has not been formatted; you must format the disc before you can use it with BASIC (as described in subsequent steps of this procedure).
    - If there is a directory and there are any files on the disc that you don't want to destroy:
        - Use another disc, or
        - Copy these files onto a back-up disc (use the "Back-Up Utility" described in the "Maintaining Your BASIC System" chapter, or use individual COPY statements), and then re-format this disc for use as the "working" disc.
    - If BASIC is to share a disc with Pascal or HP-UX, then you may need to make additional considerations. (See the note in the explanation of this step on subsequent pages).
2. Check to see that the flexible disc is write-enabled.
3. Format the disc.
4. Verify the format by using the CAT statement.

If you are copying the BASIC system for back-up purposes:

5. Using COPY to make back-ups.
6. Create working discs.

The following sections explain each of these steps; each explanation also points to the location of relevant conceptual information (in the preceding "Mass Storage Concepts" section).

Formatting Flexible Discs (Detailed Procedure)

This section contains a more detailed explanation of the steps outlined in the preceding section.

Step 1: Do You Need to Format?

After inserting the disc into a drive, look at the catalog to determine whether or not there is a directory on the disc:

`CAT`	*Catalogs the default drive.*
`CAT ":,700,1"`	*Catalogs the specified drive.*

There are two possible results of the CAT operation:

- A catalog is displayed (indicating the disc is formatted).
- An error is reported (the disc is probably not formatted).

The next page describes how to respond to each situation.

The Disc Is Formatted

```
:CS80,700
VOLUME LABEL: B9836
```

```
:CS80, 700
LABEL: MyVol
FORMAT: HFS
AVAILABLE SPACE:     60168
```

If the **formatted** disc **does have** the directory format you want (LIF or HFS), then you do not need to re-format. If the disc contains any valuable files that you want to keep, then either:

- use another disc, or
- make a back-up copy of each on another disc (see "Backing Up Files" in the "Maintaining Your BASIC System" chapter for instructions).

If the **formatted** disc **does not** have the desired directory format, then you will need to re-format it with the desired directory format as described in the subsequent steps.

The Disc Is Not Formatted

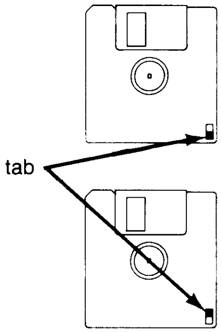
```
Error 85  Media uninitialized
```

If the disc is **not formatted**, you will need to format it as described in the subsequent steps.

Step 2: Write-Enabling a Flexible Disc

If you have a flexible disc, you may need to enable it for writing.

3½-inch, double-sided discs



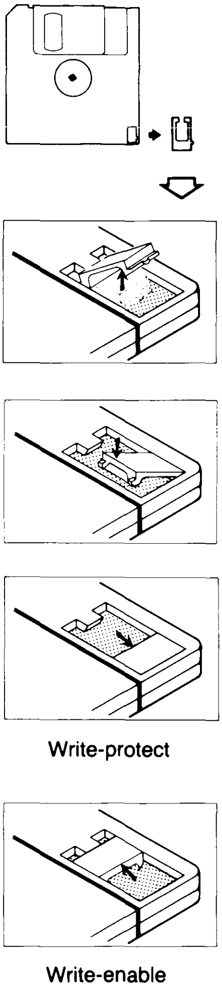
To write-protect the disc, make sure the you can see through the write-protect hole in the corner of the disc.

To write-enable the disc, make sure you can **not** see through the write-protect hole in the corner of the disc.

Figure 4-2. Write-Enabling a Flexible Disc

3½-inch, single-sided discs

Perform the following steps to write-enable these discs:



1. Weaken or score the attach point so the tab is not broken. Break off the write-protect tab.

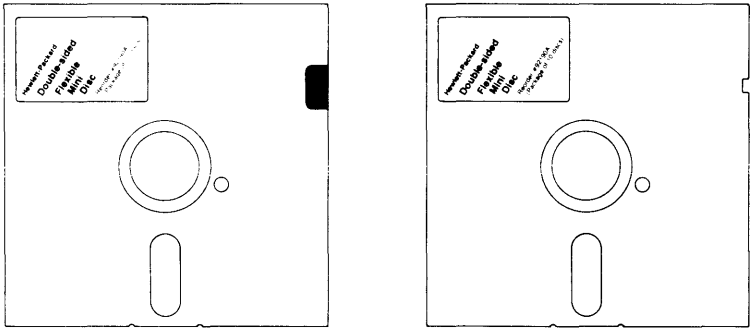
2. Align the protrusion on the tab with the groove in the disc.

3. Depress the tab into the groove–tab should fit snugly. In the position shown, the tab write-protects the disc.

4. To write-enable the disc, slide the tab up.

Figure 4-3. 3½-inch, Single-Sided Discs

5¼-inch discs: To write-enable these discs, uncover the notch in the disc jacket.



Write-protected **Write-enabled**

Figure 4-4. 5¼-inch Mini-Disc Write-Protection

Step 3: Formatting a Disc (Using INITIALIZE)

This section describes how to format discs using the INITIALIZE statement as a keyboard command.

CAUTION

FORMATTING DESTROYS ALL DATA AND PROGRAMS STORED ON A DISC. BE CERTAIN THAT YOU DO NOT REFORMAT A BASIC SYSTEM DISC, OR ANY OTHER DISC CONTAINING VALUABLE FILES.

IF THERE ARE FILES ON THE DISC THAT YOU WANT TO KEEP, THEN FIRST MAKE A BACK-UP COPY OF EACH BEFORE RE-FORMATTING THE DISC. SEE "Backing Up Files" IN THE CHAPTER CALLED "Maintaining Your BASIC System" FOR DETAILS.

IF FORMATTING A HARD DISC, DO NOT INTERRUPT THE FORMATTING PROCESS, AS THIS MAY SEVERELY DAMAGE THE DRIVE.

Note

You may not need to re-format your disc if has been previously formatted by the Series 200/300 Pascal or HP-UX systems. For further information, see the section called "Hard Disc Partitioning" in the "Mass Storage Concepts" chapter.

Here is the syntax of the INITIALIZE statement. The parameters are briefly explained below, and then several examples are shown.

`INITIALIZE " :` *volume specifier*`" ,` *interleave* `,` *sector/volume size* *volume specifier* identifies your disc. (The volume specifier should be labeled on stickers affixed to the front of all drives in your system; if not, consult the chapter called "Verifying and Labeling Peripherals".)

interleave specifies how the sectors will be numbered. (Interleave is discussed in "Disc Interleave" in the "Mass Storage Concepts" chapter.) If you are not sure about what interleave to use, then specify the default value for your particular disc (0 also specifies the default value).

Flexible Disc Type	Typical Product Numbers	Default Interleave
3½-inch double-sided (gray)	HP 9122, 9133D, 9153	2
3½-inch single-sided (blue)	HP 9121, 9133A	2
5¼-inch internal	Built-in drives of HP 9826 and 9836 computers	1
5¼-inch external	HP 82901, 82902, 9134, 9127	3

sector/volume size specifies the size of sectors and the volume on *double*-sided, 3½-inch, flexible discs only; omit this parameter or use the default value or 0 with all other discs. (See the "Disc Sectors" section of the "Mass Storage Concepts" for further explanation.)

Parameter Value	Sector Size (Bytes)	Double- or Single-Sided	Capacity (Bytes)
0 (default)		*device-dependent*	
1	256	Double-sided	630 K
2[1]	512	Double-sided	71- K
3	1024	Double-sided	788 K
4	256	Single-sided	270 K

[1] Not supported.

CAUTION

Make sure you use the **correct volume specifier**. If you accidentally use the volume specifier for a hard disc, the contents of that disc will be DESTROYED.

Examples

The following statement formats the disc on select code 7, with primary address 0 and unit number 1. The disc is formatted with the default *interleave factor* (device-dependent) and default *sector/volume size* option (also device-dependent).

```
INITIALIZE ":,700,1",0,0
```

(Since the defaults for interleave factor and sector/volume size option are 0, this is the same as `INITIALIZE ":,700,1"`.)

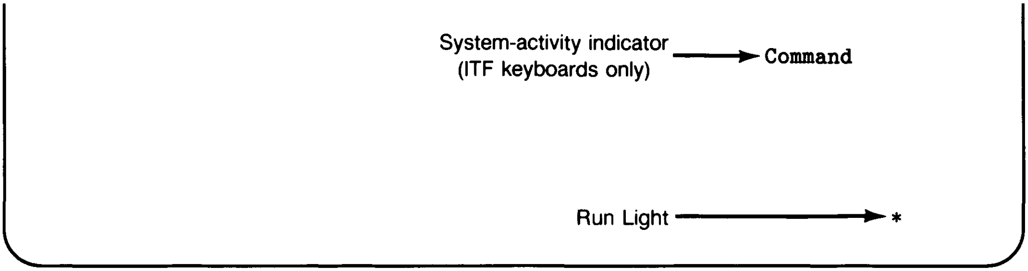
The following example formats the same disc with an interleave factor of 2 and sector/volume size option 3.

```
INITIALIZE ":,700,1",2,3
```

This last example formats the disc at select code 8, with primary address 2 and unit number 0. The disc is initialized with an interleave factor of 3 and sector/volume size option 4.

```
INITIALIZE ":,802",3,4
```

The formatting process will continue for a few minutes, as indicated by the asterisk (*) in the lower-right corner of the screen. (`Command` is present if in KEY LABELS ON mode.)



When this indicator is turned off (`Idle` will be displayed if in KEY LABELS ON mode), the process has completed.

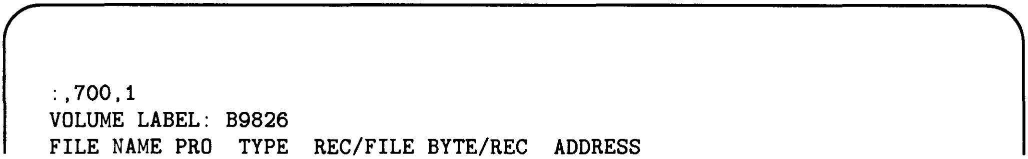
The time required to format a disc is dependent on the size of the disc. The larger the disc, the longer it will take to format the disc. So, depending on the size of your disc, you may have to wait from a few minutes up to fifty minutes for the process to complete.

Step 4: Verify the Format

After formatting is complete, you can list the catalog of the (empty) disc. Execute a `CAT` statement. Here are appropriate CAT statements for the two different volume specifier examples above:

```
   CAT ":,700,1"
```
or
```
   CAT ":,802"
```

Here are typical results of this operation for a LIF disc:



```
:,700,1
VOLUME LABEL: B9826
FILE NAME PRO  TYPE  REC/FILE BYTE/REC  ADDRESS
```

Step 5: Using COPY to Make Back-Ups

Two procedures for backing-up volumes and files are provided:

- Use **Procedure 1** if you have *two* or more flexible disc drives.
- Use **Procedure 2** if you have *only one* flexible disc drive.

Procedure 1: Copying Discs with Two Drives

This procedure shows how to copy an entire disc or an individual file using two disc drives. As an example of coping a disc, we will have you make a back-up copy of your *BASIC System* disc. The example of copying an individual file will be to copy the **SYSTEM_BA5** file on the *BASIC System* disc.

1. Make sure your source disc is **write-protected** (to avoid inadvertently destroying information it contains in the event that you make a typographical error in one of the commands shown below).

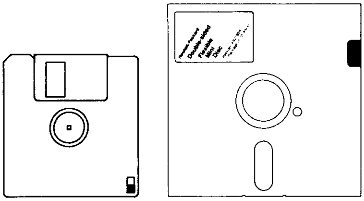


Figure 4-5. Write Protect Discs

2. Insert the source disc (the BASIC System Disc in this example) into one of your flexible disc drives.

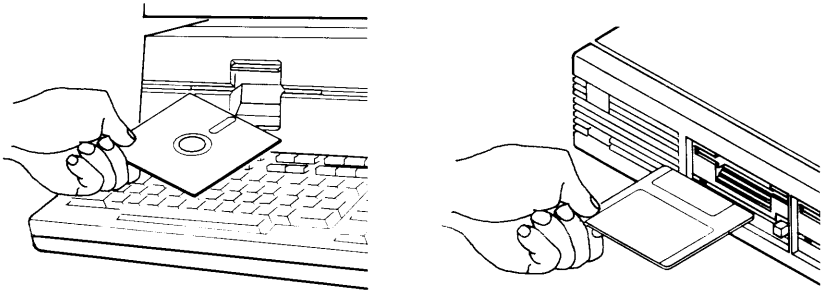


Figure 4-6. Inserting Source Disc

3. Insert the destination disc (the one to be used as the back-up copy) into your *other* flexible disc drive.

Note

If you are going to copy an entire disc, make sure that your destination disc **does not** contain any valuable files. All files on that disc will be destroyed in the disc-copy operation!

4. Use COPY to copy the disc or file.

Copying an Entire Disc: Substitute the volume specifier of your source and destination drives into the following statement (the volume specifier of every drive in your system should be labeled with a sticker that HP supplies with the BASIC system; if not, see the "Verifying and Labeling Peripherals" chapter of this manual).

```
COPY "source_vs" TO "destination_vs" Return
```

Suppose you have this configuration:

- the volume specifier of the drive containing the source (*BASIC System*) disc is `:,700,0`
- the volume specifier of the drive containing the destination (back-up) is `:,700,1`

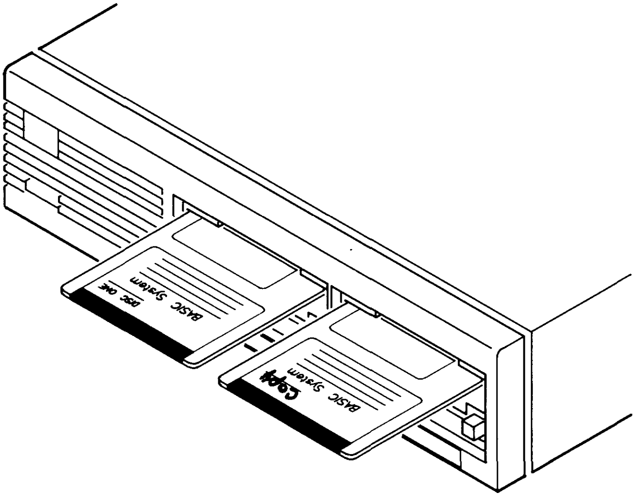


Figure 4-7. COPY Example Configuration

The appropriate COPY statement would be:

```
COPY ":,700,0" TO ":,700,1" Return
```

Copying an Individual File: Substitute the name of your source file into the statement below. And, as in the above example, also substitute the volume specifier of your source and destination drives into the as appropriate.

```
COPY "source_file:source_vs" TO "dest_file : dest_vs" Return
```

If you were copying the SYSTEM_BA5 file from the *BASIC System* disc onto the back-up disc, this statement would be appropriate (assuming the vs–volume specifier–of source and destination disc are as shown in the preceding example):

```
COPY "SYSTEM_BA5:,700,0" TO "SYSTEM_BA5:,700,1" Return
```

5. Wait a few minutes until the asterisk (*) disappears from the lower right corner of the screen. (If you have KEY LABELS ON, the message **Command** is also shown on the screen.)

6. Verify the COPY by using CAT:

```
CAT ":,700,1"
```

7. **If copying individual file(s):** Repeat steps 4 through 6 for each file to be copied.

   If copying an entire disc: Remove the source (*BASIC System*) disc and the back-up (duplicate) disc from their drives.

   Print a label for the duplicate disc; use the same information that appears on the original disc. Write "Copy" on the label and record the date. Peel the backing off of the label, and put it on the disc.

   Store the original discs in a safe place. Use the new duplicate discs for daily operation.

   Repeat steps 1 through 7 for each disc to be copied.

Procedure 2: Copying Discs with One Drive

This procedure shows how to copy an entire disc or an individual file using one disc drive. As an example of coping a disc, we will have you make a back-up copy of your *BASIC System* disc. The example of copying an individual file will be to copy the SYSTEM_BA5 file on the *BASIC System* disc.

In this procedure, you will be creating a memory volume, copying a disc (or file) into it, removing the source disc from the drive and inserting the destination (back-up) disc into the same drive, and then copying the disc (or file) from the memory volume onto the destination disc.

1. Create a "memory volume" in your computer's main memory. If you have not already created one, then use the following statement (to create a 270-Kbyte disc):

   ```
   INITIALIZE ":MEMORY,0" [Return]
   ```

2. Make sure your source (original) disc is **write-protected** (to avoid inadvertently destroying information it contains in the event that you make a typographical error in one of the commands shown below).

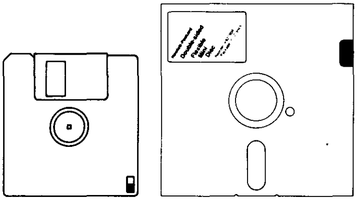


Figure 4-8. Write-Protect Discs

3. Insert the source disc (the *BASIC System* disc for this example) into your flexible disc drive.

4. Use COPY to copy the disc (or file).

   Copying an Entire Disc: Substitute the volume specifier of your source and destination drives into the following statement (the volume specifier of every drive in your system should be labeled with a sticker that HP supplies with the BASIC system; if not, see the "Verifying and Labeling Peripherals" chapter of this manual).

   ```
   COPY "vs" TO ":MEMORY,0" [Return]
   ```

   where *vs* is the volume specifier of your flexible disc drive.

Suppose the volume specifier of your drive is :,700,0. Then you would type:

```
COPY ":,700,0" TO ":MEMORY,0" [Return]
```

Copying an Individual File: Substitute the name of your source file into the statement below. And, as in the above example, also substitute the volume specifier of your source and destination drives into the as appropriate.

COPY "*source_file*:*source_vs*" TO ":MEMORY,0" [Return]

If you were copying the SYSTEM_BA5 file from the *BASIC System* disc onto the back-up disc, this statement would be appropriate (assuming the volume specifier of the source disc is as shown in the preceding example):

```
COPY "SYSTEM_BA5:,700,0" TO "SYSTEM_BA5:MEMORY,0" [Return]
```

5. Wait a few minutes until the asterisk (*) disappears from the lower right corner of the screen. (If you have KEY LABELS ON, the message `Command` is also shown on the screen.)

6. Remove the source disc (the *BASIC System* in this example) from the disc drive and insert the destination (back-up) disc.

7. Use COPY again to copy the disc (or file) from memory to the destination disc.

   Copying an Entire Disc: Using the preceding assumptions, you would type:

   ```
   COPY ":MEMORY,0" TO ":,700,0" [Return]
   ```

   Copying an Individual File: To copy the file named SYSTEM_BA5 to the disc assumed in the preceding example, type:

   ```
   COPY "SYSTEM_BA5:MEMORY,0" TO "SYSTEM_BA5:,700,0" [Return]
   ```

8. Wait a few minutes until the asterisk (*) disappears from the lower right corner of the display.

9. Verify the COPY by using CAT:

   ```
   CAT ":,700,0"
   ```

10. **If copying an individual file:** Repeat steps 4 through 9 for each file to be copied.

    If copying an entire disc: Remove the source (*BASIC System*) disc and the back-up (duplicate) disc from their drives.

    Print a label for the duplicate disc; use the same information that appears on the original disc. Write "Copy" on the label and record the date. Peel the backing off of the label, and put it on the disc.

    Store the original discs in a safe place. Use the new duplicate discs for daily operation.

    Repeat steps 1 through 9 for each disc to be copied. Remove the duplicate disc from the disc drive and label it with the same information that appears on the original disc. Write "Copy" on the label and record the date.

Step 6: Create Working Flexible Disc(s)

After you have created back-up copies for your system, you should format several blank discs for use with the system. A working disc is a disc you use to store programs and files. Take several (one or two to start) blank discs and follow the same procedure for formatting the discs as in Steps 1-4. If you are re-using discs, you can simply remove any unwanted files and use the disc as is.

What To Do Next

Now that you have backed-up BASIC, you are ready to use the system. Your next task can be any one of the following:

Task/Topic	Chapter
If you did not install BASIC on your hard disc, and wish to.	5
Verify the peripherals other than the mass storage units, if you have not already done so.	2
Check or modify your system's configuration.	7
Learn how to load and run programs.	9
Learn how to use and manage files.	10
Learn how to enter, edit, secure, document and store programs.	11
Learn how to maintain your system.	12-17

Putting BASIC on a Hard Disc 5

There are several configurations possible when putting BASIC on a hard disc. This page folds out to reveal several tables on the reverse side which help you find the correct sections to read, depending on the configuration of your system. This chapter is broken into six (6) steps:

0. Install other operating system (if sharing a hard disc ,with the other system)
1. Backup the current files on the hard disc
2. Format the hard disc
3. Store the system and binaries on the hard disc
4. Create directories on the hard disc (optional)
5. Restore the files to the hard disc (if you backed-up in step 1)
6. What to do next.

Prerequisites

Before you do anything, make sure your hardware is configured. You should have your disc drives labeled. If not, see the "Verifying and Labeling Peripherals" chapter.

When you have satisfied the prerequisites, you need to take two things into consideration before you begin to install BASIC on a hard disc:

What is the current format of the hard disc? **LIF**, **HFS**, or **unformatted** (a new disc). To check the format of a disc, see the section "Checking the Disc's Format" in "Mass Storage Concepts" chapter.

Which disc format do you want? **LIF** or **HFS** (if you don't know which format you want, see "Choosing a Directory Format" in the "Mass Storage Concepts" chapter). We recommended you choose the HFS format for a hard disc.

Once you know what the current format is, and what format you want, you can look at the matrix on the next page (fold it out) to see which table you need to reference on the back side of this fold-out page.

Which Steps Do I Need To Perform?

Look down the "Current Format" column for the entry which best describes your current situation (for example, if you have a new hard disc with no format, go to the "New unformatted disc" entry). Then go horizontally to the column which best describes what format you want on the disc (for example, if you want the new disc to be HFS formatted, go to the column "HFS" under the "Desired Format" heading). The entry will tell you the name of a table listed on the back side of this page (in our example, you would perform the steps in Table 1). The table gives you the steps you need to follow for installing BASIC on your system.

		Desired Format		
		Only BASIC System on HFS Disc	**Only BASIC System on LIF Disc**	**Share Disc With Another System (HFS[1] or LIF)**
Current Format	**Currently unformatted**	Table 1: New Hard Disc	Table 1: New Hard Disc	Table 4: Share Disc with Other Operating System
	Currently HFS	Table 2: Keep the Same Disc Format	Table 3: HFS to LIF is not recommended	Table 4: Share Disc with Other Operating System
	Currently LIF	Table 3: Reformat the Disc to HFS	Table 2: Keep the Same Disc Format	Table 4: Share Disc with Other Operating System

You can keep this page folded out to expose the tables on the opposite side while you are completing the steps in this chapter.

Fold out →

[1] Requires HP-UX revision 5.0 (or later) or Workstation Pascal 3.2 (or later).

Which Steps Do I Need To Perform?

Look down the "Current Format" column for the entry which best describes your current situation (for example, if you have a new hard disc with no format, go to the "New unformatted disc" entry). Then go horizontally to the column which best describes what format you want on the disc (for example, if you want the new disc to be HFS formatted, go to the column "HFS" under the "Desired Format" heading). The entry will tell you the name of a table listed on the back side of this page (in our example, you would perform the steps in Table 1). The table gives you the steps you need to follow for installing BASIC on your system.

		Desired Format		
		Only BASIC System on HFS Disc	**Only BASIC System on LIF Disc**	**Share Disc With Another System (HFS[1] or LIF)**
Current Format	**Currently unformatted**	Table 1: New Hard Disc	Table 1: New Hard Disc	Table 4: Share Disc with Other Operating System
	Currently HFS	Table 2: Keep the Same Disc Format	Table 3: HFS to LIF is not recommended	Table 4: Share Disc with Other Operating System
	Currently LIF	Table 3: Reformat the Disc to HFS	Table 2: Keep the Same Disc Format	Table 4: Share Disc with Other Operating System

You can keep this page folded out to expose the tables on the opposite side while you are completing the steps in this chapter.

Fold out →

[1] Requires HP-UX revision 5.0 (or later) or Workstation Pascal 3.2 (or later).

Table 1: New Hard Disc (not sharing it with HP-UX or Pascal)

Steps To Complete	Page Number
2. Format the hard disc	5-7
3. Store the system and binaries	5-12
4. Create directories (optional on HFS discs)	5-18
6. What to do next	5-26

Table 2: Keep the Same Format

Steps To Complete	Page Number
3. Store the system and binaries	5-12
4. Create directories (optional on HFS discs)	5-18
6. What to do next	5-26

Table 3: Reformat the Disc (and not share with other operating systems)

Steps to Complete	Page Number
1. Backup the current files	5-4
2. Format the hard disc	5-7
3. Store the system and binaries	5-12
4. Create directories (optional on HFS discs)	5-18
5. Restore the files to the hard disc	5-19
6. What to do next	5-26

Table 4: Share Disc with Other Operating System

Steps to Complete	Page Number
0. Install other operating system Backup the current system[2] (not required for new discs) Format the hard disc[2] Store other operating system on hard disc[2] Restore the files on hard disc[2]	5-3
Boot BASIC (see Chapter 1)	1-1
3. Store the system and binaries	5-12
4. Create directories (optional on HFS discs)	5-18
6. What to do next	5-26

[2] Covered in other operating system's documentation. Return to this manual when finished.

0. Install Other Operating System

When sharing a disc with other operating systems, such as Series 200/300 HP-UX or Workstation Pascal, you should generally **install the other system first**. There are some good reasons for this order of installation:

- When formatting an HFS disc with BASIC, the swap space is set to 0. The HP-UX system cannot operate with this setting, and therefore you would need to backup the disc, re-format it with HP-UX, and then restore BASIC to the disc. So install HP-UX first, and save yourself a lot of work. (BASIC will respect any disc parameters set by the HP-UX system.)
- Similarly, Pascal can set non-default fragment-size or block-size disc parameters that BASIC cannot set. (But BASIC will also respect these parameters.)

Note

Before you do the BASIC installation portion of this step, read the section in Chapter 1, "Loading BASIC" called, "Using BASIC with HP-UX".

There is Always an Exception

If you want to share a hard disc with Pascal and do not mind if Pascal partitions the disc into two or more "logical volumes", you can install Pascal first. (However, BASIC can **only** access the first logical volume on the disc.) You could alternatively make sure that Pascal does not partition the disc, so that BASIC can access the entire disc. See the "Hard Disc Partitioning" section of the "Mass Storage Concepts" chapter for details on partitioning.

Where to Find Installation Instructions

Installation information for Series 200/300 Workstation Pascal is in the *Pascal User's Guide*, as well as in the *Pascal Workstation System, Volume 2* manual.

Installation information for Series 200/300 HP-UX systems is in the *HP-UX System Administrator* manual. (Installing an AXE system is described in *AXE System Administrator* manual.)

Next Step

After you install the other system and are ready to install BASIC, go back to the table in the front of this chapter to see the next section to follow.

1. Backing Up the Current System

The steps listed in this section are but **rough** steps for backing up and restoring files. There are other options than those described here such as backing up selected files. For a detailed description of the procedures, see the "Backing Up Files" chapter in the **Maintaining Your BASIC System** section.

Select the Backup Media

Select a backup media, either tape or flexible disc. Then, insert the media into the proper drive, and take note of the volume specifier of the drive.

If your media is a flexible disc and it is not formatted (to check, use the same steps in "Check the Hard Disc's Format", "Mass Storage Concepts" chapter) you should first format the discs if they are blank: see "Formatting Flexible Discs" in chapter 4, "Preparing Flexible Discs".

Loading the BACKUP Utility

The BACKUP Utility is located on either:

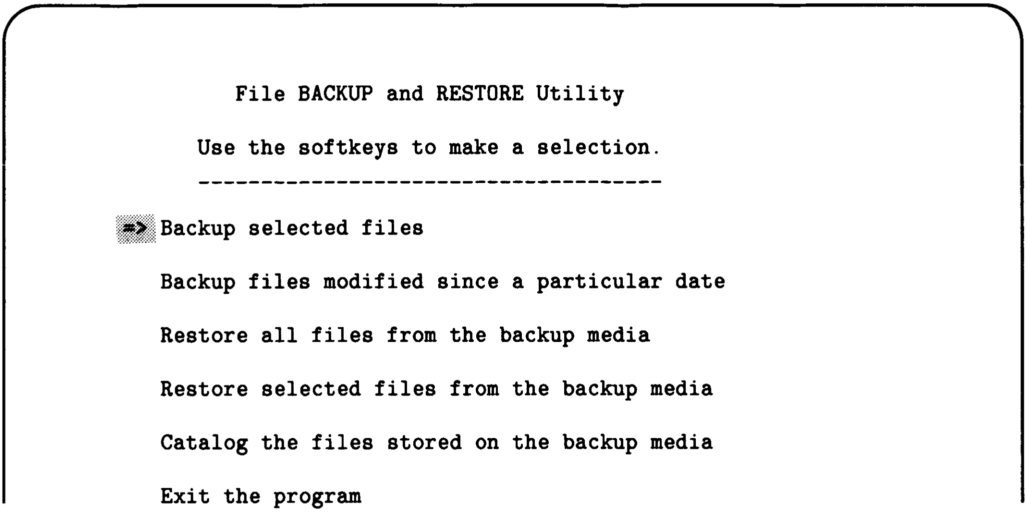
- The *BASIC Utilities* disc (with *double*-sided, 3½-inch discs).
- The *BASIC Utilities 2* disc (*single*-sided, 3½-inch discs; or all 5¼-inch discs).

Use the LOAD statement to load the program into computer memory (the second example shows the volume specifier on the end if necessary):

```
LOAD "BACKUP"
 or
LOAD "BACKUP:,700"
```

Run the program by either pressing the RUN key (f3 in the System menu of an ITF keyboard), or execute the RUN statement from the keyboard.

You should see the following display:

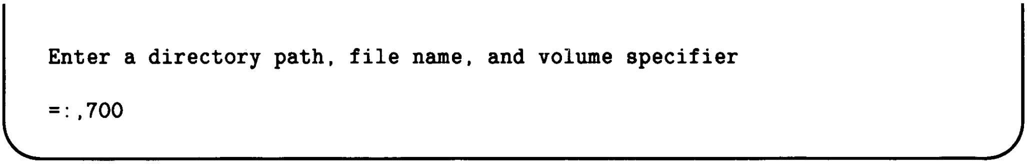


```
                  File BACKUP and RESTORE Utility

               Use the softkeys to make a selection.
               -------------------------------------

      => Backup selected files

         Backup files modified since a particular date

         Restore all files from the backup media

         Restore selected files from the backup media

         Catalog the files stored on the backup media

         Exit the program
```

Backup Files

This section assumes you wish to backup **all** files. If you need to backup only some of the files, you may need to read the description of wildcards in the "Backing Up Files" chapter.

First, insert the formatted backup media into a disc or tape drive. Then select the "`Backup selected files`" option. You should see the following:



```
Enter a directory path, file name, and volume specifier

=:,700
```

The = character is known as a *wildcard*, which is described in the "Backing Up Files" chapter.

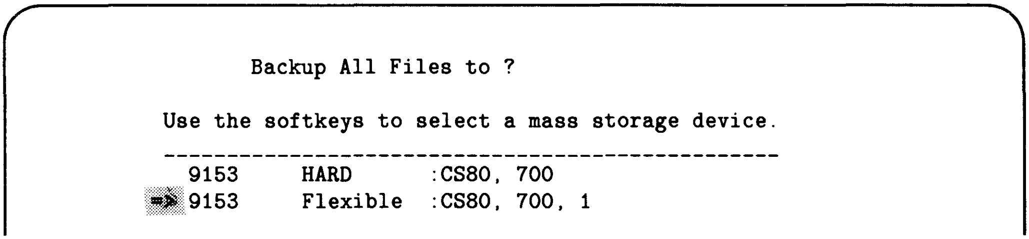
Backing Up All Files

1. If necessary, change the default prompt (for example, `=:,700`) to one that specifies all files on the volume you want to back up. For instance:

   `=:,702` Return

   Then press the softkey labeled `Proceed` (if you do not have any other volumes to back up).

2. The utility then prompts you to specify the back up media:



```
               Backup All Files to ?

          Use the softkeys to select a mass storage device.
          ------------------------------------------------
             9153     HARD       :CS80, 700
          => 9153     Flexible   :CS80, 700, 1
```

   Use the softkeys to select the backup device.

3. The utility then copies the specified files and/or directories from the source onto the backup media. (**If the back up media overflows, the utility will prompt you to install a different backup disc or tape.**) When the backup is complete,

   ```
   Backup is finished.
   ```

   is displayed. Press Return for the Main menu.

Problems

If you have any problems with this section, refer to the "Backing Up Files" chapter.

2. Formatting a Hard Disc

If you have a system already on the disc (like HP-UX or a previous release of BASIC) or the disc is already formatted (see "Check the Hard Disc's Format", Chapter 3), you should **NOT** read this section. Read the instructions on the first page of this chapter ("Putting BASIC on a Hard Disc") to see what section to read.

If your hard disc is "blank" (shipped from the factory), there are no files on it. In fact, there is not even an empty directory on it. You therefore need to put one on before using it. This process is called "formatting" the disc[1]. You will probably want to store your entire BASIC system on this disc, since it simplifies and speeds up the booting (loading) process. This section describes these operations on BASIC system.

If you need to specify non-default interleave factor or sector/volume size, then use the INITIALIZE statement to format the disc; turn to the section "Formatting with INITIALIZE" towards the end of this chapter for details.

CAUTION

FORMATTING DESTROYS ALL DATA AND PROGRAMS STORED ON A DISC. BE CERTAIN THAT YOU DO NOT REFORMAT A BASIC SYSTEM DISC, OR ANY OTHER DISC CONTAINING VALUABLE FILES.

IF THERE ARE FILES ON THE DISC THAT YOU WANT TO KEEP, THEN FIRST MAKE A BACK-UP COPY OF EACH BEFORE FORMATTING THE DISC. SEE "Backing Up Files" IN THE CHAPTER CALLED "Maintaining Your BASIC System" FOR DETAILS.

IF FORMATTING A HARD DISC, DO NOT INTERRUPT THE FORMATTING PROCESS, AS THIS MAY SEVERELY DAMAGE THE DRIVE.

[1] On some systems, this is called "initializing" a disc. See "What Is Initialization?" in the "Mass Storage Concepts" chapter for a description of initialization.

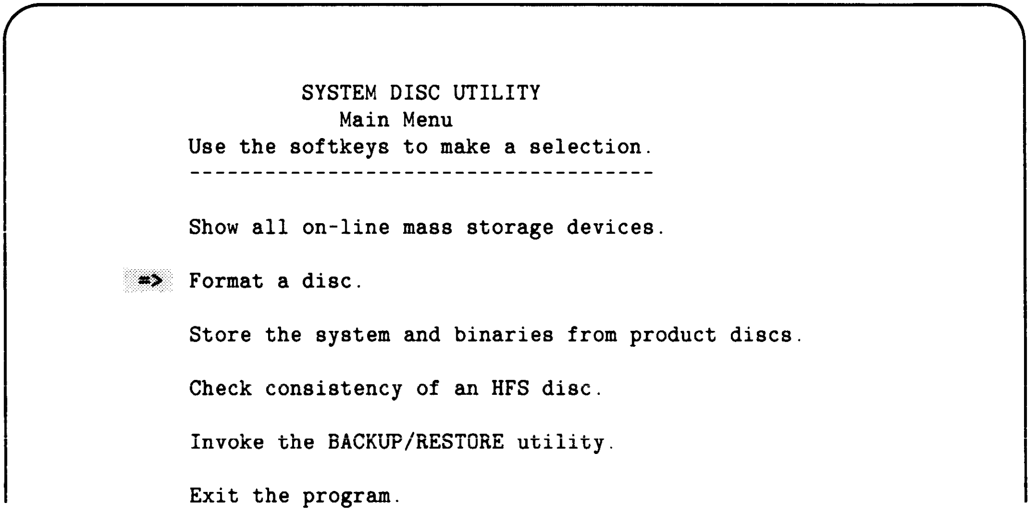
Step 1: Load and Run the System Disc Utility

The "System Disc Utility" is a tool used for formatting discs. Insert the *HFS Utilities* disc into the default drive (the one from which you loaded BASIC), and load the utility by typing (**add the volume specifier if the drive is not the default drive**):

```
LOAD "DISC_UTIL"  Return
```

It may take a minute or so to load the large utility.

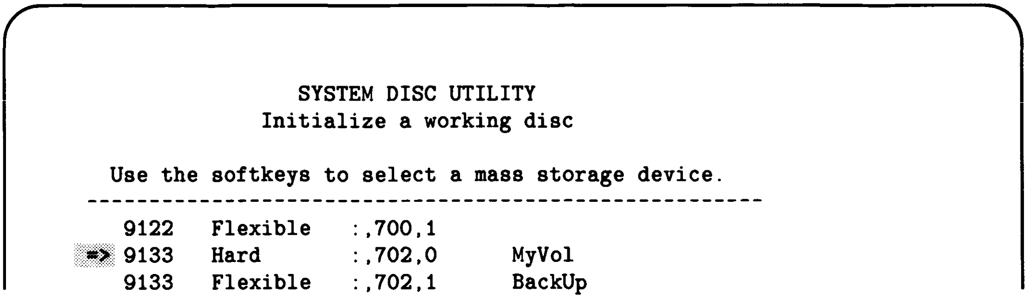
Now run the program by pressing RUN or by executing `RUN` Return. After the *legend* screen (press CONTINUE), you should see the following display.



```
                    SYSTEM DISC UTILITY
                       Main Menu
              Use the softkeys to make a selection.
              -------------------------------------

              Show all on-line mass storage devices.

          ==> Format a disc.

              Store the system and binaries from product discs.

              Check consistency of an HFS disc.

              Invoke the BACKUP/RESTORE utility.

              Exit the program.
```

Step 2: Select the "Format a disc" Option

From the main menu of the System Disc Utility, select the "`Format a disc`" option. You should see a display something like this:

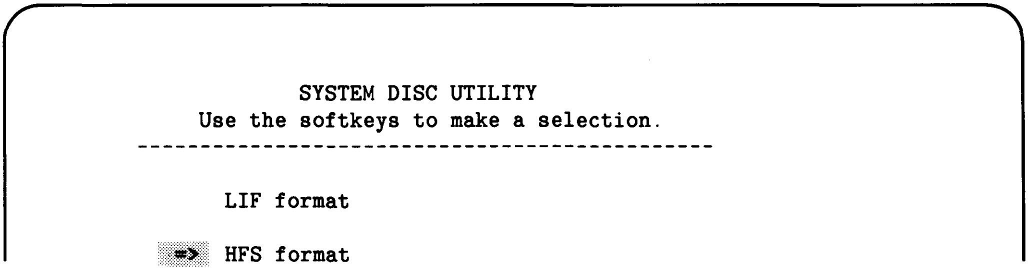


```
                    SYSTEM DISC UTILITY
                 Initialize a working disc

     Use the softkeys to select a mass storage device.
   -----------------------------------------------------
      9122   Flexible    :,700,1
   => 9133   Hard        :,702,0      MyVol
      9133   Flexible    :,702,1      BackUp
```

Then select the disc drive you want formatted (the hard disc).

Step 3: Select a Directory Format

Once you have selected which disc to format, the utility will prompt you to select either the LIF or the HFS format.



```
                    SYSTEM DISC UTILITY
            Use the softkeys to make a selection.
       ----------------------------------------------

             LIF format

       =>    HFS format
```

For hard discs, the general recommendation is to use the **HFS** format. (Both formats and criteria for choosing one are further explained in the section called "Choosing a Directory Format" in the "Mass Storage Concepts" chapter.)

If the disc is already formatted, the utility will prompt:

```
Do you wish to proceed?

YES
```

If you are sure there are no valuable files on the disc, then type YES and press Return.

Step 4: Wait for Formatting to Complete

The utility will then begin formatting your disc. You will need to wait for this operation to complete. Depending on the size of your disc, the formatting can take from five to fifty minutes. The larger the disc, the longer the formatting time.

When formatting is complete, something like:

```
:CS80, 702 has been formatted.
```

will appear near the top of the screen.

Step 5: Verify the Format

After formatting is complete, you can list the catalog of the (empty) disc. First, exit the utility by returning to the main menu (press **MainMenu**) and exit the utility (press **Done**).

For example:

```
CAT [Return]
  or
CAT ":,702"  [Return]
```

You should see a catalog with two directories: `lost+found` and `WORKSTATIONS`.

Return to the fold-out table or the next step. (In most cases the next step is to "Store the System and Binaries".)

3. Storing the System and Binaries

This option stores the BASIC system configuration currently in the computer (the "core" system, and all *currently loaded* "driver" and "language extension" binaries). It also copies all binaries from the *BASIC Language Extensions* and *BASIC Drivers* discs onto the hard disc.

If you are sharing BASIC with an already existing HP-UX system, you need to be sure the root is write enabled. Refer to the first paragraph in "Steps In Loading BASIC" in Chapter 1.

Why Do This?

The main reason for storing the system and binaries on a hard disc is to make the booting process faster and simpler; for instance:

- You want to keep from having to swap discs during the "autostart" program.
- You want to begin using your hard disc because it is larger and faster.

Prerequisites

There are only two things you need to do before choosing to use this option:

- Make sure that you have **enough disc space** to store the BASIC system and all binaries. All hard discs have sufficient capacity, as long as there is adequate "unused space" on the disc (as shown in the subsequent calculation).
- **If you have used this BASIC system before**, and know which binaries you will need, you may want to take time now to verify that you have the appropriate set of binaries for your peripheral devices and programming tasks. This will save you having to manually load missing binaries later. (See the chapter called "Customizing Your BASIC System" chapter for details.)

Note

If the disc is not formatted, the utility will offer you the opportunity to format it. If the disc contains any files you do not have copies of elsewhere, you need to make backup copies of these files; see "Step 1" in the "Reformatting a Hard Disc" section above.

Here is an example of determining whether you have **enough disc space**:

1. Calculate the amount of the memory required to store the current BASIC system.

   ```
   Memory before booting          780 000   Display at boot time

   Memory after booting         - 310 000   Use SYSTEM$("AVAILABLE MEMORY")
                                ----------
   Memory required for BASIC      470 000
   ```

2. Calculate the amount of disc space currently available.

   Available space = 256 × **AVAILABLE SECTORS**   *Obtained from* **CAT**
   = 256 × 9900
   = 2 534 400

3. Decide whether or not the system will fit on the disc.

   In the above example, the memory required for BASIC (470 000 bytes) is less than the available space (2 534 400 bytes), so the system will fit on the disc.

Using the System Disc Utility

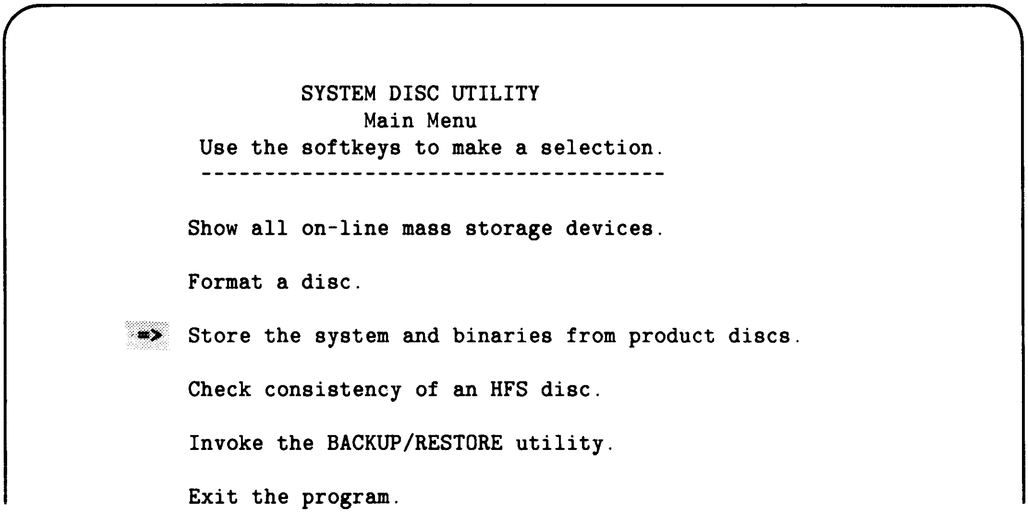
You can use either the System Disc Utility or the STORE SYSTEM and COPY statements to put BASIC on a hard disc. This section only discusses how to use the utility (see the STORE SYSTEM and COPY statements in the *BASIC Language Reference* for details of the other method).

Step 1: Load and Run the System Disc Utility

Do this step if you do not have the utility still in memory. Insert the *HFS Utilities* disc into the default drive, and load the utility by typing:

```
LOAD "DISC_UTIL" Return
```

Run the program by pressing RUN or by executing a RUN command from the keyboard, then press CONTINUE when you are instructed to in the legend screen. You should see the following display:

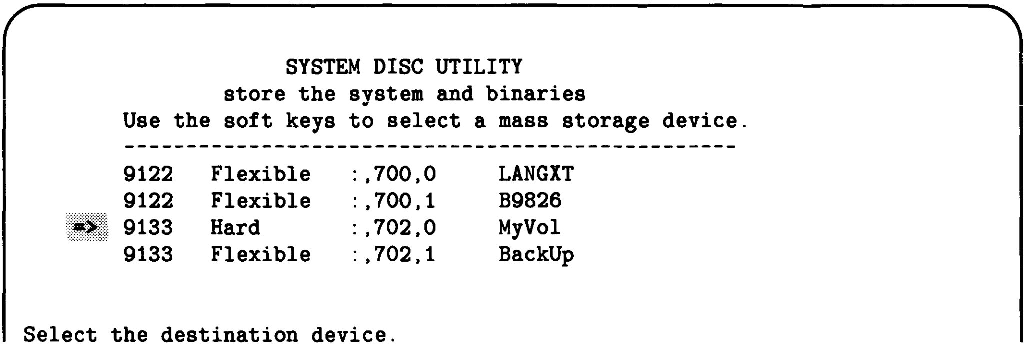


```
                    SYSTEM DISC UTILITY
                         Main Menu
                Use the softkeys to make a selection.
                -------------------------------------

               Show all on-line mass storage devices.

               Format a disc.

          =>   Store the system and binaries from product discs.

               Check consistency of an HFS disc.

               Invoke the BACKUP/RESTORE utility.

               Exit the program.
```

Step 2: Insert BASIC Language Extensions and Drivers disc into disc drive.
All of the binaries are on one disc with double-sided, 3½-inch media labeled *BASIC Language Extensions and Drivers* (DISC TWO).

Step 3: Select the "Store system and binaries" Option

From the main menu of the System Disc Utility, select the "`Store system and binaries from the product discs`" option. You should now see a display something like this:



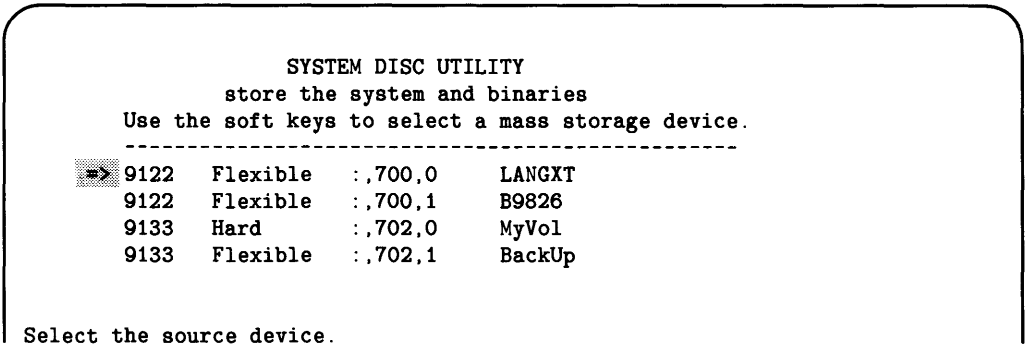
```
                   SYSTEM DISC UTILITY
              store the system and binaries
      Use the soft keys to select a mass storage device.
      -------------------------------------------------
      9122   Flexible   :,700,0     LANGXT
      9122   Flexible   :,700,1     B9826
  =>  9133   Hard       :,702,0     MyVol
      9133   Flexible   :,702,1     BackUp


Select the destination device.
```

Step 4: Select a Destination Device

Choose the destination disc (onto which you want to store the BASIC system and binary files, hard disc for example).

Step 5: Select a Source Device

The utility now shows something like the following display:



```
                   SYSTEM DISC UTILITY
              store the system and binaries
      Use the soft keys to select a mass storage device.
      -------------------------------------------------
  =>  9122   Flexible   :,700,0     LANGXT
      9122   Flexible   :,700,1     B9826
      9133   Hard       :,702,0     MyVol
      9133   Flexible   :,702,1     BackUp


Select the source device.
```

Choose the source drive (from which you want to copy the BASIC binary files). This will be a floppy drive, into which you inserted the *BASIC Language Extensions* and *BASIC Drivers* discs.

Step 6: Wait for the Operation to Complete

After switching to DISC THREE when prompted, wait until you see:

```
SYSTEM and BINARIES have been stored.
```

Select **MainMenu** (f8), then select "`Exit the program`" or **Done**.

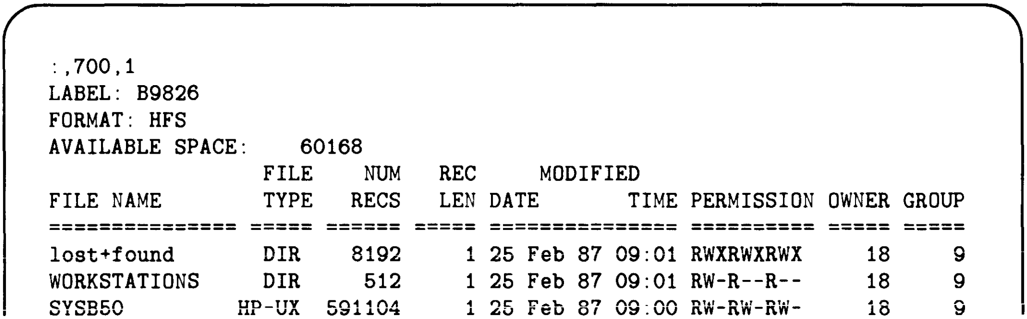
Step 7: Catalog the Destination Volume

To verify that the system has been stored on the specified disc, execute a CAT statement on this disc. For instance:

```
CAT ":,702"  Return
```

You should see the SYSB50 file in the directory.

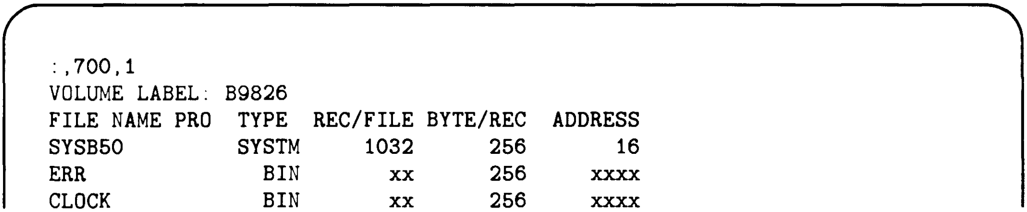
Here are typical results of this operation for an HFS disc:



```
:,700,1
LABEL: B9826
FORMAT: HFS
AVAILABLE SPACE:    60168
                   FILE    NUM    REC      MODIFIED
FILE NAME          TYPE   RECS    LEN DATE         TIME PERMISSION OWNER GROUP
=============== ===== ====== ===== ============== ========== ===== =====
lost+found         DIR    8192      1 25 Feb 87 09:01 RWXRWXRWX     18     9
WORKSTATIONS       DIR     512      1 25 Feb 87 09:01 RW-R--R--     18     9
SYSB50           HP-UX  591104      1 25 Feb 87 09:00 RW-RW-RW-     18     9
```

(The binaries are stored in the "/WORKSTATIONS/BIN5.0" directory.)

Here are typical results of this operation for a LIF disc:



```
:,700,1
VOLUME LABEL: B9826
FILE NAME PRO  TYPE  REC/FILE BYTE/REC  ADDRESS
SYSB50         SYSTM     1032      256       16
ERR              BIN       xx      256     xxxx
CLOCK            BIN       xx      256     xxxx
```

Step 8: Test the New System Configuration

Verify the new system file works by loading from the hard disc. Use either of the following methods:

- Execute a SYSBOOT statement (press the space bar following this command if you have multiple systems).

```
SYSBOOT  Return
   or
SYSBOOT "SYSB50:,702"  Return
```

- Turn the computer off and then back on again.

If you have multiple systems on-line and are not sure how to specify a system during the booting process, see the "Loading BASIC" chapter, Step 6.

4. Create directories (optional)

When you restore your files to the new file system, you may want to place the files in directories. This will let you catalog your files in an orderly fashion (you will need to reference the "Backing Up Files" chapter for details on how to put specific files in specific directories). See the "Using Directories and Files" chapter in the "Creating Directories" section for details of how and where to create directories.

In order to create directories, you will need to exit the File BACKUP and RESTORE Utility. You need to execute the **CREATE DIR** command to create a directory. Here are some examples of creating directories:

```
CREATE DIR "/USERS/MARK"
CREATE DIR "/USERS/DAVE:,700"
CREATE DIR "/WORKSTATIONS/OLD_BINS:,702,1"
```

If the hard disc is not the default drive, you will need to add the volume specifier to the end of the statement, like the second and third example.

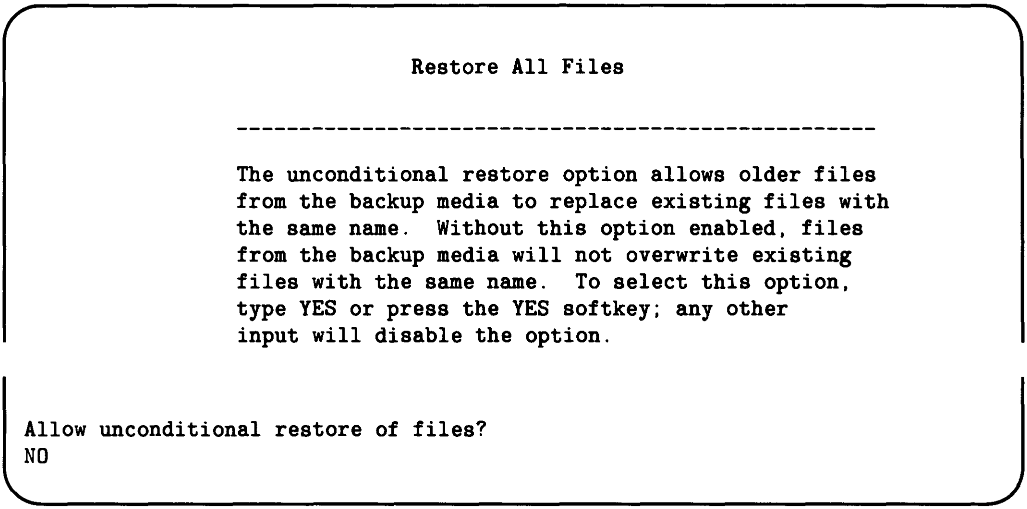
5. Restore the System

This section is applicable **only** if you backed-up your system in step 1 (which means you are reformatting a hard disc).

If you created directories, re-enter the BACKUP utility by loading it (`LOAD "BACKUP"` [Return]) and typing `RUN` [Return] or press [RUN].

Place the backup media in the drive (tape or drive).

Select the "`Restore all files from the backup media`" option. The utility gives the following prompt:



```
                         Restore All Files

                  ---------------------------------------------------

                  The unconditional restore option allows older files
                  from the backup media to replace existing files with
                  the same name.  Without this option enabled, files
                  from the backup media will not overwrite existing
                  files with the same name.  To select this option,
                  type YES or press the YES softkey; any other
                  input will disable the option.


Allow unconditional restore of files?
NO
```

Note: This example assumes that you have *already* created the directories on the hierarchical volume. If not, create them now. (Refer to "Creating Directories" in Step 4, or the "Using and Managing Files" chapter for details.)

1. Your answer to the above prompt indicates whether or not files on the backup media are to **replace** files of the same name on the media being restored.

   - NO indicates that files on the backup media are **not** to replace any files with the **same name** on the media being restored.
   - YES indicates that files on the backup media **are** to replace any files with the **same name** on the media being restored (the existing files are overwritten, which destroys any data they contain).

   For this example, we will assume that you **don't** want to automatically restore all files (without a chance to confirm the replacement), so press the softkey labeled NO or press Return.

2. The utility then presents this prompt:

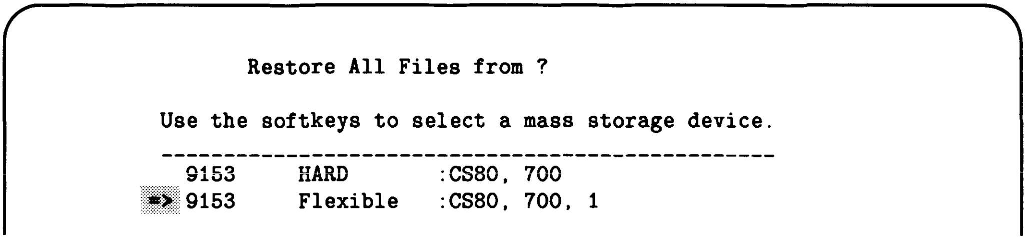
```
Allow renaming?
NEITHER
```

   Your answer to this prompt indicates whether or not you want the option of *renaming each file and directory* that is being restored from the backup media:

   - NEITHER indicates that files on the backup media are **not** to be renamed (all files will retain current name and directory location).
   - YES indicates that you want the **option of renaming** files on the backup media as they are being restored (you will be prompted for each file's name and directory location).
   - D indicates change destination paths and volume specifiers only.
   - F indicates change both destination paths and individual file names.

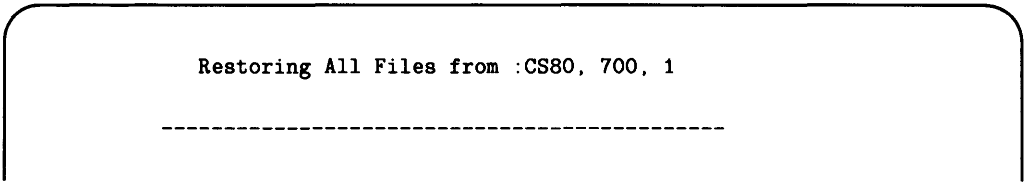
   In this example, we will restore the files to the same locations from which they came; so press the softkey labeled NEITHER.

3. The utility then prompts you to specify the backup media:



```
          Restore All Files from ?

      Use the softkeys to select a mass storage device.
      ------------------------------------------------
        9153     HARD        :CS80, 700
     => 9153     Flexible    :CS80, 700, 1
```

4. The utility then copies the specified files from the source onto the backup media.



```
        Restoring All Files from :CS80, 700, 1


      --------------------------------------------
```

(If the back up media overflows, the utility will prompt you to install a different backup disc or tape). When the backup is complete, the utility returns to the main menu.

5. Exit the Backup/Restore utility and **return to the fold-out table to see what your next step is.**

Formatting with INITIALIZE

This section describes how to format discs using the INITIALIZE statement as a keyboard command. Follow this section **ONLY IF YOU WERE SENT HERE BY STEP 2** ("Formatting a Hard Disc").

- If you have already formatted the disc, using the utility described in the preceding sections, then you should skip this section.
- If you don't need to select non-default *interleave factor* or *sector/volume size*, then it is easier to use the `Format a disc` option of the System Disc Utility. Go back to the beginning of this chapter to see which section to read (step 2).

CAUTION

FORMATTING DESTROYS ALL DATA AND PROGRAMS STORED ON A DISC. BE CERTAIN THAT YOU DO NOT RE-FORMAT A BASIC SYSTEM DISC, OR ANY OTHER DISC CONTAINING VALUABLE FILES.

IF THERE ARE FILES ON THE DISC THAT YOU WANT TO KEEP, THEN FIRST MAKE A BACK-UP COPY OF EACH BEFORE RE-FORMATTING THE DISC. SEE "Backing Up Files" IN THE CHAPTER CALLED "Maintaining Your BASIC System" FOR DETAILS.

IF FORMATTING A HARD DISC, DO NOT INTERRUPT THE FORMATTING PROCESS, AS THIS MAY SEVERELY DAMAGE THE DRIVE.

Note

You may not need to re-format your disc if has been previously formatted by the Series 200/300 Pascal or HP-UX systems. For further information, see the section called "Hard Disc Partitioning" in the "Mass Storage Concepts" chapter.

Here is the syntax of the INITIALIZE statement. The parameters are briefly explained below, and then several examples are shown.

`INITIALIZE " :` *volume specifier*`" ,` *interleave* `,` *sector/volume size*

volume specifier identifies your disc. (The volume specifier should be labeled on stickers affixed to the front of all drives in your system; if not, consult the chapter called "Verifying and Labeling Peripherals".)

interleave specifies how the sectors will be numbered. (Interleave is discussed in "Disc Interleave" in the "Mass Storage Concepts" chapter.) If you are not sure about what interleave to use, then specify the default value for your particular disc (0 also specifies the default value). *Note that the interleave of some hard discs cannot be changed.*

Table 5-1. Default Interleaves

Flexible Disc Type	Typical Product Numbers	Default Interleave
3½-inch double-sided (gray)	HP 9122, 9133D, 9153	2
3½-inch single-sided	HP 9121, 9133A	2
5¼-inch internal	Built-in drives of HP 9826 and 9836 computers	1
5¼-inch external	HP 82901, 82902, 9134, 9127	3

sector/volume size specifies the size of sectors and the volume on *double*-sided, 3½-inch, flexible discs only; omit this parameter or use the default value or 0 with all other discs. (See the "Disc Sectors" section of the "Mass Storage Concepts" for further explanation.)

Table 5-2. Size Parameter

Parameter Value	Sector Size (Bytes)
0 (default)	*device-dependent*
1	256
2[1]	512
3	1024
4	256

Examples

The following statement formats the disc on select code 7, with primary address 0 and unit number 1. The disc is formatted with the default *interleave factor* (device-dependent) and default *sector/volume size* option (also device-dependent).

```
INITIALIZE ":,700,1",0,0
```

(Since the defaults for interleave factor and sector/volume size option are 0, this is the same as `INITIALIZE ":,700,1".`)

The following example formats the same disc with an interleave factor of 2 and sector/volume size option 3.

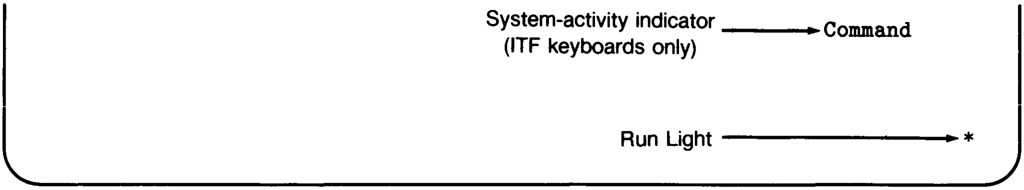
```
INITIALIZE ":,700,1",2,3
```

This last example formats the disc at select code 8, with primary address 2 and unit number 0. The disc is initialized with an interleave factor of 3 and sector/volume size option 4.

```
INITIALIZE ":,802",3,4
```

[1] Not supported.

The formatting process will continue for a few minutes, as indicated by the asterisk (*) in the lower-right corner of the screen. (The `Command` annunciator is present if in KEY LABELS ON mode.)



When this indicator is turned off (`Idle` will be displayed if in KEY LABELS ON mode), the process has completed.

Formatting times can last from five to fifty minutes, depending on the size of your disc. The larger the disc, the more time you should expect for formatting.

6. What To Do Next

Now that you have installed BASIC, you are ready to use the system. Your next task can be any one of the following:

Task/Topic	Chapter
Verify the peripherals other than the mass storage units, if you have not already done so.	2
Check or modify your system's configuration.	7
Learn how to load and run programs.	9
Learn how to create directories.	10
Learn how to use and manage files.	10
Learn how to enter, edit, secure, document and store programs.	11
Learn how to maintain your system.	12

Putting BASIC on the SRM 6

If you have a Shared Resource Manager (SRM) system, you will probably want to install BASIC on it so that you can boot the system from the SRM and keep files on it. This chapter describes this installation operation.

Prerequisites

You must have already installed and configured your SRM system, and have your workstation connected to it (via the HP 98629 SRM interface and associated cabling).

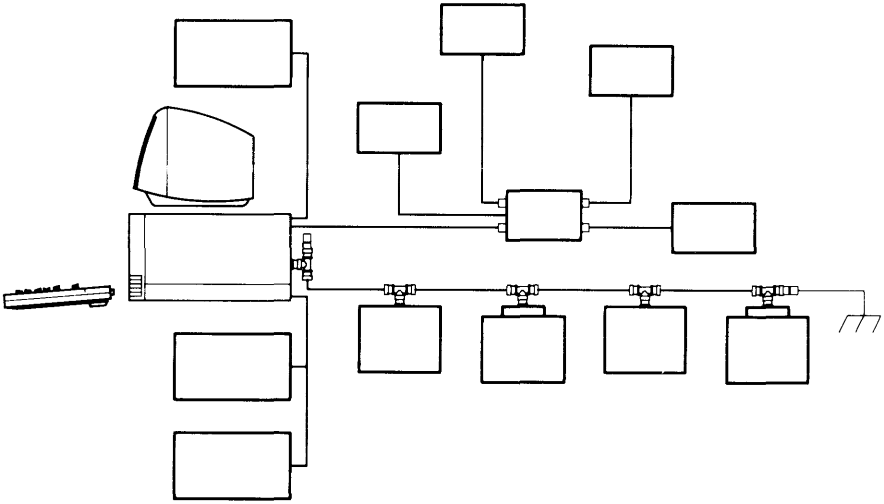


Figure 6-1. SRM Configuration Example

If you do not have this equipment already set up, then locate the SRM documentation and install it now.

Is the SRM Set Up?

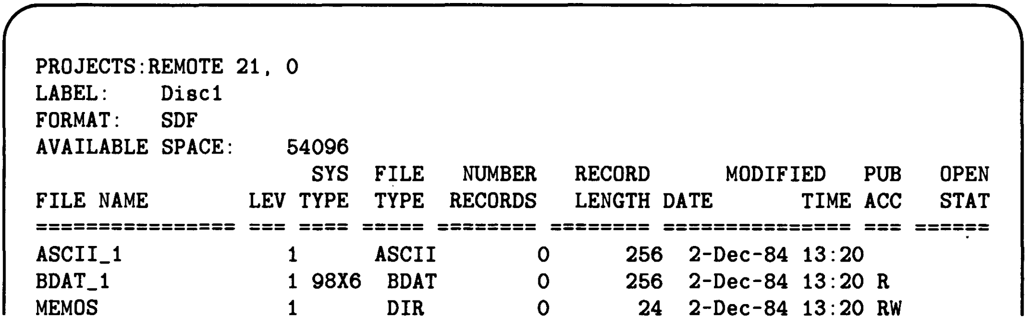
In order to use the SRM system, you must have already set it up as described in the SRM documentation. Once set up, you can install the BASIC system on it.

To see whether the SRM system is ready for use with BASIC, you can list the catalog of the disc.

Execute a CAT statement. Here are typical CAT statements for looking at the SRM's "root" directory:

```
CAT ":REMOTE,21"
      or
CAT ":,21"
```

Here are typical results of this operation:



```
PROJECTS:REMOTE 21, 0
LABEL:    Disc1
FORMAT:   SDF
AVAILABLE SPACE:    54096
                      SYS  FILE   NUMBER   RECORD      MODIFIED   PUB   OPEN
FILE NAME        LEV TYPE  TYPE  RECORDS   LENGTH DATE       TIME ACC   STAT
================ === ==== ===== ======== ======== =============== === ======
ASCII_1             1      ASCII        0      256  2-Dec-84 13:20
BDAT_1              1 98X6  BDAT        0      256  2-Dec-84 13:20 R
MEMOS               1       DIR         0       24  2-Dec-84 13:20 RW
```

To see what each column means, see Chapter 10.

Store the System and Binaries

This option stores the BASIC system configuration currently in the computer (the "core" system, and all *currently loaded* "driver" and "language extension" binaries). The main reason for using this option is to make the booting process faster and simpler; for instance:

- You want to keep from having to swap discs during the "autostart" program.
- You want to begin using your SRM disc because it is larger and faster.

Prerequisites

There are only two things you need to do before choosing to use this option:

- Make sure that you have a disc large enough to copy the entire system and binaries. All discs supported with the SRM system have sufficient capacity to store the BASIC system (as long as there is enough "unused space" on the disc to store BASIC, as shown in the subsequent calculation).
- You may (optionally) want to verify that you have the appropriate set of binaries for your peripheral devices and programming tasks; this will save you the effort of loading binaries later. (See the chapter called "Language Extensions, Drivers, and Configuration" chapter for details.)

Is There Room on the SRM for BASIC?

Here is an example calculation that shows the steps in determining whether or not BASIC will fit in the existing space on an SRM system:

1. Calculate the amount of the memory required to store the current BASIC system.

Memory before booting	`780 000`	*Display at boot time*
Memory after booting	`- 310 000`	*Use* `SYSTEM$("AVAILABLE MEMORY")`
	`----------`	
Memory required for BASIC	`470 000`	

2. Calculate the amount of disc space currently available.

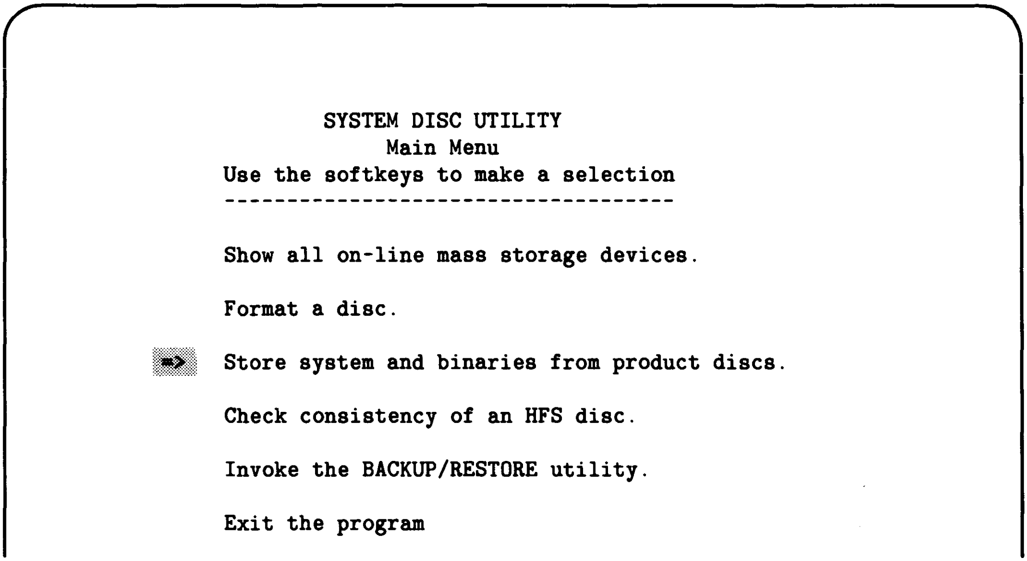
Available space = 256 × `AVAILABLE SPACE` *Shown in* `CAT` *listing*
= 256 × 60900
= 15 xxx xxx

3. Decide whether or not the system will fit on the disc.

In the above example, the memory required for BASIC (470 000 bytes) is less than the available space (15 xxx xxx bytes), so the system will fit on the disc.

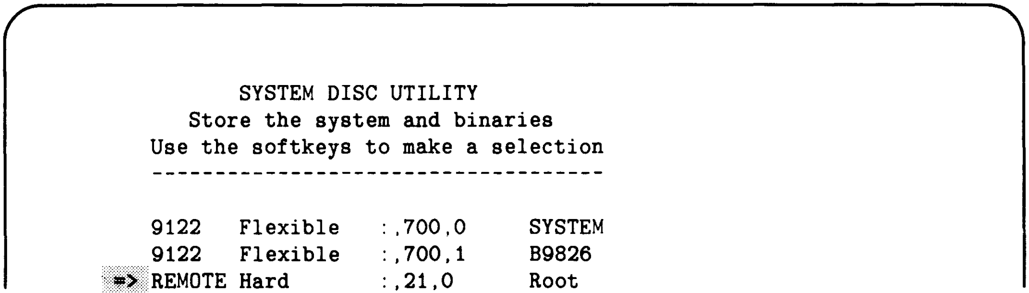
Using the “System Disc” Utility

1. From the main menu of the System Disc Utility (loaded from the *HFS Utilities* disc), select the “`Store system and binaries`” option.



```
               SYSTEM DISC UTILITY
                    Main Menu
        Use the softkeys to make a selection
        ------------------------------------

        Show all on-line mass storage devices.

        Format a disc.

  =>    Store system and binaries from product discs.

        Check consistency of an HFS disc.

        Invoke the BACKUP/RESTORE utility.

        Exit the program
```

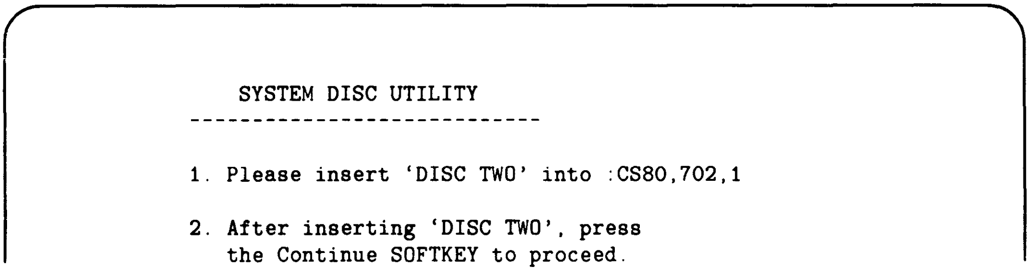
2. Select the destination disc. (If you are storing the system on a flexible disc, you should insert the disc into the drive now.)



```
               SYSTEM DISC UTILITY
            Store the system and binaries
         Use the softkeys to make a selection
         ------------------------------------

         9122   Flexible   :,700,0     SYSTEM
         9122   Flexible   :,700,1     B9826
      => REMOTE Hard       :,21,0      Root
```

3. Select the source from the next menu. This will usually be a flexible disc.

4. You should see this screen after you select the destination device:



```
               SYSTEM DISC UTILITY
            ----------------------------

            1. Please insert 'DISC TWO' into :CS80,702,1

            2. After inserting 'DISC TWO', press
               the Continue SOFTKEY to proceed.
```

Insert DISC TWO into the flexible disc drive you specified to be the source device. The language extensions, device drivers, and interface driver binaries will be loaded. After DISC TWO is copied, you will be prompted to insert DISC THREE and press Continue. You will see the prompt:

```
Storing the system.
```

After the system is stored, you can remove DISC THREE, exit the utility, and use the system. You should, however, first verify that the system was installed proplerly by following the instructions in the next section.

Verifying the Stored System

1. First verify that the system has been stored on the specified disc, by executing a CAT statement on this disc. For instance:

   ```
   CAT ":,21"
   ```

   You should see the system file in the directory.

2. Now verify that the new system file by loading and using it. The Boot ROM in your computer can find and load operating systems from the SRM system, as long as they are in the following directories.

`/SYSTEMS/SYSOpSysName`	Name of a file containing a "bootable" system.
`/SYSTEMS/SYSTEM_B5`	Typical name of the BASIC 5.0 system file.
`/WORKSTATIONS/SYSTEMnn`	Location BASIC expects to find an AUTOST program.
`/WORKSTATIONS/SYSTEM18`	Directory containing AUTOST program for node number 18.

Re-boot the system:

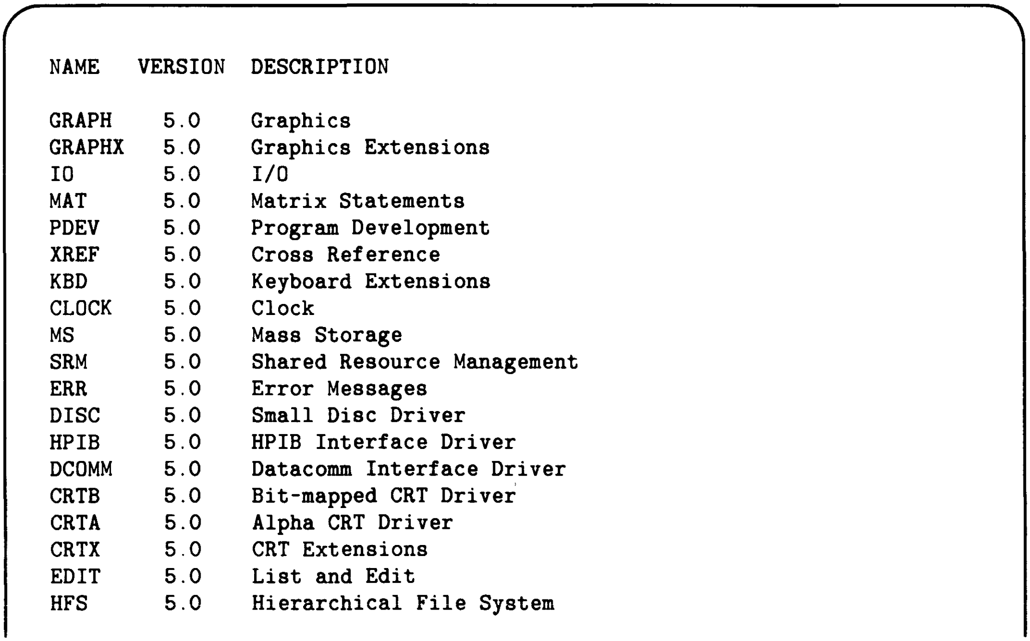
- Execute a SYSBOOT statement: `SYSBOOT "/SYSTEMS/SYSTEM_B5:,21"` `where :,21` is the volume specifier for the SRM disc.
- Or, turn the computer off and then back on again.

If you have multiple systems on-line and are not sure how to specify a system during the booting process, then see the "Loading the BASIC System" chapter.

3. Now verify that the system is loaded by executing the LIST BIN command:

   ```
   LIST BIN
   ```

   The system should respond by showing the binaries currently loaded:



```
NAME   VERSION  DESCRIPTION

GRAPH    5.0    Graphics
GRAPHX   5.0    Graphics Extensions
IO       5.0    I/O
MAT      5.0    Matrix Statements
PDEV     5.0    Program Development
XREF     5.0    Cross Reference
KBD      5.0    Keyboard Extensions
CLOCK    5.0    Clock
MS       5.0    Mass Storage
SRM      5.0    Shared Resource Management
ERR      5.0    Error Messages
DISC     5.0    Small Disc Driver
HPIB     5.0    HPIB Interface Driver
DCOMM    5.0    Datacomm Interface Driver
CRTB     5.0    Bit-mapped CRT Driver
CRTA     5.0    Alpha CRT Driver
CRTX     5.0    CRT Extensions
EDIT     5.0    List and Edit
HFS      5.0    Hierarchical File System
```

What to Do Next

Task/Topic	Chapter Number
Check or modify your system's configuration.	7
Learn how to use keyboard, display and commands.	8
Learn how to load and run programs.	9
Learn about how to use and manage files.	10
Learn about how to enter, edit, secure, document, and store programs.	11
Learn about how to maintain your system.	12-17

Notes

Language Extensions, Drivers, and Configuration 7

The BASIC system is partitioned into small components so that you can load only those that you need, thus optimizing memory and disc usage. This chapter describes the partitioning and shows you how to create an “optimal” system that contains only the components that you will need.

BASIC System Components

When the computer is turned on, it loads an operating system[1]. With BASIC (as it is shipped from the factory), this "main" component also contains many BASIC language keywords. An "autostart" program then loads a set of optional binaries[2] that allow you to use many other keywords and devices.

In this section, you will see how to determine which binaries are installed, so that you can compare this list to the list of binaries that you will need for your programming tasks and for the set of interfaces and devices you will be using.

Before you can determine whether or not your current system configuration suits your needs, you need to answer these questions:

- What BASIC system components are **available**?
- Which components do you **need**?
- Which ones are **currently loaded** in your system?

[1] For a closer look at how the system boots and configures itself, see "A Closer Look at the BASIC Booting Process" near the end of this chapter.

[2] The term "binary" is customarily used for a machine-language program file that cannot be easily read by a human.

Overview of Components

The BASIC system is divided into several components, as shown in the following drawing:

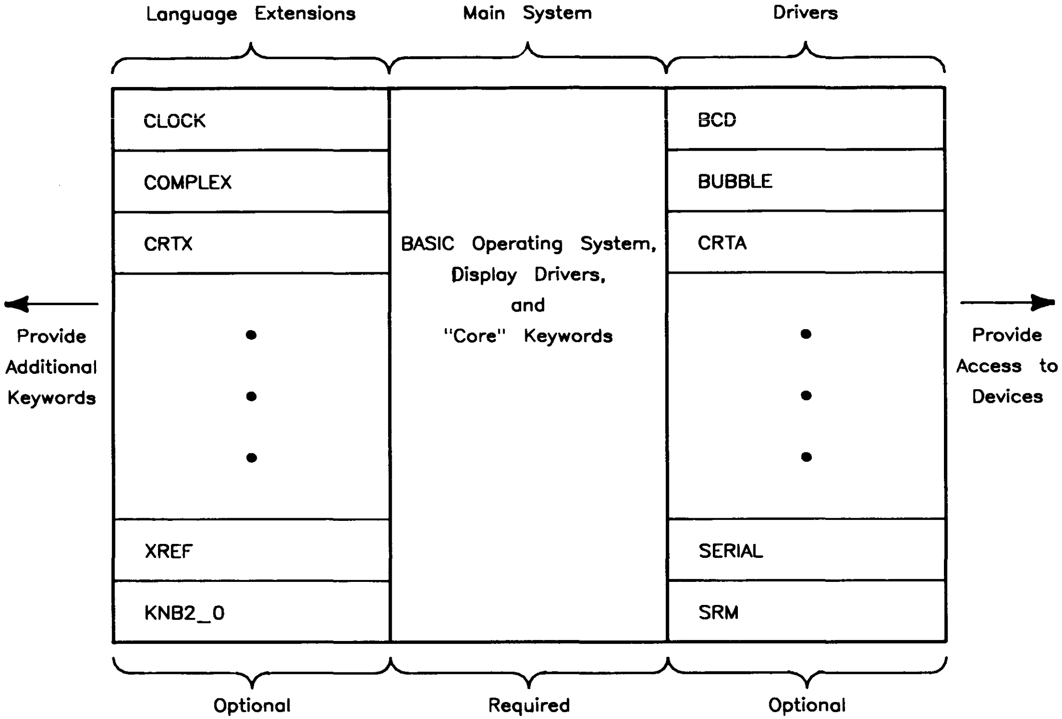

Language Extensions
Main System
Drivers
CLOCK
COMPLEX
CRTX
XREF
KNB2_0
BASIC Operating System, Display Drivers, and "Core" Keywords
BCD
BUBBLE
CRTA
SERIAL
SRM
Provide Additional Keywords
Provide Access to Devices
Optional
Required
Optional

Figure 7-1. BASIC System Components

These components fall into three classes:

- The "Main" operating system—this contains the heart of the operating system. Included are the portions of the system that execute programs and commands, as well as more than half of the available BASIC keywords.
- Language extension binaries—provide *optional* BASIC keywords.
- Driver binaries—provide *optional* routines which allow you to access interfaces (such as the HP-IB and GPIO cards), and devices (such as flexible and hard discs, and EPROM cards).

Thus, you do not have to use memory or disc space to store all of the available components; you may choose to use only those optional components required for your hardware configuration and programming tasks.

Determining When to Re-Configure

In order to determine whether or not you need to modify your current configuration—that is, to decide whether or not to load additional binaries—you will need to answer two questions:

- Which binaries are "currently loaded" in my system?
- Which binaries are "needed?"

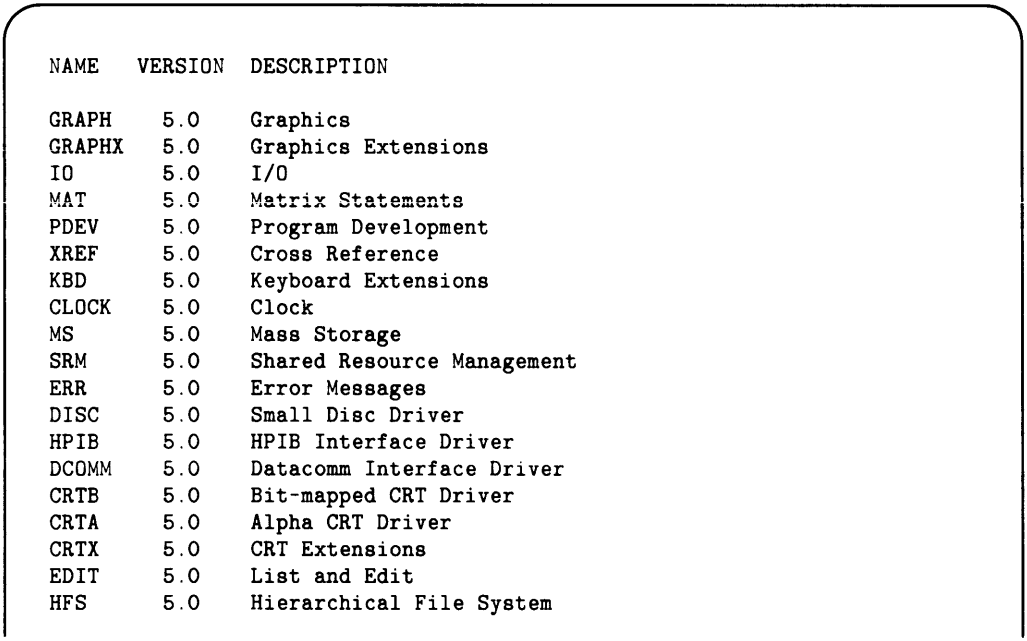
The next ten pages of this chapter show how to answer these questions.

Determining Your Current Configuration (Which Binaries Are Currently Loaded?)

You can list the language extensions and drivers currently in the system by executing this command:

```
LIST BIN [Return] or [ENTER]
```

The listing should look something like this:



```
NAME   VERSION  DESCRIPTION

GRAPH    5.0    Graphics
GRAPHX   5.0    Graphics Extensions
IO       5.0    I/O
MAT      5.0    Matrix Statements
PDEV     5.0    Program Development
XREF     5.0    Cross Reference
KBD      5.0    Keyboard Extensions
CLOCK    5.0    Clock
MS       5.0    Mass Storage
SRM      5.0    Shared Resource Management
ERR      5.0    Error Messages
DISC     5.0    Small Disc Driver
HPIB     5.0    HPIB Interface Driver
DCOMM    5.0    Datacomm Interface Driver
CRTB     5.0    Bit-mapped CRT Driver
CRTA     5.0    Alpha CRT Driver
CRTX     5.0    CRT Extensions
EDIT     5.0    List and Edit
HFS      5.0    Hierarchical File System
```

As you read through the following descriptions of the available binaries, check the **Currently Loaded?** column for each binary in the list that appears on your CRT display.

This column

Language Extension Binary	Description	Approx. Size (Kbytes)	Currently Loaded?	Needed?
CLOCK	Provides ...	4		
COMPLEX	Provides ...	7		
CRTX	Provides ...	2		

Which Binaries Do You Need?

In order to determine whether or not you need a binary, you will have to know what it can provide for you. In general, there are two types of binaries:

- Those needed for your particular programming tasks (language-extension binaries)
- Those needed for your particular set of interfaces and devices (driver binaries)

The next sections describe all of the available binaries in each of these two categories. As you read through the list, place a check in the "Needed?" column if you need the binary.

This column

Language Extension Binary	Description	Approx. Size (Kbytes)	Currently Loaded?	Needed?
CLOCK	Provides ...	4		
COMPLEX	Provides ...	7		
CRTX	Provides ...	2		

Language Extensions Available

Language extensions provide additional keywords that you can use for programming tasks. If you need at least one of the keywords in a language extension, then you will need to LOAD BIN that language extension file. Each language extension in the following list is a file located on the *Language Extensions* disc (*Language Extensions and Drivers* disc with Media Option 045).

If you cannot decide whether you need a language extension file after reading the brief descriptions provided in this section, pick up the *BASIC Language Reference* and read about the statements you think you might need. There you will find detailed information on the function of each statement.

Language Extension Binary	Description	Approx. Size (Kbytes)	Currently Loaded?	Needed?

`CLOCK`	Provides additional access to clock and periodic timers (TIMEDATE is in the "Main" system).
	`DATE` `OFF DELAY`
	`DATE$` `ON TIME`
	`ON CYCLE` `OFF TIME`
	`OFF CYCLE` `TIME`
	`ON DELAY` `TIME$`
		4		

`COMPLEX`	Provides complex math and hyperbolic trigonometric functions.
	`ARG` `ACSH` `COSH`
	`CMPLX` `ASNH` `SINH`
	`COMPLEX` `ATNH` `TANH`
	`CONJG` `REAL`
	`IMAG`
		7		

Language Extension Binary	Description	Approx. Size (Kbytes)	Currently Loaded?	Needed?
CRTX	Provides several CRT display "eXtended" capabilities. ALPHA PEN ALPHA HEIGHT ALPHA MASK CHRX CHRY SET CHR DISPLAY FUNCTIONS DISPLAY MASK KEY LABELS KEY LABELS PEN KBD LINE PEN PRINT PEN	2		
EDIT	Provides the BASIC program editor/lister, as well as some "global editing" keywords. (Note that in previous revisions of BASIC, this was in the Main system.) The following statements are enabled: EDIT LIST SAVE MOVELINES COPYLINES CHANGE FIND INDENT	22		
ERR	Provides a textual description of error messages, instead of just the error number.	9		

Language Extension Binary	Description	Approx. Size (Kbytes)	Currently Loaded?	Needed?
GRAPH	Provides the following “fundamental” graphics keywords: ALPHA ON ALPHA OFF AXES CLIP CSIZE DRAW DUMP DEVICE IS DUMP GRAPHICS FRAME GCLEAR GINIT GLOAD GRAPHICS OFF GRAPHICS ON GRID GSTORE IDRAW IMOVE IPLOT LABEL LDIR LINE TYPE LORG MOVE PEN PEN UP PDIR PIVOT PLOT SHOW VIEWPORT WINDOW SEPARATE ALPHA MERGE ALPHA GSEND	46		
GRAPHX	Provides the following “extended” graphics capabilities: AREA COLOR AREA INTENSITY AREA PEN DIGITIZE GESCAPE GRAPHICS INPUT IS IPLOT(*) PLOT(*) POLYGON POLYLINE READ LOCATOR RECTANGLE RPLOT(*) SET ECHO SET LOCATOR SET PEN COLOR SET PEN INTENSITY SYMBOL TRACK IS ON TRACK IS OFF WHERE	29		

Language Extension Binary	Description	Approx. Size (Kbytes)	Currently Loaded?	Needed?
IO	Provides the following keywords for general device input/output (I/O): ABORT BREAK CLEAR DISABLE INTR ENABLE INTR LOCAL LOCAL LOCKOUT ON INTR OFF INTR ON SIGNAL OFF SIGNAL PASS CONTROL PPOLL CONFIGURE PPOLL RESPONSE PPOLL UNCONFIGURE REMOTE REQUEST RESET SEND SIGNAL TRIGGER	11		
KBD	Provides the following keywords for enhanced control of the keyboard; in addition, it also contains drivers for the HP Human Interface Link (HIL) devices (such as mouse, TouchScreen, ID Module, etc.): CDIAL EDIT KEY HILBUF$ HIL SEND KBD CMODE LIST KEY LOAD KEY OFF CDIAL ON CDIAL OFF HIL EXT ON HIL EXT RE-STORE KEY SCRATCH KEY SET KEY SOUND STORE KEY SYSTEM MENU USER MENU	20		

Language Extension Binary	Description	Approx. Size (Kbytes)	Currently Loaded?	Needed?
LEX	Provides one keyword for changing collating sequences; in addition, it is required when using non-US-ASCII keyboards: `LEXICAL ORDER IS`	10		

MAT	Provides keywords for handling arrays (and matrices):
	`BASE` `MAX`
	`DET` `MIN`
	`DOT` `RANK`
	`MAT` `REDIM`
	`MAT REORDER` `SIZE`
	`MAT SEARCH` `SUM`
	`MAT SORT`
		40		

MS	Provides extensions to the CAT statement, as well as the following mass storage keywords:
	`CHECKREAD` `READ LABEL`
	`CREATE` `PRINT LABEL`
		10		

PDEV	Provides keywords which are used during program development:
	`CHANGE` `LOADSUB FROM`
	`COPYLINES` `SECURE`
	`FIND` `TRACE ALL`
	`INDENT` `TRACE OFF`
	`MOVELINES` `TRACE PAUSE`
	Note that CHANGE, COPYLINES, FIND, INDENT, and MOVELINES also require the EDIT binary.
		14		

Language Extension Binary	Description	Approx. Size (Kbytes)	Currently Loaded?	Needed?
TRANS	Provides advanced transfer capabilities which utilize buffers and "background-process" transfers: ABORTIO ON EOR OFF EOR ON EOT OFF EOT TRANSFER WAIT FOR EOR WAIT FOR EOT	32		
XREF	Provides one keyword which is used to obtain a cross reference of identifiers used in a program or subprogram: XREF	7		
KNB2_0	Causes the KNOBX function to operate the way that it did in revisions 1.x and 2.x of the BASIC system. (It is recommended that you **not** use this binary.) Refer to the "Porting to 3.0" chapter of *BASIC Programming Techniques* for further information.	1		

Drivers Available

Drivers are programs that allow you to "talk to" interfaces and devices. Here are the drivers available with the BASIC 5.0 system. They are located on the *BASIC Drivers* disc (with Media Option 045, they are on the *BASIC Language Extensions and Drivers* disc.)

Language Extension Binary	Description	Approx. Size (Kbytes)	Currently Loaded?	Needed?
BCD	Provides a driver for the HP 98623A BCD (Binary-Coded Decimal) interface.	3		
BUBBLE	Provides a driver for the HP 98259A Magnetic Bubble Memory card.	4		
CRTA	Provides a driver for the Series 200 displays which have *separate* alpha and graphicsi rasters (also called "non-bit-mapped-alpha" displays). This driver is a part of the "core" system as shipped from the factory (i.e., the SYSTEM_BA5 file on the *BASIC System* disc).	5		
CRTB	Provides a driver for Series 200 (Models 217 and 237) and Series 300 displays which have alpha and graphics on the *same* raster (also called "bit-mapped-alpha" displays). This driver is also a part of the "core" system as shipped from the factory (i.e., the SYSTEM_BA5 file on the *BASIC System* disc).	10		
CS80	Provides a driver for "Command Set/80" type discs (most hard discs) and "Sub-Set/80" type discs (*double*-sided 3½-inch flexible discs such as the HP 9122). Note that either the HPIB or FHPIB driver is also required in order to access these discs.	9		
DCOMM	Provides a driver for the HP 98628 Datacomm Interface, and for SRM Interfaces. (If loading from an SRM system, then the SRM binary must be loaded **before** the DCOMM binary.)	6		

Language Extension Binary	Description	Approx. Size (Kbytes)	Currently Loaded?	Needed?
DISC	Provides a driver for "non-CS80" disc drives (5¼-inch flexible discs such as the HP 82901, and HP 9135 discs; and the HP 9121 single-sided 3½-inch flexible disc).	7		
EPROM	Provides a driver for the HP 98255 EPROM (Erasable, Programmable, Read-Only Memory) card.	3		
FHPIB	Provides a driver for the HP 98625 High-Speed Disc Interface.	3		
GPIO	Provides a driver for the HP 98622 GPIO (General-Purpose Input and Output) Interface.	6		
HFS	Provides the device driver for Hierarchical File System (HFS) volumes. Also provides HFS keywords (CHGRP, CHOWN, and PERMIT), and is required to use CREATE DIR on an HFS volume.	53		
HP9885	Provides a driver for the HP 9885 8-inch disc drive. (Note that the GPIO driver is also required, since this disc is connected to the computer through the GPIO interface.)	4		
HPIB	Provides a driver for the built-in HP-IB interfaces, and for the HP 98624 optional HP-IB interface.	12		
SERIAL	Provides a driver for the HP 98626 and 98628 RS-232C Serial Interfaces.	5		
SRM	Provides the driver for using the Shared Resource Manager (SRM) system. (When loading from an SRM system, the SRM driver must be loaded **before** the DCOMM driver.) Also provides SRM keywords (LOCK and UNLOCK), and is required to use CREATE DIR on an SRM volume.	46		

Making the Re-Configuration Decision (Should You Re-Configure?)

As you scanned the preceding descriptions of available binaries, you should have checked the **Needed?** and **Currently Loaded?** columns. If not, you should do so now.

Compare these two columns

Language Extension Binary	Description	Approx. Size (Kbytes)	Currently Loaded?	Needed?
CLOCK	Provides ...	4		
COMPLEX	Provides ...	7		
CRTX	Provides ...	2		

After checking the list of binaries in the preceding sections, you can compare the set of **Currently Loaded?** binaries to the set of **Needed?** binaries.

- **If the lists match:** you don't need to modify your system's configuration. Skip the rest of this chapter—if you wish—and proceed to your next task (loading and running programs, using files, etc.).
- **If the lists don't match:** you have two possibilities:
    - If there are **more** binaries already loaded than you need, you can:
        - Re-configure now, or
        - Put it off until you learn more about the system.
    - If there are **fewer** binaries than you need, you will need to load the ones you need before proceeding.

The procedures for deleting and loading binaries are discussed in the next section.

Methods of Creating a Custom Configuration

At this point, you should have already decided that you *do* need or want to modify your system's current configuration. (If you have not made this decision, see the preceding section called "Do You Need to Modify the Configuration?")

Here are the two general methods of creating customized systems that contain only the components that you need:

- **System-File Method:** Delete and load the desired components by typing commands from the keyboard. If you have a double-sided, 3½-inch micro-disc or hard disc, then you can store this configured system in one "system file" that can be loaded automatically whenever you turn the computer on[1]. (This is the method used in the "Putting BASIC on a Hard Disc" chapter.)
- **Autostart-Program Method:** Create your own autostart program (or modify the supplied program) that automatically loads the desired components whenever you turn the computer on[1].

Recommendations for Choosing a Method

The "general" recommendation is to the system-file method if you have enough disc space, because:

- System files load binaries a little more quickly than an Autostart program can.
- When using a system files, you do not need to swap discs.
- System files (that contain all the desired binaries) also run a little faster on a properly configured Model 310 computer.

[1] For an explanation of how these configurations are performed, see "A Closer Look at the BASIC Booting Process" near the end of this chapter.

However, storing a complete custom BASIC system in a system file can require a lot of disc space. For this reason, you **probably cannot** store a custom system file on a single-sided flexible disc. A hard disc or double-sided 3½-inch flexible disc will usually be required. If a STORE SYSTEM command results in the message:

```
ERROR 64  Mass storage medium overflow
```

then you don't have enough room on the disc to store the system. In such instances, you will have to use the Autostart file approach.

Conversely, you may want to have an Autostart program that performs other configurations (such as changes the screen height, specifies a sub-directory as the current default volume, etc). In such cases, you may want to just use the Autostart program to load the binaries, since it eliminates the effort required to also create a custom system file.

Creating a Custom System File

This section discusses how to load binaries into your BASIC system and then store the system in a "system" file, which is loaded by the Boot ROM in a single operation. The next section, called "Creating an Autostart Program", describes how to configure your system with an Autostart program; it starts on page 7-19.

Note

If you have a non-U.S. ASCII Keyboard, read the "Non-U.S. ASCII Keyboard" section near the end of this chapter before beginning this procedure. If you don't follow the instructions in that section, you will not be able to type in the commands given in this section.

Condensed Procedure

Here is an overview of the steps in creating and storing a custom system file. (Each step is fully explained in subsequent sections.)

1. Delete all existing binaries by executing:

   `SCRATCH BIN` [Return] or [ENTER]

2. Load the appropriate binaries by executing LOAD BIN; for instance, this example loads the ERR binary from the default drive:

   `LOAD BIN "ERR"` [Return] or [ENTER]

3. Store the system on a disc (must have sufficient unused space) with the STORE SYSTEM command:

   `STORE SYSTEM "SYS_B5"` [Return] or [ENTER]

4. Verify that the system works and contains the desired binaries by re-booting with this new file (turn the computer off and the on, or use the SYSBOOT command).

Subsequent sections explain each step in greater detail.

Step 1: Deleting Binaries

If there are any unwanted binaries in the system, and you want to use the memory they occupy for other purposes, then you can delete the binaries with this statement:

```
SCRATCH BIN [Return] or [ENTER]
```

Note, however, that the SCRATCH BIN statement deletes **all** binaries—except the one for the currently active display device. Therefore, you will need to re-load all of the binaries that you do want, and then load the additional ones that you were going to add in the first place. (Loading binaries is described in the next section, "Step 2: Loading Binaries".)

Note About CRT Binaries

If you have a Series 200 computer with a non-bit-mapped alpha display, you may skip this section.

There are only two drivers in the "core" BASIC system (that is, the **SYSTEM_BA5** file as shipped from the factory):

- CRTA—allows BASIC to use a non-bit-mapped-alpha display (only Series 200 computer Models 216, 217, 220, 226, 236, and the HP 98546 Display Compatibility Interface for Series 300 computers).
- CRTB—allows BASIC to use a bit-mapped-alpha display (Series 300 computers, and Series 200 Model 237 computers).

If you have a Series 300 computer with a bit-mapped-alpha display as well as an HP 98546 Display Compatibility Interface, then you will need both CRT drivers[1]. Otherwise, if you have a Series 300 computer with **only one** type of display, then you can delete the CRT driver that you don't need. Type:

```
SCRATCH BIN [Return]
```

This removes the unneeded CRT driver. For instance, it removes **CRTA** but leaves **CRTB** if you have only a bit-mapped-alpha display.

[1] You can also make a system file that contains only the CRTA driver, for the Display Compatibility Interface, and still use the Series 300 display for graphics.

Note about Using "Non-U.S. ASCII" Keyboards

The term "non-U.S. ASCII" indicates that the keyboard is localized for a country other than the United States. However, it is still an ASCII keyboard, because it produces ASCII codes which the BASIC operating system interprets as ASCII characters.

The non-U.S. ASCII keyboard operates as a U.S. ASCII keyboard until the LEX binary on the *Language Extensions Disc* is loaded. The characters displayed correspond to characters on the U.S. keyboard keys without regard for the legend on the non-U.S. keyboard keycaps.

Note that the AUTOST program supplied with the BASIC system checks the keyboard and automatically loads the LEX binary if appropriate.

Step 2: Loading Binaries

Use the LOAD BIN statement to load the language extensions and drivers required.

1. Get the list of binaries you made while looking through the preceding section. (This will be the binaries that are in the "Needed?" column but not in the "Currently Loaded?" column.) You will be loading these binaries into your computer's memory.
2. Find the disc labeled *Language Extensions.* (If you have double-sided, 3½-inch discs, this will be labeled the *Language Extensions and Drivers* disc.)
3. Remove the *BASIC System Disc* from the "default" disc drive and insert the *Language Extensions* disc.
4. Load the desired language extensions into memory. (You will load the drivers in subsequent steps.) For each language extension in your list, you will need to execute a LOAD BIN statement:

   LOAD BIN "*language_extension*" [Return]

   in which the *language_extension* parameter is the name of the file listed in the preceding "Language Extensions" section. To execute this statement, just substitute the name of the first language extension file into the statement above.

(continued on next page)

Example: Suppose ERR is the name of the first file you want. Then you would type:

```
LOAD BIN "ERR" Return
```

If the file is not on the default volume, you will need to specify which volume (and/or directory) the file is in. For instance:

```
LOAD BIN "/WORKSTATIONS/BIN5.0/ERR" Return
  or
LOAD BIN "/SYSTEMS/ERR:REMOTE" Return
```

5. **Wait for the following message to appear at the bottom of the screen:**

   ```
   ERR 5.0
   ```

   where ERR is the name of the language extension file you just loaded.

6. Repeat steps 4 and 5 to load the remaining language extensions that you have chosen.

7. Determine whether you have loaded all of the desired language extensions by executing this statement (note that you have not yet loaded the drivers):

   ```
   LIST BIN Return
   ```

   If any of the desired language extensions are missing, repeat steps 4 and 5 to load them.

8. Now you are ready to load the drivers for your devices.

   Refer to the list of drivers in the preceding "Drivers Available" section. You will be loading all of the ones with checks in the **Needed?** column.

9. Find the disc labeled *Drivers Disc*, and insert it into the default drive. (If you have *double*-sided, 3½-inch discs, this disc is labeled *Language Extensions and Drivers*; it should still be in your disc drive from the i preceding steps.)

10. Load the desired drivers into memory. For each driver checked as **Needed?** in your list, type:

    ```
    LOAD BIN "driver" [Return]
    ```

    where *driver* is the name of the driver file.

    Example: Suppose DISC is the name of the first driver on your worksheet. Then you would type:

    ```
    LOAD BIN "DISC" [Return]
    ```

11. Wait for the following message to appear at the bottom of the screen:

    ```
    DISC 5.0
    ```

    where DISC is the name of the driver file you just loaded. (Note that if the binary is already loaded, this message will not be shown.)

12. Repeat steps 10 and 11 to load the remaining drivers. If a driver appears more than once on your list, load it only once.

13. Determine whether you have loaded all of the desired drivers by executing this statement:

    ```
    LIST BIN [Return]
    ```

    If any driver binaries are missing, repeat steps 10 and 11 to load them.

Step 3: Storing Your Customized BASIC System

Instead of manually loading binaries each time you use BASIC, you can let the computer load them for you automatically. You can even create several custom configurations designed for different applications; for example, you could create one for general-purpose programming, and another for graphics programming.

1. If the disc onto which you will be storing your custom system is not in the default drive, you will need to determine the volume specifier of the destination disc. One of the following statements should be true:

   - All of your discs should be labeled with the *volume specifier.* (If not, instructions for labeling them are in the "Verifying and Labeling Devices" chapter.)
   - You can look up the *volume specifier* of this drive in the Mass Storage Devices section of the BASIC System Worksheet (which you may have filled out while using the *Peripheral Installation Guide*).

2. Type:

   STORE SYSTEM "SYSTEM_*xxx*:*volume specifier*" [Return]

   where *xxx* is three or fewer letters that uniquely name your custom system, and *volume specifier* identifies the disc on which you want to store your custom system.

   If you have Boot ROM revision 3.0 or A.0, or later revisions, then you can use SYS instead of SYSTEM_; this allows you to choose up to 7 letters in the system file's name.

   STORE SYSTEM "SYS*xxxxxxx*:*volume specifier*" [Return]

   (With files on HFS volumes, the system file name is limited to 9 characters total: "SYSTEM_" and two additional characters, or "SYS" and six additional characters.)

   Example: Suppose you decide to name your system SYSTEM_MY. Assuming that the drive containing your disc is located at ,700,1. Then you would type:

   ```
   STORE SYSTEM "SYSTEM_MY:,700,1" [Return]
   ```

   Examples of Custom System Names:

   - **SYSTEM_IO** for an I/O programming environment (i.e., contains the IO language extension binary).
   - **SYSGRAPH** for a graphics programming environment (only possible on a computer with Boot ROM revisions 3.0, A.0, or later).

Step 4: Booting Your New Custom System

Boot a custom system file just like you would any other system:

- If you have *only one* custom system currently on-line:
  - Turn the computer off, then on again. The system will be booted automatically.

  or

  - Execute the SYSBOOT statement to re-boot your system.
- If you have *more than one* custom system on-line:
  - Turn the computer off, then on again, and press the space bar a few times. Select the custom system you want by typing the letters that identify the system file; for instance, 2 B.

  or

  - Execute a SYSBOOT command, specifying the system file which contains the system to be booted: If your system file is named "SYSTEM_ZZ" and it is on the volume ":,700,0" then you could type:

    ```
    SYSBOOT "SYSTEM_ZZ:,700,0" Return
    ```

    Note that you can also boot other operating systems, such as HP-UX or Pascal, using this technique.

If you have questions on any of these procedures, see the "Loading BASIC" chapter for further explanation.

Note about Early Boot ROMs

If you have boot ROM 1.0 or 2.0, the computer loads the first file it finds whose name begins with the letters SYSTEM_. To load a different file, do either of the following:

- Take all other systems off-line (turn off the drives), or
- RENAME all other system files to eliminate the SYSTEM_ prefix. For example, assume you have three system files named SYSTEM_BA1, SYSTEM_BA2, and SYSTEM_BA3. To load SYSTEM_BA3, rename the other files to BA1 and BA2.

Autostart Files

An “autostart” file is a BASIC program that is automatically loaded and run when the BASIC system is booted[1].

The Autostart File May Load Additional Components

The autostart program shipped with the system sets up a “standard” configuration by loading the most-frequently-used optional components; these components include drivers for most disc types, the BASIC editor, the ERR binary, and other language extensions and drivers that we assume you will need to begin using your system. (The preceding section, called “Determining the Current Configuration,” shows how to list the components loaded by the default autostart program.) You may want to make your own customized autostart program that loads just the binaries you want. The autostart program can also do other things for you, too, such as specifying the default volume, changing operating modes, and loading another program.

An Example Autostart Program

The autostart program shown in this section is an example of a program that loads some commonly used language extensions and drivers. (If you want information about entering BASIC programs, see the “Editing and Storing Programs” chapter.)

```
100  PRINT "Loading Language Extensions.  Please wait."
110  LOAD BIN "GRAPH"
120  LOAD BIN "ERR"
130  LOAD BIN "KBD"
140  PRINT "Remove 'BASIC Language Extensions' disc,";
150  PRINT "and insert 'BASIC Drivers' disc. "
160  INPUT "Now press 'Return' or 'ENTER'.",C$
170  PRINT "Loading 'Driver' files.  Please wait."
180  LOAD BIN "DISC"
190  LOAD BIN "HPIB"
200  LOAD BIN "SERIAL"
210  PRINT "BASIC configuration complete."
220  PRINT "Remove the 'BASIC Drivers' disc."
230  !
240  KEY LABELS ON    ! Now add some mode changes.
250  PRINT "The time is: ";TIME$(TIMEDATE)
260  PRINT "The date is: ";DATE$(TIMEDATE)
270  MASS STORAGE IS ":,700"
280  ! etc., etc.
290  !
300  LOAD "NextProg",1   ! Load and run another program.
310  END
```

[1] A description of this process is given in “The BASIC Booting Process” near the end of this chapter.

Note

Before using STORE "AUTOST" to store the program on the BASIC System Disc, you should have already made a back-up copy of that disc. Otherwise, this AUTOST file will **replace** the AUTOST file shipped with the BASIC system.

The program loads some binaries, and then performs some additional system configuration and mode changes. Just about any programmable operation available to BASIC programs are available in AUTOST programs. (BASIC keywords which require certain language extension binaries may not be available, since the binary may not be present at the time that the AUTOST program is loaded and run. However, they can be used in another program that the AUTOST program loads and runs.)

Note that the CRTA or CRTB driver is not included in the program because it is loaded automatically when BASIC is booted. There is no need to try to load it again.

Also note that you may add or delete LOAD BIN statements from the program depending on the number of binaries you need to load.

Booting BASIC with the Preceding Example AUTOST File

Here are the steps you would take in booting BASIC when the preceding example AUTOST program file is on the same disc.

1. Boot as normal (see the "Loading BASIC" chapter if you need a full explanation).
2. Remove and insert discs as instructed on the screen.
3. Wait for the message:

```
BASIC configuration complete.
Remove the 'BASIC Drivers' disc.
```

A Closer Look at the BASIC Booting Process

The process of loading an operating system into the computer and beginning to execute it is known as "booting". This term is derived from the analogy of the computer "pulling itself up by its own bootstraps." Learning about this will help you to determine which type of customization method, if any, you will want to use in customizing your system configuration.

The Boot ROM Searches for Systems

When you turn on the computer, it begins executing a program in ROM[1]. This program, which for simplicity we'll call "the Boot ROM", searches for all "on-line" operating systems, which have the following characteristics:

- They are files of type "SYSTM" (or HP-UX on an HFS volume).
- Their names begin with the letters "SYS".
- They are on mass storage media (such as disc or tape) that is currently operational and accessible to the Boot ROM.

For BASIC, this is a file named `SYSTEM_BA5`, in which the suffix `5` indicates the major revision of the system—5.x in this case.

Once the "Main" BASIC operating system is loaded, it looks for an "autostart" program — a file of type `PROG` named `AUTOST` on the "default" volume. (The default volume is the volume on which the `SYSTEM_BA5` file was found, or for ROM-based systems, the first mass storage device found with media.)

The Boot ROM Loads an Operating System

The Boot ROM then loads an operating system, which is either:

- the first one it finds (according to the search order described at the end of the "Loading BASIC" chapter), or
- the one that you select. (Boot ROMs with revision 3.0 or A.0, or later, allow you to explicitly choose a system. This topic is also described in the "Loading BASIC" chapter.)

[1] ROM stands for Read-Only Memory, a permanent storage area which is retained even when power is off.

What to Do Next

Task/Topic	Chapter Number
Learn about how to begin using the BASIC system to execute commands, load and run programs, manage directories and files, or edit programs.	8 thru 11
Learn about how to maintain your system.	12 thru 17
Learn about utilities available with the BASIC system.	18
Learn about each key on your keyboard.	19 thru 21

Notes

Using BASIC: Introduction

The “Using BASIC” chapters of this manual describe in detail how to type commands, load and run programs, and how to edit programs.

Prerequisites

You should have already performed the following tasks:

- Installed (and optionally configured) your BASIC system; tasks include booting BASIC, verifying and labeled peripherals, and formatting a flexible or hard disc.
- If you are using an application software pack, your particular program may require special hardware or software configuration. Consult the program’s documentation before reading these chapters.

Introduction to the System 8

This chapter introduces you to using the keyboard and display of HP Series 200/300 computers. Topics include how to type in and execute commands, using typing-aid softkeys, determining how much memory is available, and determining which device is the system printer.

Using the Keyboard

There are three different types of keyboards available with Series 200 and Series 300 computers. Here are the options, along with the names by which we will refer to them throughout the BASIC manual set.

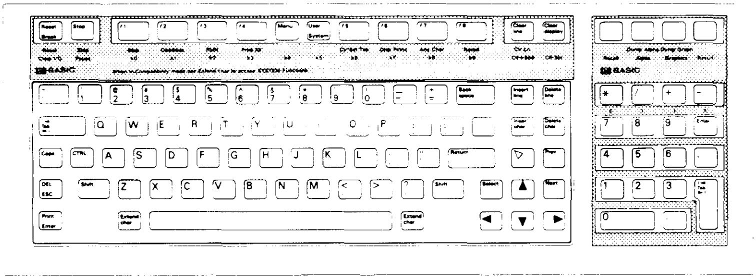


Figure 8-1. ITF Keyboard (with BASIC Keyboard Overlay)

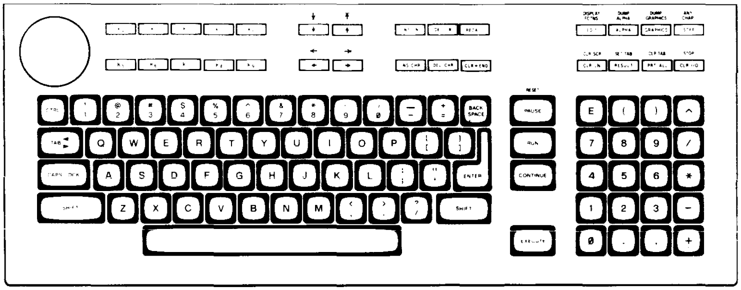


Figure 8-2. HP 98203B/C Keyboards

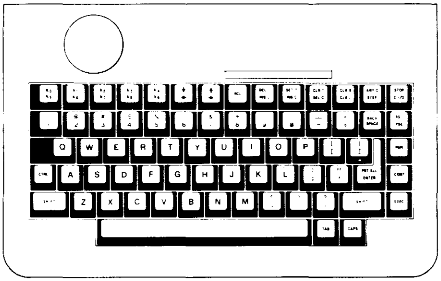


Figure 8-3. HP 98203A Keyboard

What Can You Do at the Keyboard?

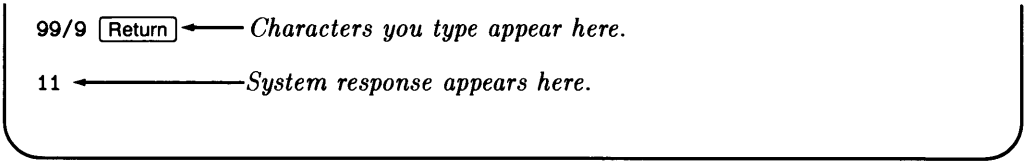
There are several things you can do at the keyboard of this system:

- Perform calculations.
- Type in and execute commands.
- Load and run programs, and control program execution.
- Type in, edit, and store programs.

Let's now look at how to perform calculations, and type in and execute commands. (The last two of these operations are discussed in subsequent chapters.)

Performing Calculations at the Keyboard

This BASIC system has an command interpreter that can also evaluate numeric expressions. For example, you could type:



99/9 Return ← *Characters you type appear here.*

11 ← *System response appears here.*

Note that with HP 98203 keyboards, you will press the ENTER key instead of the Return key.

Typing and Executing Commands

You can almost always type in and execute commands from the keyboard with this BASIC system. Just about the only times that you cannot are when:

- There is currently a command being executed, with another one already entered and waiting to be executed.
- When there is a program running that traps keystrokes (with ON KBD) or disables the keyboard (with SUSPEND INTERACTIVE).

At all other times, you can type in commands and press Return or ENTER to present them to the system for execution. The system parses the command, and takes the appropriate action; for instance, many commands request that the system respond to you with some sort of information.

Example Command (Determining Available Memory)

For instance, type in and execute the following command (characters will appear in the "Keyboard Input" line, near the bottom of the display):

```
SYSTEM$("AVAILABLE MEMORY")  [Return] or [ENTER]
```

The system returns something like this (in the "System Messages" line, at the bottom of the display):

```
123456
```

Example Command (Checking and Setting the System Clock)

The following functions allow you to check the setting of the system will clock:

```
DATE$(TIMEDATE), TIME$(TIMEDATE)  [Return] or [ENTER]
```

The system returns something like this:

```
 17 Mar 1987      10:27:32
```

You can set the time and date with the SET TIMEDATE statement; here is an example:

```
SET TIMEDATE DATE("17 Mar 1987")+TIME("10:30:00")  [Return] or [ENTER]
```

Note that if you are sharing a hard disc with an HP-UX system, you should also use the TIMEZONE IS statement for compatibility with time stamps on files. See the *BASIC Language Reference* for details.

Keyboard Input Line Length

As you type in a command, note that it appears on the "keyboard input" line (see the preceding drawing). This "line" is actually two display lines in size, allowing you up to 100 characters (Model 226 only), 160 characters (most models), or 256 characters (models with hi-resolution displays).

Notation Used for Text You Should Type

The notation used in the BASIC manuals is that the characters that you can type in appear in:

```
DOT-MATRIX FONT
```

Whenever you see characters in `this font`, you should remember that you can type these characters exactly as shown in a command (or program line, depending on the example).

This `DOT-MATRIX` font is also used to show the system's response to your commands, etc. The difference between the two is determined by the context.

Notation Used for Keys

The notation used for keys is to have the key's label enclosed in a rounded box, like this: Reset. This indicates that you are supposed to press the key with that label.

When two keys are joined by a hyphen (Shift-Break), it indicates that you should first press the key on the left (Shift in this example), and then press the key on the right (Break) while still holding down the other key.

Caps Mode Indicator (ITF Keyboard Only)

At the lower, right-hand corner of the display, there is an area used specifically to show when you are in Caps Mode. This indicator has two states:

`Caps` indicates that an uppercase letter will be produced when you press an alphabetic key—A through Z. (Use Shift with one of these keys to produce a lowercase letter.)

(blank) indicates that a lowercase characters will be produced when you press an alphabetic key. (Use Shift with an alphabetic key to produce an uppercase character.)

Pressing the Caps key toggles between these two modes.

Note that the `Caps` indicator remains on while in the Caps Mode.

Caps Lock Indicator (98203 Keyboards Only)

When you press the CAPS LOCK key on an HP 98203 keyboard, the BASIC system displays a mode indicator in the System Message/Results line:

`Caps Lock On` indicates that an uppercase letter will be produced when you press an alphabetic key—A through Z. (Use SHIFT with one of these keys to produce a lowercase letter.)

`Caps Lock Off` indicates that a lowercase characters will be produced when you press an alphabetic key. (Use SHIFT with an alphabetic key to produce an uppercase character.)

Each time you press the CAPS LOCK key, you will toggle between these two modes.

Note that the `Caps Lock` indicator is erased the next time that the System Message/Results line is updated.

The Significance of Lettercase

While typing commands into this system, lettercase is *usually* not important. For instance, these two commands are recognized as being equivalent by the system as the BEEP statement:

```
beep
BEEP
```

However, lettercase *is* significant when typing in things like literal text (enclosed in "quotes"). Also, BASIC keywords can be all uppercase or all lowercase; however, when lettercase is mixed in a keyword, the system interprets it as a variable name. For instance, `Beep` would be interpreted as a variable name, not as a keyword. Here are some examples that are **not** equivalent:

Example Commands that Are NOT Equivalent		Reason
`LOAD "MyFile"`	`LOAD "MYFILE"`	Literal string
`system$("MSI")`	`system$("msi")`	Literal string
`BEEP`	`Beep`	Can't mix lettercase in keywords

Using Typing-Aid Softkeys (Required KBD Binary)

There are "softkeys" on all three of the keyboards available on Series 200/300 computers. Here are the locations of these keys:

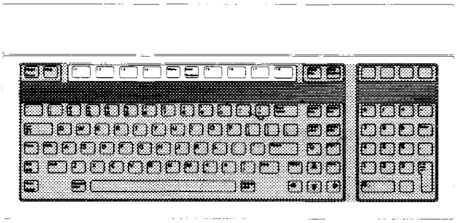


Figure 8-4. Location of Softkeys on ITF Keyboard

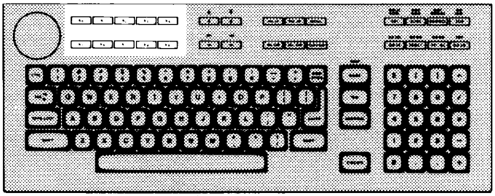


Figure 8-5. Location of Softkeys on 98203B/C Keyboard

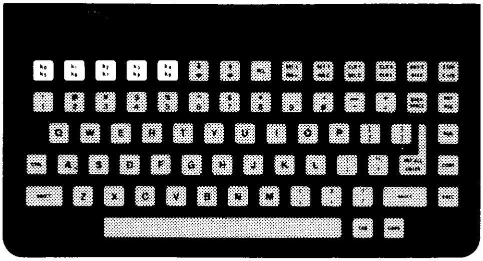


Figure 8-6. Location of Softkeys on 98203A Keyboard

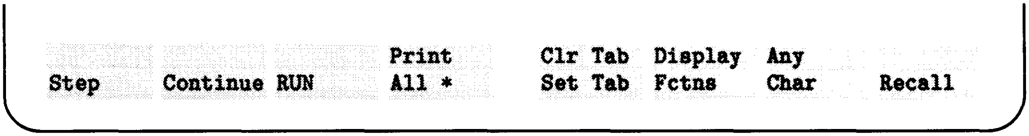
Softkey Labels

Before using any of the softkeys, it is usually helpful to be sure that the one you have chosen to press is the one you really want. To see the current definitions of the softkeys, execute the following statement:

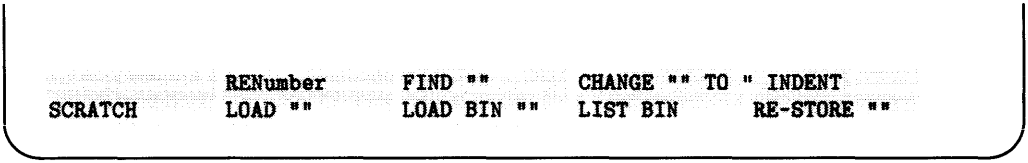
KEY LABELS ON Return or ENTER *(Requires CRTX binary)*
  or
CONTROL CRT,12;2 Return or ENTER

(On an ITF keyboard, you can also press the Menu key to turn on the softkey labels.)

This command turns on the "Key Labels" area at the bottom of the display, as shown below (this example is for a system with an ITF keyboard):



The following diagram shows the format of the softkey labels when an HP 98203 keyboard is in use:



Example Typing-Aid Softkey Definition

Typing-aid softkeys produce a sequence of characters[1] on the keyboard input line (or on the "current line" in EDIT mode). For instance, pressing f1 (on an ITF keyboard) produces these characters:

```
EDIT
```

You can also re-define these keys, so that each produces a special set of characters that are specific to your needs. See the subsequent section called "Re-Defining the Typing-Aid Softkeys" for details.

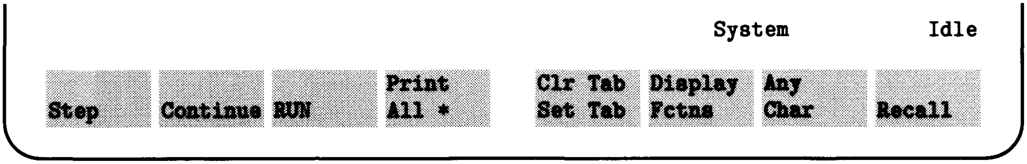
[1] Typing-aid softkeys are only active when a running program has not defined them to produce an interrupt. See the "Program Structure and Flow" chapter discussion of ON KEY for details.

Softkey-Menu Indicator (ITF Keyboard Only)

On computers with ITF keyboards, the softkeys (f1 through f8) have **four independent sets of definitions.** You can select which set of definitions (and labels) to activate by using the System, User (Shift-System), and Shift-Menu keys. You can turn the labels on and off with the Menu key.

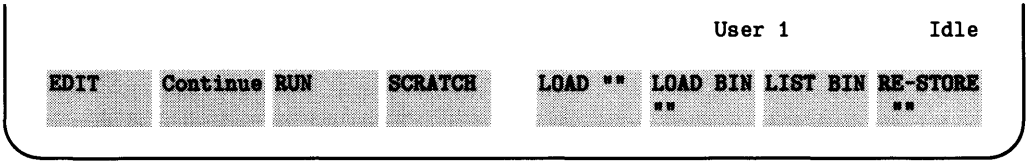
With each set of definitions, there is a menu that lists the keys' current definitions.

The System softkey menu shows definitions for "immediate-execute" system operations such as Step, RUN, and Recall. When this menu is active, the keys f1 through f8 have the definitions shown below:



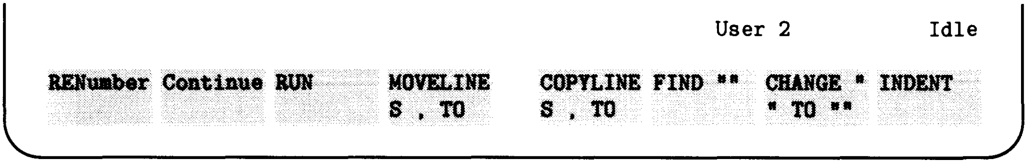
The System menu is the default menu at system power-up, SCRATCH A, etc. (when the KBD binary is not present).

The User 1 menu shows typing-aid definitions for the User 1 softkeys. These keys will produce specific sequences of characters when pressed. For instance, pressing f1 will produce the characters `EDIT` while in this menu.



The User 1 menu is the default menu when the KBD binary is loaded.

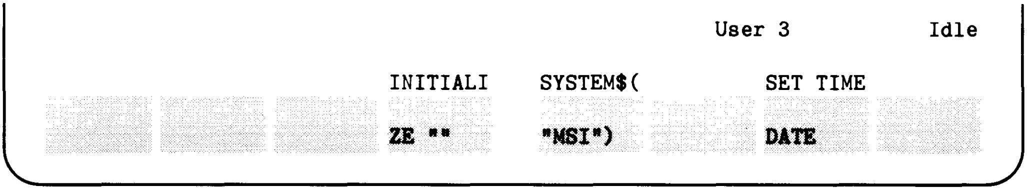
The User 2 softkey menu has the following default definitions:



```
                                               User 2          Idle

RENumber Continue RUN      MOVELINE    COPYLINE FIND ""  CHANGE " INDENT
                           S , TO      S , TO            " TO ""
```

Note that softkeys [f3] through [f8] are only defined if the PDEV and EDIT binaries are loaded at the time that the default definitions of these keys are set up (such when SCRATCH A, LOAD KEY, or LOAD BIN "KBD" are executed).

The User 3 softkey menu has the following default definitions:



```
                                               User 3          Idle

                           INITIALI    SYSTEM$(          SET TIME

                           ZE ""       "MSI")            DATE
```

Using the System keys is described in the relevant section of this manual; for instance, using the [RUN] key is described in the chapter called "Loading and Running Programs". For further information about using and re-defining the typing-aid keys, see the section of that name later in this chapter.

What State Is the System In?

When power is first switched on and the BASIC system is booted, memory is cleared and various system elements are assigned default values. For example, the CRT display is assigned as the system printer. This condition is called the "power-on state" of the computer, which is very predictable[1].

If your computer has been used since power-on, it may be in a somewhat unknown state. For instance, there may be an unwanted program in memory, or the default printer or volume may have been changed to specify devices that you don't want. This section explains:

- How to find out what your computer is doing, and what state it is in.
- How to get it into a state in which you can begin to use it properly.

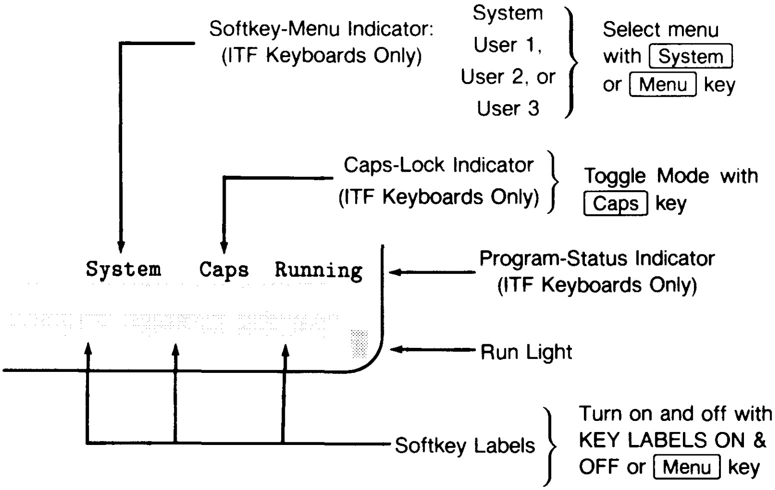
Is a Program Running?

If there are several people sharing a computer, you should generally find out whether the computer is currently in use so that you will not destroy someone else's work. It is also convenient to be able to quickly glance at the display to find out whether your own program is running, waiting for input or for a device to become available, and so forth.

[1] For a complete list of power-on defaults, see the "Useful Tables" appendix of the *BASIC Language Reference.*

Program-Status Indicators

You can determine the current status of the system by looking at the lower right-hand corner of the CRT. The BASIC system uses this area to display information about whether a program is currently running, what softkey menu is currently active, and so forth.


Softkey-Menu Indicator: (ITF Keyboards Only)
System
User 1,
User 2, or
User 3
Select menu with System or Menu key
Caps-Lock Indicator (ITF Keyboards Only)
Toggle Mode with Caps key
System
Caps
Running
Program-Status Indicator (ITF Keyboards Only)
Run Light
Softkey Labels
Turn on and off with KEY LABELS ON & OFF or Menu key

The character in this corner is referred to as the "run light". The following table shows the various indications of the run light and their meaning.

Status Indicator[1]	Run Light	System State
Idle	*(blank)*	Program stopped; can execute commands; CONTINUE not allowed.
Running	▒	Program running; can execute commands.
Paused	_	Program paused; can execute commands; CONTINUE *is* allowed.
Transfer	IO	Program paused, but an overlapped TRANSFER (I/O) operation is still in progress; can also execute commands.
Input?	?	BASIC program waiting for input from keyboard; cannot execute commands.
Command	*	System executing command entered from keyboard; can enter 1 more command, but it will not be executed until after the current command is completed.

[1] These indicators are displayed only if softkey labels are currently on. Use the KEY LABELS ON statement (or the Menu key on an ITF keyboard) to turn these labels on.

Pausing and Stopping Programs

If the computer operator does not intervene, a program will run until it reaches an END, STOP, or PAUSE statement, or until it pauses to input data or report an error. If you wish to pause or stop a program before its normal completion, use the following keys:

Large Keyboard	ITF Keyboard	Small Keyboard	Effect
PAUSE	Stop	PSE	Pauses program execution after the computer finishes the line it is on and any I/O operations in progress. This is useful for pausing a program that is executing an INPUT statement. It leaves all necessary internal information intact, so that program execution can be resumed again with the CONTINUE key (see below) or CONT command.
CONTINUE	f2 (System menu)	CONT	Pressing CONTINUE after a PAUSE (key or statement) causes program execution to resume in a normal manner from the place where it was paused.

Large Keyboard	ITF Keyboard	Small Keyboard	Effect
CLR I/O	Break	C I/O	Aborts any I/O statement in progress (such as ENTER or TRANSFER) and pauses the program. The program counter is returned to the beginning of the aborted I/O statement, so that the CONTINUE key or the CONT command cause the program to resume program execution beginning with that same statement. This is useful when the computer is "hung" trying to output to a device that is down.
STOP (SHIFT-CLR I/O)	Stop (Shift-Stop)	STOP (SHIFT-C I/O)	Stops the program after the computer finishes executing the line it is on (and returns the program to the main context, if executing a subprogram or user-defined function). However, it does not affect the interfaces, the CRT, program memory, the values of variables, tabs, or the RECALL key's buffer. CONTINUE is not allowed after a STOP.
RESET (SHIFT-PAUSE)	Reset (Shift-Break)	RST (SHIFT-PSE)	This is the most drastic and complete way to halt program execution. The program stops immediately, all I/O operations are aborted, any open files are closed, and all interface cards are reset. However, the printout area of the CRT, program or variable memory, tabs, and the RECALL key's buffer are not affected. CONTINUE is not allowed after a RESET.

The easiest way to remember the key definitions on the ITF keyboard is to use the keyboard overlay and keep the softkey labels on. (To turn softkey labels on, either press the Menu key or execute the KEY LABELS ON statement.)

Is There a BASIC Program in Memory?

If a BASIC program is currently in memory, then use this command to list the lines on the current *system printer* (the default is the display screen[1]):

```
LIST [Return] or [ENTER]
```

If there is a program in memory, you will see something like this:

```
10  PRINT "Short program."
20  PRINT "Goodbye."
30  END
```

After the listing is complete[2], the system displays the amount of memory currently available for BASIC programs and variables at the lower left corner of the screen:

```
Available memory = 123456
```

(If there is no program currently in the computer, then the amount of available memory is all that will be displayed by LIST.)

Clearing the Program

You can easily clear the computer's memory with the SCRATCH command (see the section called "Clearing the Computer" on page 8-24). However, before you do so, you might want to determine what the current defaults are so that you can reset them if they are modified when you use SCRATCH to clear the current program; the next section provides instructions for determining these defaults.

[1] The subsequent section called "Determining Current System Defaults" shows how to determine which device is the current system printer.

[2] If listing the program takes longer than you want to wait, you can stop it by pressing the [Break] key ([Clrl/O] on large keyboards; [Cl/O] on small keyboards).

Determining Current System Devices and Binaries

You can determine the current state of several of these system defaults by using the following statements:

Method	Explanation	Default
`SYSTEM$("PRINTER IS")`	returns the select code of the current "system printer" (the destination of PRINT operations).	`PRINTER IS CRT`
`SYSTEM$("PRINTALL IS")`	returns the select code of the current "printall printer" (the destination of system messages when PRINTALL ON is active).	`PRINTALL IS CRT`
`SYSTEM$("DUMP DEVICE IS")`	returns the select code of the current "system dump device" (the destination of DUMP ALPHA and DUMP GRAPHICS operations).	`DUMP DEVICE IS 701`
`SYSTEM$("MSI")`	displays the current "default drive" (the mass storage device that will be used when one is not explicitly specified).	device from which you booted the BASIC system[1]
`SYSTEM$("AVAILABLE MEMORY")`	returns the amount of memory available for application programs and their data.	*(not applicable)*

[1] If you have ROM-based BASIC, this device will be the first mass storage device found with storage media present. See the "Order of Devices Searched by the Boot ROM" section of the "Loading BASIC" chapter for a list of the devices searched by the Boot ROM.

Method	Explanation	Default
`SYSTEM$("VERSION:BASIC")`	returns the revision number of the BASIC system.	*(not applicable)*
`SYSTEM$("VERSION:ERR")`	returns the revision number of the ERR binary (if present), or returns 0 (if the binary is not present). To check for other binaries, substitute the binary's name for **ERR** in the **SYSTEM$** function call; for instance, `SYSTEM$("VERSION:EDIT")` would return the version number of the EDIT binary, if currently loaded, or 0 if it is not loaded.	*(not applicable)*
`LIST BIN`	displays a list of **all** language extension and driver binaries currently residing in memory. (For further information about language extension and driver binaries, see the "Language Extensions, Drivers, and Configuration" chapter.)	`CRTA` and `CRTB`

See the the *BASIC Language Reference* for further capabilities of SYSTEM$; the "Customizing Your BASIC System" describes LIST BIN and the binaries available with the system.

Re-Defining Typing-Aid Softkeys

The default typing-aid softkey definitions are useful, but you may want to define a key that is more specific to your own needs. This section describes how to change the current definitions of typing-aid keys; it also shows how to store these definitions in a file so that you can programmatically load them at a later time.

KBD Binary Is Required

In order to re-define typing-aid keys, make sure that the KBD binary is currently loaded (see the preceding section for examples of LIST BIN and SYSTEM$ statements).

Examples of Re-Defining Typing-Aid Softkeys

The first example shows how to re-define softkey f1 (k1) to produce the characters "My very own keystrokes" on the display whenever you subsequently press this key.

1. Enter the edit-softkey mode for the desired softkey. There are two ways to enter this mode:

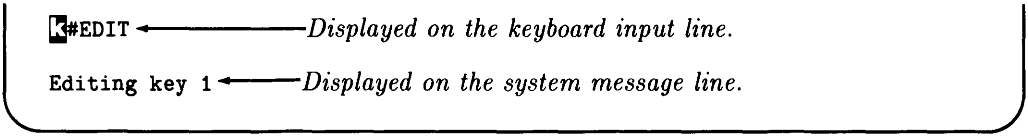
   - Use the EDIT KEY statement, specifying the number of the softkey to be defined; for instance, to edit the current definition of typing-aid softkey 1, type:

     `EDIT KEY 1` Return or ENTER

   - Press the EDIT key (or softkey), and then press the softkey to be re-defined. For instance:

     - On an ITF keyboard, get into the `User 1` softkey menu and press the EDIT key (f1), and then press f1.
     - On a 98203 keyboard, press the EDIT key, and then press k1.

   The system then displays the key's current definition (on the keyboard input line), followed by a message to indicate that you can now modify the definition of the softkey:



```
K#EDIT
Editing key 1
```

Displayed on the keyboard input line.

Displayed on the system message line.

(The keyboard input line is blank if the key currently has no definition.)

(The keyboard input line is blank if the key currently has no definition.)

2. Press CLR LN (Clear line) to clear the key's current definition.
3. Enter the new definition.

   Type the desired characters on the keyboard input line. In this example, type:

```
My very own keystrokes.
```

4. Exit the softkey-edit mode.

   After you have finished modifying the softkey's definition, press Return or ENTER to exit the softkey editing mode and update the softkey's definition. If you don't want to update this softkey's definition, press PAUSE (Break on an ITF keyboard) to abort the re-definition and retain the existing definition.
5. Verify that the key works as desired.

   Press the softkey f1 (k1). The following characters:

```
My very own keystrokes.
```

should be reproduced exactly as you typed them. (If not, go to step 1 and try again.)

Another Example

Let's now suppose that you want to define a typing-aid key that: clears the line you are on, types a command, and then executes that command. Here is what you could type:

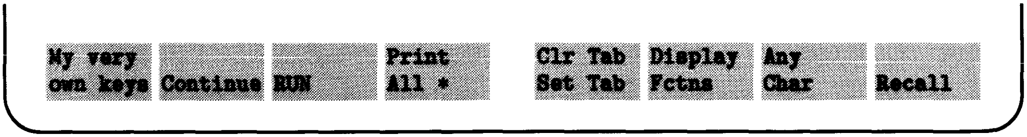
`EDIT KEY 2` Return

Clear line CTRL-Clear line L I S T CTRL-Return Return

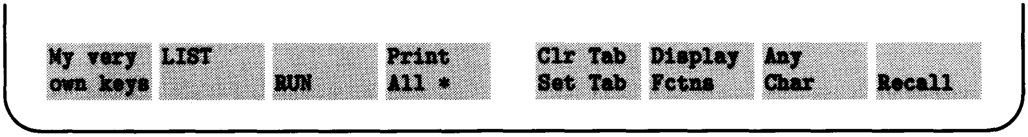
The notation CTRL-Return indicates that you are to hold down the CTRL key and then press the Return key, then release the Return key followed by releasing the CTRL key. The CTRL key tells the system that you don't want to execute that key's function, but that you want it to produce a sequence of characters on the screen. The sequence begins with the inverse-video "k" character, CHR$(255), followed by another character; for instance, CTRL-Clear line produces "k#", and CTRL-Return produces "kE".

Softkey Labels

You may have noticed the two different softkey labels produced in the preceding examples. The label produced in the first example should have begun with the characters **My very own keys** (the rest of the characters in the softkey's definition are not shown in the label).



However, the label of the second example was simply **LIST**.

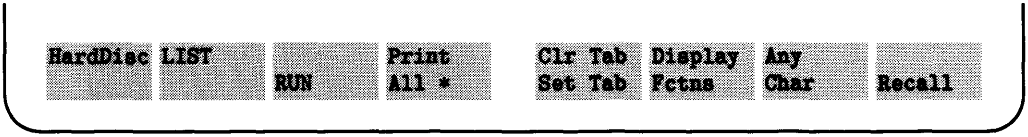


This second result happens because the BASIC system automatically assumes that you don't want the Clear line keystroke characters to be shown in the label, and so suppresses them; the label begins with the 3rd character of the sequence (the **L** of **LIST** in this example).

You can also use a similar effect of the Clear line key in softkey labels to have nice-looking labels that are not just the first *n* characters in the key's definition (in which *n* is the width of the softkey label). For instance, suppose that you want to have a key that produces the characters :CS80,700,0,1, but you would rather the label be **HardDisc**. Here is what you would enter in the softkey's definition:

```
HardDisc        CTRL-Clear line  :CS80,700,0,1
```

The label **HardDisc** would be displayed momentarily whenever the key is pressed, but the Clear line keystroke would soon erase the characters, leaving only the subsequent :CS80,700,0,1 characters. (The spaces following **HardDisc** move the "k#" characters out of the label.)



Memory Available for Typing-Aid Definitions

There is approximately 1024 bytes of memory used by the system for the purpose of storing the typing-aid softkey definitions. Since there is a small amount of overhead required for each key, there is actually only about 1000 bytes available for keystrokes.

The maximum number of characters you can put into each definition is 256 characters. However, if you produce a 256-character typing-aid softkey definition—on a system with a high-resolution display—and then try to *edit* it on a system with a medium-resolution display, you will only be able to get 160 characters into the modified typing-aid definition. There is also an additional restriction: attempting to *use* a softkey definition that contains more characters than will fit into the Keyboard Input Line will result in lost characters.

Listing the Current Typing-Aid Definitions

You can list all current typing-aid softkey definitions by executing this statement:

```
LIST KEY
```

The keys are listed on the current default printer (usually the CRT).

You can also list them on another device by specifying that device's select code. For instance, to list them on a printer, execute a command something line one of the following:

```
LIST KEY #PRT
  or
LIST KEY #701
```

Since most printers cannot print the inverse-video "K" that would show up in the keys' definitions, LIST KEY substitutes the letters `System key:` for this character when the keys are displayed. For instance:

```
Key 2:
System key: #←————————'Clear line' key.
LIST
System key: E←————————'Return' key.
```

Typing-Aid Softkey Files

If you have modified any of the softkeys, you will probably want to store the new definitions somewhere. With the STORE KEY statement, you can store all of the current typing-aid softkey definitions in a file. Use LOAD KEY to subsequently load these definitions back into the computer.

Storing the Current Typing-Aid Softkey Definitions in a File

If you want your current typing-aid softkey definitions to be stored in a file called "MyKeys", you could use this statement:

```
STORE KEY "MyKeys"
```

If the file already exists, you must use this statement:

```
RE-STORE KEY "MyKeys"
```

Loading Typing-Aid Softkey Definitions from a File

To load the typing-aid softkey definitions stored in the preceding example, execute this statement:

```
LOAD KEY "MyKeys"
```

Restoring Power-Up Typing-Aid Definitions

To restore the power-up default definitions, execute the statement without specifying a file:

```
LOAD KEY
```

Defining Typing-Aid Softkeys Programmatically

You can also write a program that defines the typing-aid softkeys using the SET KEY statement. See the "Communicating with the Operator" chapter of *BASIC Programming Techniques* for details.

Clearing the Computer

As discussed in the preceding section, your computer may or may not be in a desirable state.

- If it is in an undesirable state, then you can "clear" it as described in this section.
- If everything is acceptable, then you can skip this section and go to "Loading and Running Programs".

When power is first switched on and the BASIC system is booted, computer memory is cleared and various system elements are assigned default values. Turning power off and then back on again is one way to clear the computer. However, this is not necessary and often is not convenient. An easier and faster way is to use SCRATCH commands.

There are several forms of this command to allow a choice of clearing actions. The following paragraphs give general descriptions of the choices; complete descriptions are in the *BASIC Language Reference* under SCRATCH in the keyword dictionary and in the "Reset Tables" of the "Useful Tables" appendix.

`SCRATCH R` clears the RECALL key's buffer.

`SCRATCH KEY` clears typing-aid softkey definition(s). See the descriptions of typing-aid keys in preceding sections of this chapter for further information.

`SCRATCH C` clears **all** variables from the computer's memory, including COM variables. However, the current program and softkey definitions are left intact.

`SCRATCH` clears all program lines currently in the computer's memory. It also clears all variables which are not in COM. See the "Subprograms" chapter of *BASIC Programming Techniques* for a description of COM.

`SCRATCH A` clears almost everything from the computer's memory, restoring the system to its power-on state. Just about the only exceptions are the RECALL key's buffer and the real-time clock.

`SCRATCH BIN` removes all of the current binaries in memory (except the driver of the currently active CRT display), and re-executes all of the steps in a power-up operation (except loading and running the "autostart" program). See the "Language Extensions, Drivers, and Configuration" chapter for details.

What to Do Next

Task/Topic	Chapter Number
Learn about how to use and manage files and directories.	9
Learn how to load and run programs.	10
Learn about how to enter, edit, secure, document, and store programs.	11
Learn about how to maintain your system.	12

Notes

Loading and Running Programs 9

In this chapter, you will learn how to load and run programs written in BASIC.

A Brief Look at Loading and Running Programs

This section gives you a brief overview of loading and running BASIC programs. The next section provides a closer look at these operations.

To load and then run a program, follow this simple procedure:

1. Insert the disc containing the program into one of your disc drives. The best choice is the "default drive", since you will not have to include the drive's *volume specifier* when you use this device.

2. Execute the CAT statement to make sure that your program is on the disc.

   `CAT` Return or ENTER

   ```
   :CS80,700
   VOLUME LABEL: B9836
   FILE NAME PRO TYPE  REC/FILE BYTE/REC   ADDRESS

   MY_PROG       PROG        14      256        16
   VISI_TOOL     ASCII       29      256        30
   GRAPH         BIN        171      256        59
   GRAPHX        BIN        108      256       230
   ```

   - If you see the name of your file in the catalog listing, then you are ready to load and run it.
   - If you don't see your file in the catalog listing, then remove the disc and insert the one that contains your program. Use CAT again to verify that the program is on the disc.

3. Use the LOAD or GET command to load the program into computer memory:

   - Use LOAD if your program is in a file of type PROG (like `MY_PROG` above).
   - Use GET if your program is in a file of type ASCII (like `VISI_TOOL` above).

   The CAT listing from the preceding step listed the file type.

Here is an example of using LOAD; substitute the name of your program where *file_name* appears below:

```
LOAD file_name [Return] or [ENTER]
```

For instance, the following command loads the file named MY_PROG from the default mass storage device:

```
LOAD "MY_PROG"  [Return] or [ENTER]
```

This command gets the program (in an ASCII file) named VISI_TOOL from the default mass storage device:

```
GET "VISI_TOOL" [Return] or [ENTER]
```

This last example shows loading the program named MY_PROG from the device with :,700,1 as its volume specifier:

```
LOAD "MY_PROG:,700,1" [Return] or [ENTER]
```

(Each of your drives should have its *volume specifier* labeled on a sticker affixed to the front panel. If not, talk to your system administrator, or see the "Verifying and Labeling Peripherals" chapter for instructions.)

Note about File Protection

Some files are "protected" against being read by BASIC. If you get an error such as 62 (invalid LIF protect code), or 462 or 484 (invalid SRM password), then you will need to do one of the following things:

- Specify the appropriate protect code or password with the file name; the person who protected the file should know it. (This is not possible with files or directories on HFS volumes.)

- Remove the protection from the file. (You can change and remove file and directory protections using PROTECT or PERMIT, as described in the "Using Files and Directories" chapter of this manual.)

4. To run the program you just loaded, type:

   `RUN` [Return] or [ENTER]

   On 98203 keyboards, you can also use the [RUN] key. On ITF keyboards, [f3] in the System menu, and User 1 and 2 menus, serves the same purpose. (If key labels are not currently displayed, then execute KEY LABELS ON or press the [Menu] key to turn them on.)

Note about Software Security

Some software is "secured" against being run without proper authorization, which is usually accomplished by the software requiring a special "codeword" that is somehow related to:

- your machine's serial number (stored in permanent memory)
- a serial number stored in an optional HP 46084 ID Module

If the program prompts you to enter a codeword, you will need to get it from the software vendor.

A Closer Look at Loading Programs

As mentioned in the preceding section, there are two statements used in BASIC for retrieving programs from mass storage:

- LOAD—retrieves programs stored in a PROG file (using STORE).
- GET—retrieves programs stored in an ASCII file (using SAVE).

These statements can be executed from the keyboard as commands or included in a program. When executed as commands, they are used to bring a program into the computer's memory so that it can be edited or run. When included in a program, they are used to link together the segments of large programs.

Using LOAD

The LOAD command is used to bring in programs from a PROG file, with the option of beginning program execution at a specified line. To clear any existing program from the computer's memory and load the contents of a PROG file, the command is simply the keyword LOAD followed by the file name. For example, the command:

```
LOAD "CANNON"
```

clears the program memory and brings in the contents of the PROG file called "CANNON". If the file is not a PROG file, the LOAD is not performed and error 58 (improper file type) is reported. If the file is not found on the volume, error 56 (file not found) is reported. If any lines require a version of BASIC different from the one currently installed, those lines cannot be listed, edited, or executed. However, the LOAD operation proceeds without error.

The LOAD command can also specify that program execution is to begin. This is done by adding a line identifier. For example, the command:

```
LOAD "STONE",10
```

causes the computer to load the program in file "STONE" and begin execution at line 10. The line identifier may be a label or a line number, but it must identify a line in the main program segment (not in a subprogram or user-defined function). See the "Program Flow" chapter of *BASIC Programming Techniques* for further information.

The LOAD command cannot be used to bring in arbitrary program segments or append to a main program like GET can. Subprogram segments can be appended using LOADSUB, as described in the "Subprograms" chapter of *BASIC Programming Techniques.*

Using GET

The GET command is used to bring in programs or program segments from an ASCII file, with the options of appending them to an existing program and/or beginning program execution at a specified line.

GET with Automatic Program Clearing

To clear any existing program from the computer's memory and bring in the contents of an ASCII file, the command is simply the keyword GET followed by the file name. For example, this command:

```
GET "FORMULA"
```

clears any BASIC program currently in memory, and then brings in the contents of the ASCII file called "FORMULA"—assuming that the file contains valid program lines.

If the first line does not start with a valid line number, the GET is not performed and error 68 (syntax error during GET) is reported. If the file is not an ASCII file, the GET is not performed and error 58 (improper file type) is reported. If the file is not found on the volume, error 56 (file not found) is reported.

Assuming that the file contains valid program lines that were placed in the file by a SAVE operation, and their line numbers are still valid after any renumbering that is specified, the lines will be entered into program memory. If there is a syntax error in any of the program lines in the file, the lines in error are turned into comments, error 68 is reported, and the syntax error message is sent to the system printer. This might happen if the program was written and saved on a computer that had a version of BASIC different from the one being used for the GET operation. This may also happen when a "language extension" binary is required for using the keyword, but the binary is not currently loaded into memory.

Using GET to Specify Run Line

GET can also specify that program execution is to begin. This is done by adding two line identifiers:

- The first line specifies the placement and renumbering described in the preceding section.
- The second line specifies the line at which execution is to begin.

For example, assume that there is no program in memory and that an ASCII file "RATES" contains valid program lines. A typical command to bring the contents of this file into memory and begin execution at the line 10 is:

```
GET "RATES",10,10
```

If there is already a program in memory, an append and run is allowed. For example:

```
GET "RATES",250,100
```

specifies that any existing lines from 250 to the end are to be deleted, the contents of file "RATES" is to be renumbered and appended beginning at line 250, and then normal program execution is to begin at line 100. Although any combination of line identifiers is allowed, the line specified as the start of execution must be in the main program segment (not in a SUB or user-defined function). Execution will not begin if there was an error during the GET operation. For further information about this use of GET, see the "Chaining Programs" section of the "Program Flow" chapter in *BASIC Programming Techniques.*

Other Uses of GET

You can also use GET to append the contents of an ASCII file to an existing program. See the "Editing and Storing Programs" chapter or the *BASIC Language Reference* for details.

A Closer Look at Running Programs

There are two ways to start a program running:

- Press the RUN key (f3 in the System and some User menus of ITF keyboards).
- Execute a RUN command.

These keys or command tell the system to go through an automatic *prerun* phase and then begin normal program execution with the lowest-numbered line in the main program.

You can also include a line identifier in a RUN command to indicate where program execution is to begin.

```
RUN 200
```

or

```
RUN Line_id
```

Prerun

Prerun is automatically performed by BASIC when you execute RUN or press the RUN key. There are several reasons for the prerun.

- **To reserve sufficient memory** for all the variables in the program (except those that are ALLOCATEd). This includes all variables in COM statements, those declared in DIM, REAL, and INTEGER statements, and all implicitly declared variables. The "Numeric Computation" chapter of *BASIC Programming Techniques* explains the declaration of numeric variables; the "String Manipulations" chapter covers the dimensioning of string variables; and the "Subprograms" chapter describes COM.
- **To locate all the context boundaries.** These are defined by the END, SUB, SUBEND, DEF FN, and FNEND statements.
- **To ensure correct interaction between lines.** The "Editing and Storing Programs" chapter explains that BASIC checks for syntax errors before it stores a program line. Although this is true, there are some errors that can't be detected by looking at a single line. For example, a program line that uses properly placed subscripts can appear to be correct when it is stored. However, if that line references three dimensions in an array that had previously been declared to have only two dimensions, it is in error. To detect an error of that kind, the computer needs to "look at" the entire program to see all the dimension statements as well as the variables used in each line. Some other examples of this kind of error are specifying a GOTO or GOSUB to a line that does not exist or improper matching of statements like FOR...NEXT and IF...END IF. At prerun, BASIC also "links" line identifiers (and line numbers) to all references to them, so that program execution will be faster.

Normal Program Execution

The term "program execution" is used to describe the process of the BASIC system completing the tasks "described" by a program. The steps that the system performs during program execution are summarized below.

1. Determine which program line is to be acted on next.
2. Identify the statement that follows the line number and label (if any) on that line.
3. If the statement has a run-time action, perform the action described in the statement.
4. Repeat steps 1 thru 4 until instructed to pause or stop (such as by an END, STOP, or PAUSE statement, or by a RESET from the keyboard).

The continuing process of determining which line is to be executed next is discussed in detail in the chapter called "Program Flow" in *BASIC Programming Techniques.* The RUN command determines which line is acted on first. Executing **RUN** with no parameters, or pressing the RUN key, causes the execution process to begin at the first (lowest-numbered) line of the main program. Execution can be started anywhere in the main program by using the RUN command with a line identifier. For example, the following command causes execution to begin at line 220, if there is such a line.

```
RUN 220
```

If there is no line 220 in the main program, execution begins with the line whose number is closest to and greater than 220.

The line identifier can also be a label. For example, the following command causes execution to begin with the line labeled "Spot_run".

```
RUN Spot_run
```

If there is no such label, an error results.

Note that the prerun phase is always the same, whether the actual execution begins at the program start or somewhere in the middle. Also, if a starting line is specified, that line must be in the main program. An error 3 (line not found in current context) results if you attempt to start a program in a user-defined function or subprogram. Even if the starting point is correctly specified, be alert to the effects of starting a program in the middle. Skipping over a section of the program may result in null values for some of the variables. Although it is legal to start in the middle of a subroutine, an error is generated when the RETURN statement is executed.

Non-Executed Statements

In the preceding summary of normal execution, step 3 mentioned that only statements with run-time actions are executed. The term *run-time* refers to the state that exists after the prerun, when the computer is actually performing the actions described in the program. Some statements are not executed in the course of normal program flow, but are merely "looked at" and then bypassed. The following is a list of some statements that do not cause an action as a result of run-time execution.

- Comments and REM statements. These never cause an action.
- Variable declarations; COM, DIM, REAL, and INTEGER. These are executed during prerun and skipped over at run-time. The OPTION BASE statement is part of the declaring process.
- DATA statements. These are accessed by the READ statement, not executed.
- SUB and DEF FN statements. These are used during prerun to establish the program structure and are skipped over at run-time.
- Structuring statements, such as LOOP, END LOOP, ELSE, END IF, etc. These are matched and checked for proper nesting at prerun.

Live Keyboard

When a program is running, the keyboard is still active. Commands can be executed, variables can be inspected and changed, and the state of the computer can be changed. The term "live keyboard" is used when talking about commands that are executed during a running program. One of the principal uses for live keyboard commands is the troubleshooting and debugging of programs in the development stage, as discussed in the "Debugging Programs" chapter of *BASIC Programming Techniques*. The discussions presented here are intended to demonstrate the various machine states:

- Running
- Paused
- Stopped

Pausing and Stopping Programs

If the computer operator does not intervene, a program will run until it reaches an END, STOP, or PAUSE statement, or until it pauses to input data or report an error. If you wish to pause or stop a program before its normal completion, use the following keys:

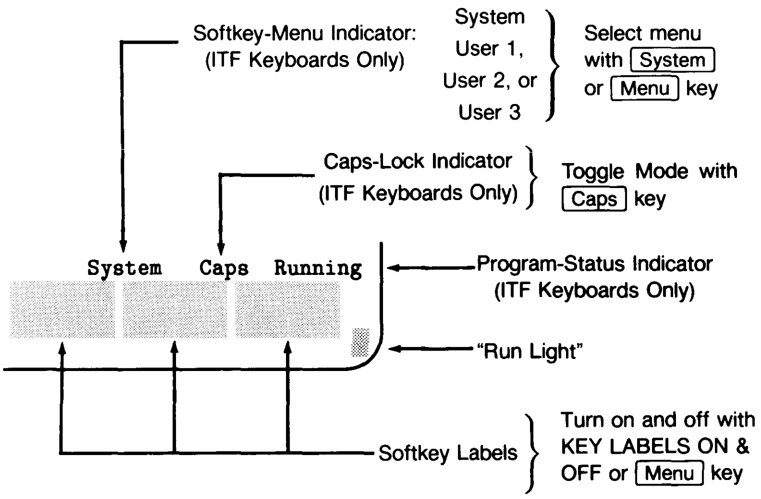
Large Keyboard	ITF Keyboard	Small Keyboard	Effect
PAUSE	Stop	PSE	Pauses program execution after the computer finishes the line it is on and any I/O operations in progress. This is useful for pausing a program that is executing an INPUT statement. It leaves all necessary internal information intact, so that program execution can be resumed again with the CONTINUE key (see below) or CONT command.
CONTINUE	f2 (System menu)	CONT	Pressing CONTINUE after a PAUSE (key or statement) causes program execution to resume in a normal manner from the place where it was paused.

Large Keyboard	ITF Keyboard	Small Keyboard	Effect
CLR I/O	Break	C I/O	Aborts any I/O statement in progress (such as ENTER or TRANSFER) and pauses the program. The program counter is returned to the beginning of the aborted I/O statement, so that the CONTINUE key or the CONT command cause the program to resume program execution beginning with that same statement. This is useful when the computer is "hung" trying to output to a device that is down.
STOP (SHIFT-CLR I/O)	Stop (Shift-Stop)	STOP (SHIFT-C I/O)	Stops the program after the computer finishes executing the line it is on (and returns the program to the main context, if executing a subprogram or user-defined function). However, it does not affect the interfaces, the CRT, program memory, the values of variables, tabs, or the RECALL key's buffer. CONTINUE is not allowed after a STOP.
RESET (SHIFT-PAUSE)	Reset (Shift-Break)	RST (SHIFT-PSE)	This is the most drastic and complete way to halt program execution. The program stops immediately, all I/O operations are aborted, any open files are closed, and all interface cards are reset. However, the printout area of the CRT, program or variable memory, tabs, and the RECALL key's buffer are not affected. CONTINUE is not allowed after a RESET.

The easiest way to remember the key definitions on the ITF keyboard is to use the keyboard overlay and keep the softkey labels on. (To turn softkey labels on, either press the Menu key or execute the KEY LABELS ON statement.)

Program-Status Indicators

The current state of the computer is indicated in the lower right-hand corner of the CRT.



The character in this corner is referred to as the "run light". The following table shows the various indications of the run light and their meaning.

Status Indicator[1]	Run Light	System State
`Idle`	*(blank)*	Program stopped; can execute commands; CONTINUE not allowed.
`Running`	▒	Program running; can execute commands.
`Paused`	_	Program paused; can execute commands; CONTINUE *is* allowed.
`Transfer`	`IO`	Program paused, but an overlapped TRANSFER (I/O) operation is still in progress; can also execute commands.
`Input?`	?	BASIC program waiting for input from keyboard; cannot execute commands.
`Command`	*	System executing command entered from keyboard; can enter 1 more command, but it will not be executed until after the current command is completed.

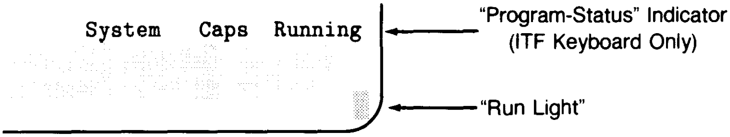
[1] These indicators are displayed only if softkey labels are currently on. Use the KEY LABELS ON statement (or the Menu key on an ITF keyboard) to turn these labels on.

Example of Controlling Program Execution

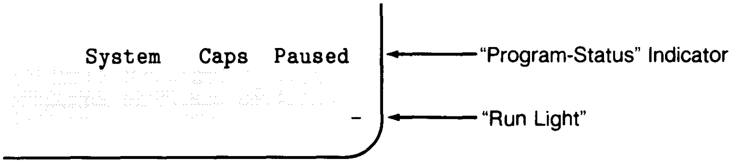
To demonstrate some of the interaction between a program and the keyboard, enter the following simple program.

```
10  DISP "Next command?"
20  X=0
30  PRINT X;
40  X=X+1
50  WAIT .1
60  GOTO 30
70  END
```

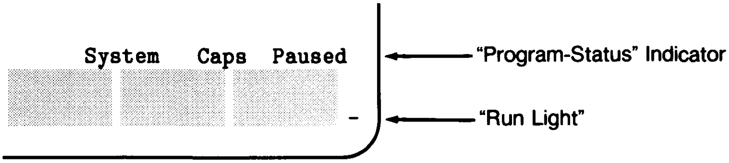
1. After you have entered the program, execute RUN and observe the CRT. Notice that the DISP message appears in the display line, the printout area fills with a sequence of numbers, and the run light indicates that a program is running.



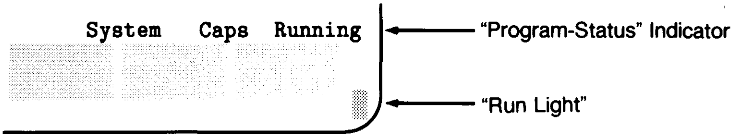
2. Press [PAUSE] ([Stop] on an ITF keyboard). The printout of numbers stops, and all the data on the CRT remains unchanged. The run light now indicates that the program is paused and can be continued. The program line that appears at the bottom of the CRT is the next line that will be executed when program execution resumes.



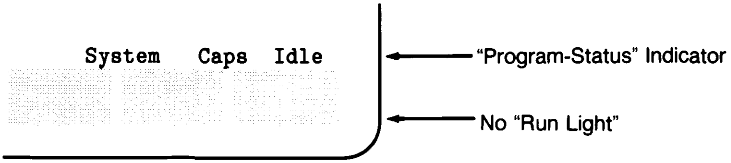
3. Press [STEP] ([f1] in the System menu of ITF keyboard) a few times. The program is executed one line at a time, as indicated by the lines changing at the bottom of the CRT. Notice that the program is still paused and continuable after each press of the [STEP] key. The [STEP] key can be a great help when you are trying to find certain kinds of problems. The "Debugging Programs" chapter of *BASIC Programming Techniques* gives the details of this and other debugging tools.



4. Press [CONTINUE] ([f2] in the System menu of an ITF keyboard). The printout on the CRT resumes with the next number in the sequence. The run light again indicates that a program is running.

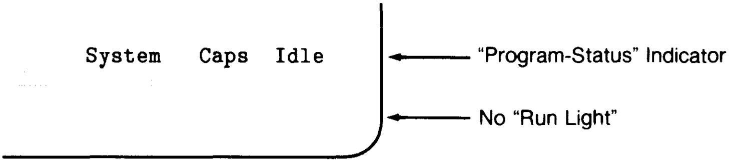


5. Press [Shift]-[Stop] (ITF keyboard) or [STOP]. The printout of numbers stops, and all the data on the CRT remains unchanged. However, the run light is off, indicating a stopped condition.

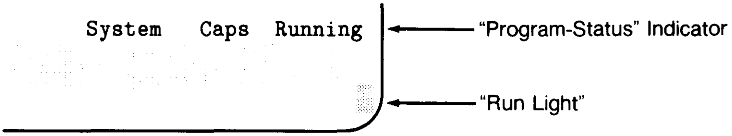


6. Press CONTINUE (f2 in the System menu of an ITF keyboard). An error results, because a stopped program cannot be continued.

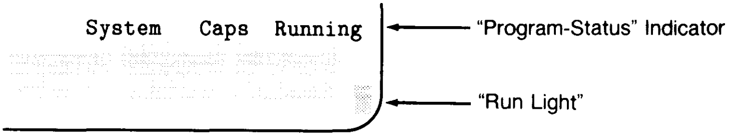
   ```
   Error 122: Program not continuable
   ```



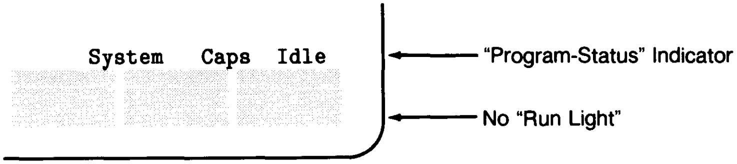
7. Press RUN(f3 in the System menu of an ITF keyboard). The program runs again, but the number sequence has restarted from the beginning, not from the next number in the seguence. RUN causes the program to restart, not resume.



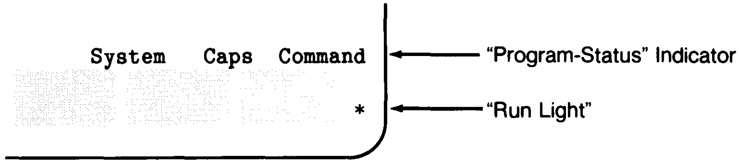
8. Type `X=1` and press Return or ENTER. Notice that the numbers being printed start over with "1". The live keyboard was used to change the value of "X", and the program used the new value from the keyboard.



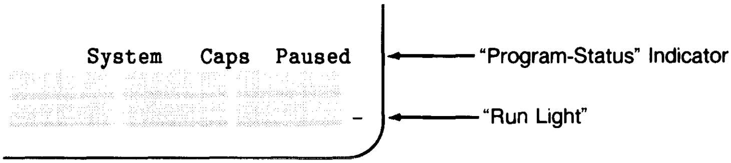
9. Press [Reset]. The program stops and the data remains in the printout area, but the display line is cleared and the message **BASIC Reset** appears at the bottom of the CRT. Although the clearing of the display line seems like a minor effect, it indicates an important point. [Reset] and [Stop] have different effects on interfaces and peripheral devices. This aspect of [Reset] is summarized in the "Reset Tables" in the "Useful Tables" appendix of the *BASIC Language Reference* and is discussed fully in the *BASIC Interfacing Techniques* manual.



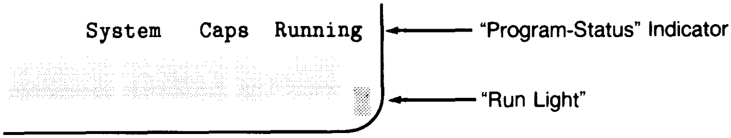
10. Press [RUN] ([f3] in the System menu of an ITF keyboard). Then type **WAIT 5** and press [Return] or [ENTER]. Notice that the run light changes to indicate that a keyboard command is being executed and the printout is delayed for five seconds while the live keyboard command is processed. Actually, the run light changed when the X=1 command was executed in step 8, but it may have happened so fast that you didn't see it.



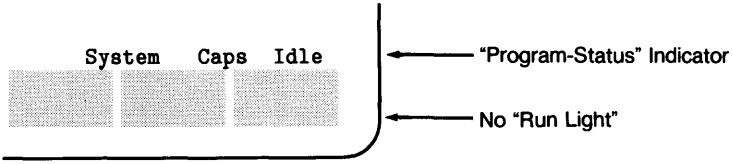
11. Press PAUSE (Stop on an ITF keyboard) and then type `EDIT` and press Return or ENTER. The display on the CRT changes to show the program. The line you were editing last appears in the current-line position. Notice that the run light is still visible in the lower right-hand corner and it indicates that the program is paused.



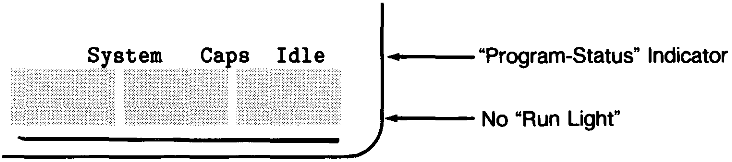
12. Press CONTINUE (f2 in the System or User 2 menu of ITF keyboard). The CRT returns to normal mode, and the printout of numbers continues in sequence. However, the previous data on the display was lost when the CRT was used for EDIT mode.



13. Press [PAUSE] ([Stop] on an ITF keyboard). Then type `EDIT 50` and press [Return] or [ENTER]. The CRT changes to EDIT mode and the program appears again. This time, line 50 is in the current-line position. Notice that the run light indicates that the program is paused. Change line 50 to `WAIT .2` and press [Return] or [ENTER]. The new line 50 is entered, but the run light changes. Editing the program caused it to move from the paused state to the stopped state.



14. Press [CONTINUE] ([f2] in the System menu of an ITF keyboard). An error results. As mentioned earlier, a program can be viewed while it is paused, but it cannot be changed. Once any program line has been changed, the program is no longer paused, and [CONTINUE] is not allowed.



This simple demonstration covers most of the highlights of live keyboard, program states, and the run light. The "waiting for input" indication can be seen when using the INPUT and LINPUT statements described in the chapter of *BASIC Programming Techniques* called "Communicating with the Operator". The "paused with I/O completing" indication is not described in this manual. It is a special state that results from the use of TRANSFER and is discussed in the *BASIC Interfacing Techniques* manual.

What to Do Next

Task/Topic	Chapter Number
Learn about how to load and run programs.	9
Learn about how to use and manage files and directories.	10
Learn about how to maintain your system.	12 thru 17
Learn about utilities available with the BASIC system.	18
Learn about each key on your keyboard.	19 thru 21

Using Directories and Files 10

The main use of a computer is to run programs that process some sort of data. Both programs and data are stored in files, which reside in mass storage volumes. The files in a particular volume are listed (and described) in directories. This chapter describes general management of directories and files.

Finding and Specifying Files

Files are collections of information—programs or data—accessed by a single name. Volumes are collections of files. (For an introductory explanation of volumes and files, see the "Mass Storage Concepts" chapter of this manual.) This section shows how to locate and specify BASIC files.

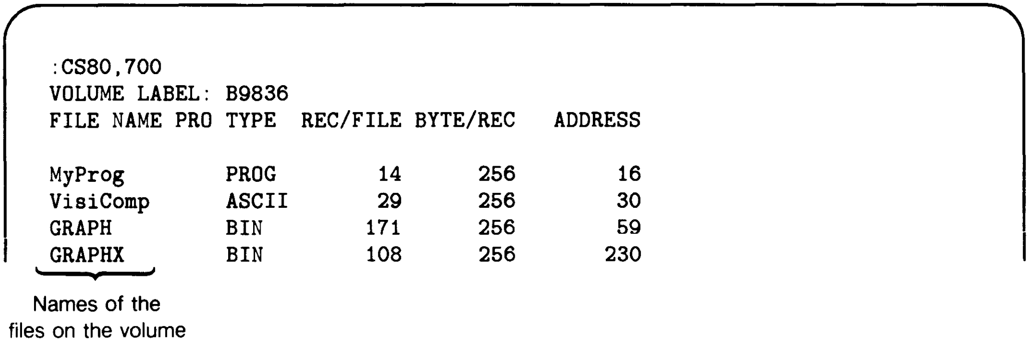
Is the File on the Default Volume?

Use the CAT (catalog) statement to determine the names of the files on a volume.

If you do not specify a volume, the default volume will be assumed. (Examples of specifying a volume other than the default volume are presented subsequently.)

```
CAT
```
Return or ENTER

You should get a listing similar to this, which you can visually scan for the presence of the desired file(s).



```
:CS80,700
VOLUME LABEL: B9836
FILE NAME PRO TYPE  REC/FILE BYTE/REC   ADDRESS

MyProg        PROG        14      256        16
VisiComp      ASCII       29      256        30
GRAPH         BIN        171      256        59
GRAPHX        BIN        108      256       230
```

Sending Catalogs to External Printers

The CAT statement normally directs its output to the current PRINTER IS device (the default is the CRT display). The CAT statement can also direct the catalog to a specified device, as shown in the following examples:

```
CAT TO #701
CAT TO #External_prtr
CAT TO #Device_selector
```

The parameter following the # is known as a device selector and is further described in the chapter of *BASIC Programming Techniques* called "Using a Printer," and in the Glossary of the *BASIC Language Reference.*

Specifying Files on the Default Volume

To specify a file on the **default volume**, merely use its name. Here is an example of loading the file named "MY_PROG" from the current default volume:

```
LOAD "MY_PROG"
GET  "PROG_ASCII"
PURGE "A_FILE"
```

If the file is *not* on the default volume or not in the current working directory (if using a hierarchical directory), then you will need to do one of the following things:

- Identify the volume in the file specifier. For instance:

  ```
  LOAD "MY_PROG:,700"
  ```

- Change the current working directory and/or default volume. For instance:

  ```
  MASS STORAGE IS "/USERS/MARK"
  LOAD "MY_PROG"

  MASS STORAGE IS ":,700"
  LOAD "MY_PROG"
  ```

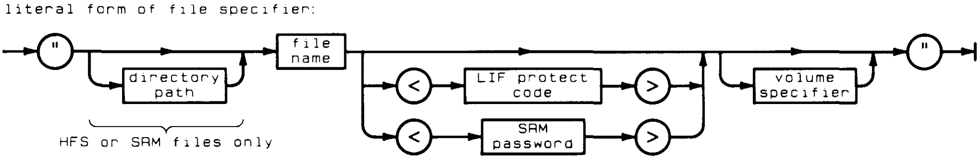
Let's first take a closer look at the alternative of specifying the volume with the file name.

Directory, File, and Volume Specifiers

Files can be uniquely identified by specifying the following information:

- The directory in which the file resides (if it is in a hierarchical directory structure).
- The file's name. (If a LIF or SRM file is currently "protected", you will need to include the "protect code" or "password" with the file name. If an HFS file is "protected", you will need to make sure you have the correct access permission. See the subsequent section of this chapter called "Protecting Files" for details.)
- The volume on which it resides. (If the file is on the default volume, then you don't need to specify the volume; BASIC *assumes* the file is on the *default volume* when you don't specify one.)

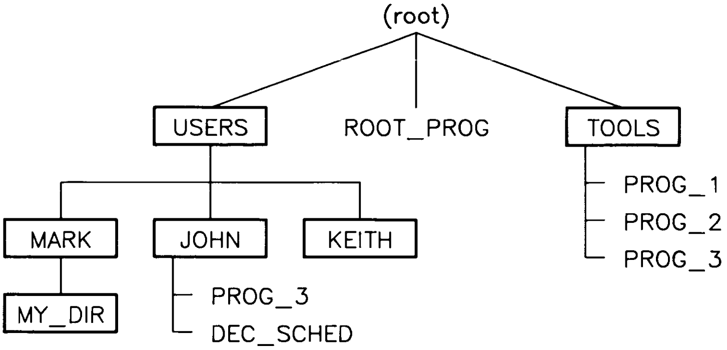
Here is a syntax drawing of the components of a file specifier:



Files Not in the Current Working Directory (Hierarchical Directories Only)

If you want to specify a file (in a hierarchical directory), but it is not in the *current working directory*, then you must include this information in the file specifier.

Here is an example directory structure that will be used in the subsequent examples in this section. Note that none of the examples change the default volume; all remain in the directory structure of the current default volume. (The next section shows examples of changing the current default volume.)



For instance, this example catalogs the root directory of the current default volume:

```
CAT "/"
```

This example loads the file “MY_PROG” from the root directory:

```
LOAD "/ROOT_PROG"
```

This example loads the file “PROG_3” from the directory named “MARK”, which is subordinate to the directory named “MARK” in the root:

```
LOAD "/USERS/MARK/PROG_3"
```

This example catalogs the “USERS” directory:

```
CAT "/USERS"
```

This example catalogs the directory which is superior to the current working directory:

```
CAT ".."
```

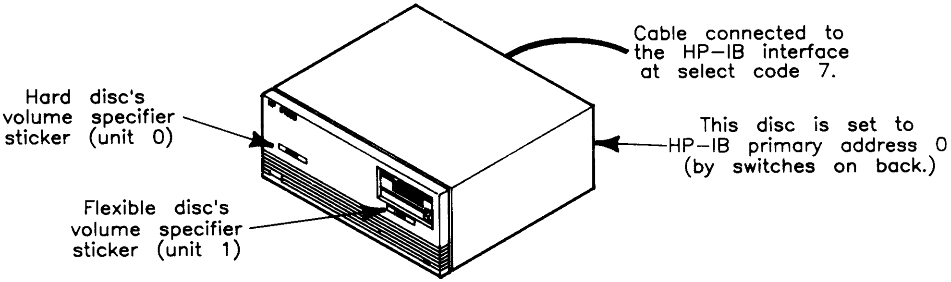
This example specifies the file named “DEC_SCHED”, located in the sub-directory named “MARK”, which is itself a sub-directory in the root-level directory named “USERS”:

```
LOAD "/USERS/MARK/DEC_SCHED"
```

Files Not on the Default Volume

If the file is not on the *default volume*, then you must include this information in the file specifier. The *volume specifier* is the information on the label which you should have affixed to each of your mass storage volumes in the procedure in the "Verifying and Labeling Peripherals" chapter of this manual.

Here is a pictorial description of a typical hardware configuration (a device with both a hard and a flexible disc drive):


Hard disc's volume specifier sticker (unit 0)
Flexible disc's volume specifier sticker (unit 1)
Cable connected to the HP-IB interface at select code 7.
This disc is set to HP-IB primary address 0 (by switches on back.)

The following LOAD statement specifies the file named "MY_PROG", located on the volume connected to the computer through the HP-IB interface at select code 7, with primary address 0, and unit number 1:

```
LOAD "MY_PROG:HP9133,700,1"
```

or

```
LOAD "MY_PROG:,700,1"
```

This CAT statement catalogs the hard-disc volume in the above example; it is at the same select code and primary address, but it has a unit number of 0 (which is the default when the unit number is not specified):

```
CAT ":,700"
```

This example catalogs another volume (at select code 8, primary address 2, unit number 1):

```
CAT ":,802,1"
```

Changing the Default Volume and Current Working Directory

The following statement sets the system mass storage to an HP 9133 drive at interface select code 7 with primary address 0; unit number 1 specifies the flexible-disc drive:

```
MASS STORAGE IS ":HP9133,700,1"
 or
MSI ":,700,1"
```

If the volume has a hierarchical directory structure (HFS and SRM volumes), then you may specify a *current working directory*:

```
MSI "/USERS/MARK:HP9133,700,1"
 or
MSI "/USERS/MARK:,700,1"
 or
MSI "/USERS/MARK"
```

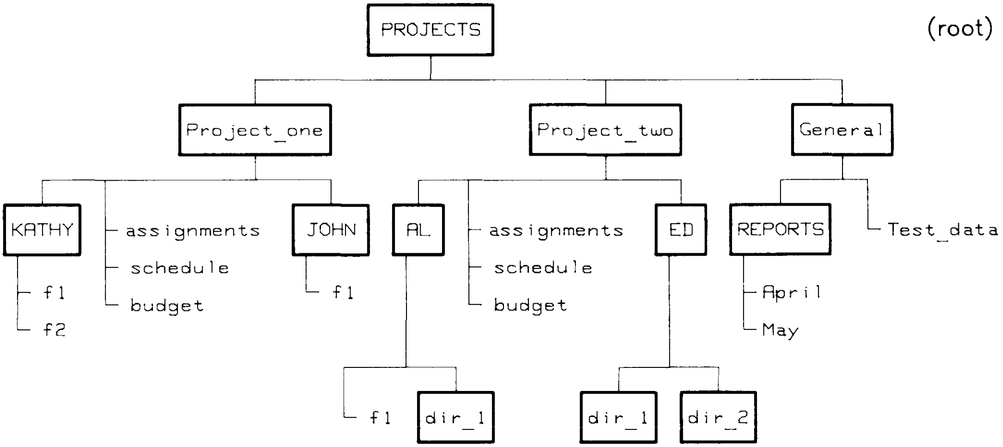
If the default volume is ":,700,1"

Creating and Using Hierarchical Directories

Directories contain information about files on a volume. If you are on a hierarchical-directory volume (such as HFS or SRM), directories also have additional capabilities. This section shows how to create and access hierarchical directories.

Example Hierarchy

Examples in the following paragraphs refer to the directory structure shown in the illustration below.



The boxed names signify directories, while the unenclosed names signify files.

Changing the Default Volume

First, it will be helpful to change the default volume to the "root" directory of your hierarchical volume by executing something like this:

```
MSI ":,700"        or
MSI ":,21,0"
```

You do not need to specify a "/", since the directory path is assumed to begin at the root directory when the volume specifier is included.

Adding Another Directory

This statement creates a directory named **CHARLIE** in this directory structure:

```
CREATE DIR "/PROJECTS/Project_one/CHARLIE"
```

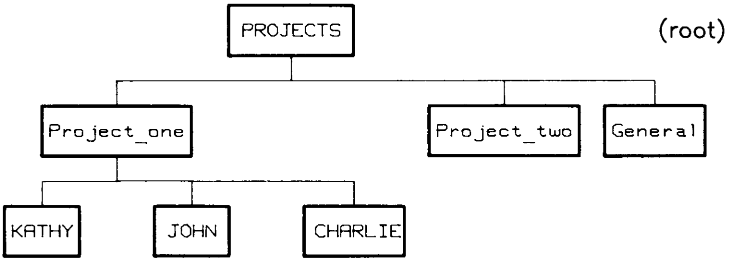
The leading slash indicates that the directory path begins at the root of the hierarchical directory structure. Each subordinate directory is listed from left to right, separated by slashes (**PROJECTS/Project_one/**). The directory being created is listed after the directory path (**CHARLIE**).

You could accomplish the same results with the following statements:

```
MSI "/PROJECTS/Project_one"

CREATE DIR "CHARLIE"
```

Note that in both of the preceding examples, the leading "/" was unnecessary since the current working directory was already the root directory.

This statement would place your newly created directory into the directory structure as shown below.



Creating Files and Other Directories Under a Directory

To create files subordinate to a new directory, you may either establish the new directory as the working directory or specify the directory path to that directory. Assuming your current working directory is the root, you could type:

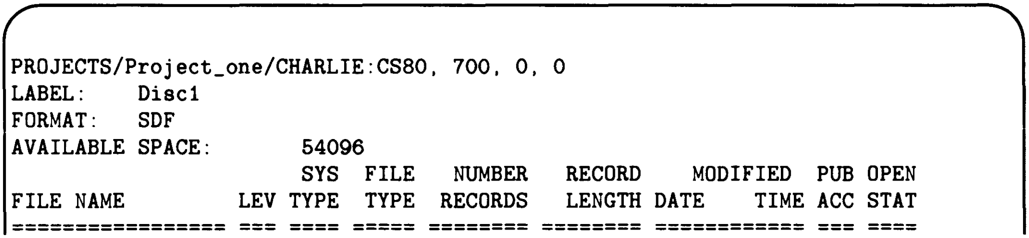
```
MSI "PROJECTS/Project_one/CHARLIE"
```

to move into the directory **CHARLIE**.

You could verify the new working directory with a catalog listing by typing:

```
CAT
```

On a computer whose screen supports an 80-character line width, the resulting listing would look something like this:



```
PROJECTS/Project_one/CHARLIE:CS80, 700, 0, 0
LABEL:    Disc1
FORMAT:   SDF
AVAILABLE SPACE:       54096
                       SYS  FILE   NUMBER   RECORD    MODIFIED  PUB OPEN
FILE NAME          LEV TYPE  TYPE  RECORDS   LENGTH DATE    TIME ACC STAT
================= === ==== ===== ======== ======== ============ === ====
```

To create an ASCII file within CHARLIE, which is named ASCII_1 and is initially to contain 100 records, execute this statement:

```
CREATE ASCII "ASCII_1",100
```

To create a BDAT file within CHARLIE, which is named BDAT_1 and is initially to contain 25 records, execute this statement:

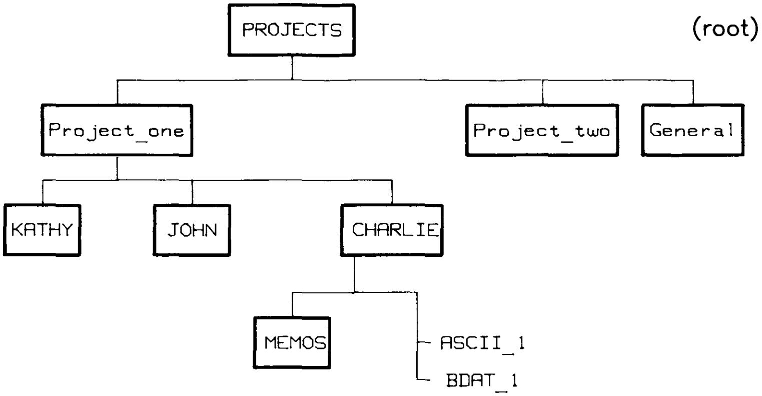
```
CREATE BDAT "BDAT_1",25
```

(When no record size is specified in the CREATE BDAT statement, the default 256-byte record size is assumed.)

To create another directory within CHARLIE called MEMOS, execute this statement:

```
CREATE DIR "MEMOS"
```

The additions would make the directory structure look like this:



The simplest form of the CAT statement:

```
CAT
```

lists the contents of the current working directory because no directory is specifically identified. If no directory name is shown in the directory header, the current working directory is the root.

If you wanted to list the contents of **CHARLIE**, but your current working directory was **not** **CHARLIE**, you could:

- Designate **CHARLIE** as the working directory with the MSI statement, then use the CAT statement's "short form." For example:

  ```
  MSI "PROJECTS/Project_one/CHARLIE:REMOTE"
  CAT
  ```

- In the CAT statement, specify the entire path to **CHARLIE**, starting at the root, by beginning the path name with a slash (/). For example:

  ```
  CAT "/PROJECTS/Project_one/CHARLIE"
  ```

  This form assumes that you have already designated remote mass storage with some form of the MSI ":REMOTE" statement. If you have not, use the form:

  ```
  CAT "PROJECTS/Project_one/CHARLIE:REMOTE"
  ```

  The leading slash is not necessary, because including :REMOTE specifies the root as the beginning of the path.

- If you were in MEMOS (the directory immediately subordinate to CHARLIE), you could use the ".." notation (explained with directory path syntax in the "BASIC Language Reference" section of this chapter). For example:

```
    CAT ".."
```

A Closer Look at Hierarchical Directory Capabilities

Directories are a type of file and, as such, can be:

- created with the CREATE DIR statement. When a directory is created, its location in the hierarchical structure is fixed.
- cataloged with the CAT statement, renamed with the RENAME statement, and protected with the PROTECT statement.
- "filled" with subordinate files and directories using the COPY, CREATE, CREATE BDAT, CREATE ASCII, CREATE DIR, SAVE, STORE, RENAME, RE-SAVE, and RE-STORE statements. Each subordinate file or directory is described in a fixed-format record in its superior directory.
- opened and closed with the MASS STORAGE IS (MSI) statement. When a user's MSI statement specifies a directory, any previously opened directory of that user is closed and the new one is opened.
- "emptied" by removing all subordinate files and directories with the PURGE statement.
- purged with the PURGE statement. You must close and empty a directory before purging it.

How SRM and HFS Directories and Files Are Stored

To most efficiently use the shared disc space, the SRM system and HFS system store files non-contiguously and adds to space allocations for files as needed.

Non-Contiguous Storage of SRM and HFS Files

To avoid wasting disc space, the SRM system may "fragment" a file to fill unused disc sectors. This process is transparent and cannot be externally controlled. By "filling the gaps" automatically, the system eliminates the need to pack the shared disc's files.

Space Allocation for SRM and HFS Directories and Files

SRM and HFS files and directories grow dynamically as data is entered into them. This type of file is called *extensible*, because its size may be automatically extended (by BASIC) whenever it would otherwise overflow. (The amount of space added to the file's current size is known as the "extent size" of the file; this amount of space is the same as the amount of space that was originally allocated to the file when it was initially created.)

Rather than restricting a file's space to that allocated when the file is created (for example, with a CREATE statement), the system determines disc space requirements when data is sent to the file (for example, by an OUTPUT statement). If additional data placed into a file would cause the file to overflow its current space allocation, the system automatically allocates more space for the file.

Similarly, directories grow only as entries are added. As a file or directory is created, another 24-byte record is added to the containing directory.

Files are extended as long as there is sufficient unused disc space on the same volume. Excess data from a file will not be placed on any other volume, however.

A Closer Look at File Catalogs

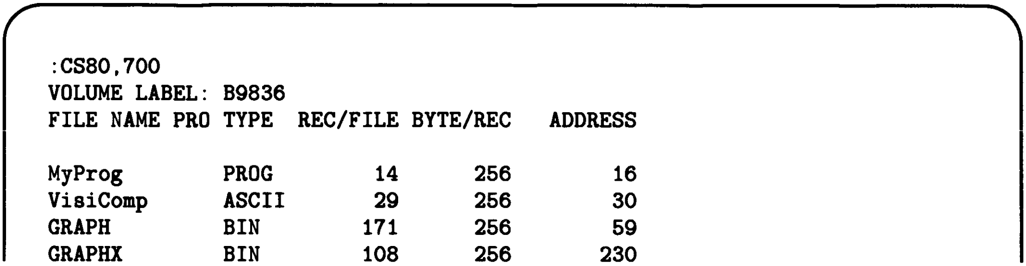
There are three types of directory structures available with BASIC. (For further information about each of these directory formats, see the "Mass Storage Concepts" chapter.)

- LIF (Logical Interchange Format)—an HP corporate standard used by many HP operating systems; allows discs and files to be transportable between many different types of computer systems.
- HFS (Hierarchical File System)—also provides a format used by several operating systems, most notably the HP Series 200/300 HP-UX and Workstation Pascal systems. Like LIF, it also allows transporting discs and files between operating systems; however, it also provides a hierarchical directory structure, which LIF does not.
- SDF (Structured Directory Format)—also a hierarchical directory format; however, with Series 200/300 computers, it is available only with SRM "file server" systems.

BASIC displays slightly different results for each type of directory structure.

LIF Catalogs

Here is a typical catalog listing of a LIF directory:



```
:CS80,700
VOLUME LABEL: B9836
FILE NAME PRO TYPE  REC/FILE BYTE/REC   ADDRESS

MyProg        PROG        14      256        16
VisiComp      ASCII       29      256        30
GRAPH         BIN        171      256        59
GRAPHX        BIN        108      256       230
```

Here is what each portion of the catalog means:

`:CS80,700` is the mass storage volume specifier (msvs) of the device.

`VOLUME LABEL` is the "name" given to the volume (in this case, it is B9836).

`FILE NAME` lists the names of the files in the directory.

`PRO` indicates whether the file has a protect code (* is listed in this column if the file has a protect code).

`FILE TYPE` lists the type of each file.

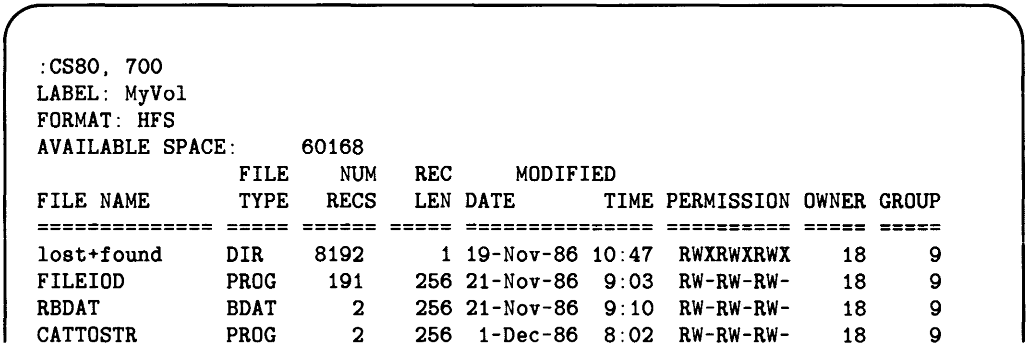
`REC/FILE` indicates the number of records (or sectors) in the file.

`BYTE/REC` indicates the record size.

`ADDRESS` indicates the number of the beginning sector in the file.

HFS Catalogs

Here is a typical catalog listing of an HFS directory:



```
:CS80, 700
LABEL: MyVol
FORMAT: HFS
AVAILABLE SPACE:      60168
                FILE    NUM    REC     MODIFIED
FILE NAME       TYPE   RECS    LEN DATE         TIME PERMISSION OWNER GROUP
============== ===== ====== ===== ============== ========== ===== =====
lost+found      DIR    8192      1 19-Nov-86 10:47  RWXRWXRWX    18     9
FILEIOD         PROG    191    256 21-Nov-86  9:03  RW-RW-RW-    18     9
RBDAT           BDAT      2    256 21-Nov-86  9:10  RW-RW-RW-    18     9
CATTOSTR        PROG      2    256  1-Dec-86  8:02  RW-RW-RW-    18     9
```

Here is what each column means:

`FILE NAME` Lists the name of the file.

`FILE TYPE` Lists the file's type (for instance, DIR specifies that the file is a directory; PROG specifies a BASIC program file; BDAT specifies a BASIC DATA file; etc.).

`NUM RECS` number of *logical* records (the number of records allocated to the file when it was created).

`REC LEN` the *logical* record size (default is 256 bytes for all files; BDAT, ASCII, and HPUX files can all have user-selectable record lengths).

`MODIFIED DATE TIME` the day and time when the file was last modified.

`PERMISSION` specifies who has access rights to the file:

`R` indicates that the file can be read;
`W` indicates that the file can be written;
`X` indicates that the file can be searched
(meaningful for hierarchical directories only).

There are 3 classes of user permissions for each file:

OWNER (left-most 3 characters);
GROUP (center 3 characters);
OTHER (right-most 3 characters).

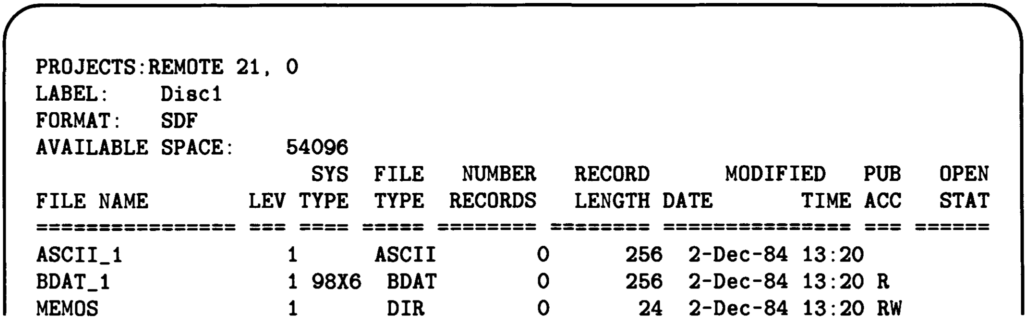
See the subsequent section called "HFS File and Directory Permissions" for further information).

OWNER specifies the owner identifier for the file (for BASIC files, the default owner identifier is always 18).

GROUP specifies the group identifier of the file or directory (for BASIC, the default group identifier is always 9, which is used for "workstations" such as Series 200/300 BASIC and Pascal).

SRM Catalogs

Here is a typical catalog listing of an SRM directory:



```
PROJECTS:REMOTE 21, 0
LABEL:    Disc1
FORMAT:   SDF
AVAILABLE SPACE:    54096
                      SYS  FILE   NUMBER   RECORD      MODIFIED   PUB   OPEN
FILE NAME        LEV TYPE  TYPE  RECORDS   LENGTH DATE       TIME ACC   STAT
================ === ==== ===== ======== ======== ==============  === ======
ASCII_1             1      ASCII        0      256  2-Dec-84 13:20
BDAT_1              1 98X6  BDAT        0      256  2-Dec-84 13:20 R
MEMOS               1       DIR         0       24  2-Dec-84 13:20 RW
```

Here is what the various columns mean:

FILE NAME is the name of the file (up to 16 characters)

LEV is always 1 (for BASIC).

SYS TYPE indicates the file type (such as PROG, ASCII, BDAT, etc.)

NUMBER RECORDS is the number of records in the file.

RECORD LENGTH is the record length used in the file (file size is the product of RECORDS×LENGTH).

MODIFIED DATE TIME Date and time of day when the file was last written or modified.

PUB ACC shows which access rights are currently public. For instance, MR would indicate that Manager and Read capabilities are public, while other rights are protected (and require a password to access them). See the subsequent section called "SRM Passwords" for details.

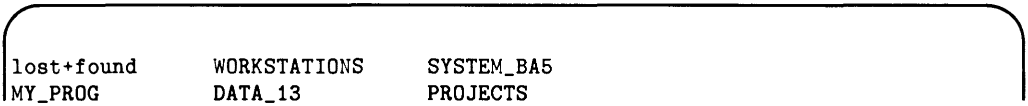
`OPEN STAT` is the current OPEN/CLOSED/LOCKED status of the file.

- OPEN means that someone currently has the file open.
- CLOSED means that no one is currently using the file.
- LOCKED means that the file is opened in "EXCLUSIVE" mode, and no one else may access the file.

Listing Only File Names

The following statement will produce a multi-column listing of the names of the files in the current working directory of the current default volume.

```
CAT ; NAMES Return
```



```
lost+found      WORKSTATIONS     SYSTEM_BA5
MY_PROG         DATA_13          PROJECTS
```

Cataloging Selected Files

You can also specify which files to list with some of the following options to the CAT statement:

`CAT ; SELECT "ABC"` Lists only the files beginning with the specified letters "ABC".

`CAT ; SKIP 30` Skips the first 30 files in the directory and lists only the remaining files.

`CAT ; COUNT Num_files` Stores the number of lines in the catalog in the numeric variable called `Num_files`. (Note that this variable must be defined in the current program or subprogram context before it can be used.)

`CAT ; NO HEADER` Suppresses the catalog heading.

`CAT ; NO HEADER, SKIP 10` Suppresses the catalog heading and skips the first 10 files in the directory.

See the *BASIC Language Reference* for a complete description of these options.

Cataloging to a String Array

You can also send a catalog to a string array, as shown in the following example program segment:

```
      .
      .
      .
170  DIM Str_array$(1:40)[80]  ! 40 lines of 80 columns each.
180  CAT TO Str_array$(*)      ! Send catalog to string.
190   .
      .
      .
```

See the "Data Storage and Retrieval" chapter of *BASIC Programming Techniques* for further information on catalogs sent to string arrays.

Cataloging Individual PROG Files

Performing CAT operations on an individual PROG file returns additional information about the file. A catalog of a PROG files yields the following information:

- A list of the binary program(s) contained in the program file and the size of each (in bytes)
- The size of the main program (in bytes).
- A list of contexts (SUB and FN subprograms) and their sizes (in bytes)

The following catalog listing is an example of a CAT performed on an individual PROG file. Note that this catalog format only requires 45 columns.

```
NEWPAGER_A
NAME                    SIZE TYPE
====================== ===== ================
MAIN                   62002 BASIC
FNBar$                  3680 BASIC
FNRoman$                 656 BASIC
Killkeys                 426 BASIC
FNTrim$                  414 BASIC
FNUpc$                   344 BASIC
FNLwc$                   416 BASIC
Table_formatter         6810 BASIC
Strip                   1260 BASIC
  AVAILABLE ENTRIES = 0
```

The **AVAILABLE ENTRIES** table entry is not currently used.

The following listing shows a program which was stored while a BIN program was resident in the computer.

```
NEWPAGER_B
NAME                    SIZE TYPE
====================== ===== ================
PHYREC (2.0) Rev A      1734 BASIC BINARY
MAIN                   56394 BASIC
FNBar$                  3218 BASIC
FNRoman$                 656 BASIC
Killkeys                 426 BASIC
FNTrim$                  414 BASIC
FNUpc$                   344 BASIC
FNLwc$                   374 BASIC
Table_formatter         7622 BASIC
  AVAILABLE ENTRIES = 0
```

In addition, if the currently loaded BASIC system version is **different** from the binary program version, a warning and the version codes of both BASIC system and binary program are included in the catalog information. The following example shows the format of the message returned.

```
Prog_phy
NAME                    SIZE TYPE
====================== ===== ================
PHYREC 1.0              1734 BASIC BINARY
*** WARNING:  System level 2.  Bin level 1.
MAIN                     222 BASIC
  AVAILABLE ENTRIES = 0
```

General File Management Operations

This section describes the mechanics of managing files in your system. These may be program files, data files that your application creates, or data files that you create from the keyboard.

Closed vs. Open Files and Hierarchical Directories

Many of the following operations can only be performed on closed files and directories. Here is what the term "closed" means for files and directories:

- Files are open when there is an

  ```
  ASSIGN @Io_path TO *
  ```

  currently active for the file.

  Files are closed by this statement:

  ```
  ASSIGN @Io_path TO *
  ```

- Directories are closed when they are not the *current working directory.* A directory is the current working directory when you have made it the MASS STORAGE IS directory:

  ```
  MASS STORAGE IS "/USERS/MARK/MY_DIR"
  ```

The SCRATCH A command also closes any currently open directories and files. All files except those opened with the PRINTER IS statement are also closed by pressing Reset (Shift-Break on an ITF keyboard, or SHIFT-PAUSE on a 98203 keyboard).

See the *BASIC Language Reference* description of these statements for details.

Protecting Files

You can "protect" files from being read, over-written, or destroyed by other users of the system. Since protecting files is slightly different for each of the three directory types, the discussion is split into three parts.

LIF Protect Codes

With LIF directories, protect codes are two-character strings that can be assigned to any BDAT, BIN, HP-UX, or PROG file with the PROTECT statement. The protect code, which does not appear in the CAT display, must be specified to subsequently modify the file. Protect codes are not unbreakable; they are only intended to prevent accidentally writing and purging files.

For example, to protect the file "SECRET" with the protect code "BS", use the statement:

```
PROTECT "SECRET","BS"
```

The protect code must subsequently be specified with the file name to allow access. For example, to RENAME the previously protected file "SECRET", the statement is:

```
RENAME "SECRET<BS>" TO "SHHHH"
PROTECT "SHHHH","BS"
```

File specifiers in mass storage statements that *write* to a file or directory must include the protect code, if the file has one. Mass storage statements that *read* a file or directory do not require the protect code (e.g., CAT, LOAD, LOAD BIN, LOADSUB, GET, and COPY).

To assign an I/O path name to the file named "SHHHH," you would now have to include the protect code.

```
ASSIGN @Path1 TO "SHHHH<BS>"
```

If you assign a protect code longer than two characters, the system will ignore everything after the second (non-blank) character. For example, the protect codes LONGPASS, LOST, and LOLLYGAG all result in the same protect code: LO. This rule holds both for PROTECTing a file and for specifying the protect code in a file specifier. For instance:

```
PROTECT "FILE1","Protect1"
```

would assign the protect code "Pr" to FILE1. To rename the file, we could write:

```
RENAME "FILE1<Prattle>" TO "FILE2"
```

"Prattle" is an acceptable protect code, since it starts with "Pr." Note that we do not include a protect code in the new file name. If you do, the system ignores it since the old protect code is passed to the new file name. FILE2 still has the protect code "Pr". To rename the file again, we might write:

```
RENAME "FILE2<Pr>" TO "FILE3"
```

Renaming a file has the effect of changing the file name in the directory and leaving everything else intact.

In addition to using the PROTECT statement, you can also assign a protect code to a BDAT file when you create it. For example:

```
CREATE BDAT "Example<xx>",10
```

The statement creates 10-record files called "Example" and gives it a protect code of "xx".

You can also do this to PROG files with the STORE and STORE BIN statements. However, since ASCII files cannot be protected, a protect code cannot be included in any CREATE ASCII, SAVE, or RE-SAVE statement.

To change a protect code, simply execute a new PROTECT statement. To change the protect code of "Example" to "yy," execute:

```
PROTECT "Example<xx>","yy"
```

Note that you must include the current protect code in the file specifier.

To completely remove a protect code from a file, PROTECT the file with a code consisting of two blanks. For example, to remove the protect code from file "Example," execute:

```
PROTECT "Example<yy>","  "
```

When specifying a file that does not have a protect code, you can either ignore the code entirely or include a code of two spaces:

```
PURGE "Example"
     or
PURGE "Example<  >"
```

HFS File and Directory Access Permissions

For HFS directories, you can use the PERMIT statement to assign and remove access permissions of a file or directory. Since this file system is compatible with the HP-UX system, BASIC uses a subset of the HP-UX file protection mechanism. (With HP-UX, the *chmod* command performs this function.)

There are 9 "permission bits" for HFS files and directories, broken into three classes (one for each class of users):

OWNER			GROUP			OTHER		
READ	WRITE	SEARCH	READ	WRITE	SEARCH	READ	WRITE	SEARCH

The three **classes of users** are:

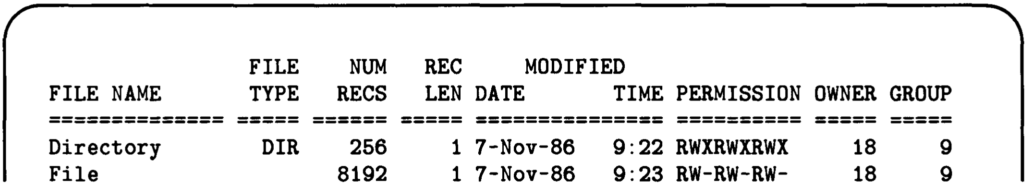
- OWNER—initially the person who created the file; however, ownership of individual files and directories can be changed with the CHOWN[1] statement. With BASIC, the system "owns" all files and directories with an owner identifier of 18.
- GROUP—initially the "group" to which the file's/directory's "owner" belongs; however, the group identifier of individual files and directories can be changed with the CHGRP[1] statement. With BASIC, the system is in the "group" with an identifier of 9, which is also the default group identifier used by the Workstation Pascal system.
- OTHER—all other users who are not the owner and are not in the same group as the owner—that is, everyone else. (This is known as "public" on the HP-UX system.)

[1] CHOWN and CHGRP are used *only* when you will also be using a disc with the HP-UX system. They give selected HP-UX users ownership or group access to files and directories. See the *BASIC Language Reference* entries for CHOWN and CHGRP for further information.

Each class of users has three **types of permissions** for accessing an HFS file or directory:

- READ—allows reading a file or directory (such as with ASSIGN, ENTER, and GET).
- WRITE—allows a user to modify a file's or directory's contents (such as with OUTPUT, RE-STORE, or CREATE).
- SEARCH—an operation on directories which allows you to "search" the directory (such as with CAT and MASS STORAGE IS). This permission has no meaning for files.

The current state of these bits are shown in the `PERMISSION` column of a CAT listing of the directory in which the file or directory resides (`R` for READ; `W` for WRITE; `X` for SEARCH; - for "no permission"):



```
                  FILE    NUM    REC      MODIFIED
FILE NAME         TYPE   RECS    LEN DATE          TIME PERMISSION OWNER GROUP
============== ===== ====== ===== =============== ========== ===== =====
Directory          DIR    256      1 7-Nov-86      9:22 RWXRWXRWX     18     9
File                     8192      1 7-Nov-86      9:23 RW-RW-RW-     18     9
```

The default permission bits for directories are: `RWXRWXRWX`.
The default permission bits for files are: `RW-RW-RW-`.

When a user class is specified, all permission bits for that class are changed:

- If a permission is *specified*, then the corresponding permission bit is *set*;
- If a permission is *omitted*, the corresponding permission bit is *cleared.*

For instance, the following example sets READ and WRITE permission for OWNER, but removes permission for SEARCH:

```
PERMIT "File"; OWNER:READ,WRITE
```

Before **After**

```
---------          RW-------
```

With these permission bits set, the owner of the file can read and write the file (with GET and RE-STORE, for example), but all other users on the system cannot access the file.

As another example, the following example sets READ and WRITE permission for OWNER, but removes permission for SEARCH (note that the PERMIT parameters are the same as in the preceding example, but the "before" permission bits are different):

```
PERMIT "File"; OWNER:READ,WRITE
```

Before	After
`R-XRW-RW-`	`RW-RW-RW-`

With these permission bits set, all classes of users can read and write the file.

If a user class is not specified (OWNER, GROUP, or OTHER), the corresponding access-permission bits are not affected. For instance, the following statement sets the permission bits for OWNER and OTHER but leaves the bits for GROUP unchanged:

```
PERMIT "File"; OWNER:READ,WRITE; OTHER:READ
```

Before	After
`R--R---W-`	`RW-R--R--`

The next example changes bits for GROUP and OTHER but leaves the bits for OWNER unchanged:

```
PERMIT "File"; GROUP:READ; OTHER:READ
```

Before	After
`RW-RW-RW-`	`RW-R--R--`

If no user class is specified, the default permissions for all groups are restored:

```
PERMIT "File"
```

Before	After
`RW-R--R--`	`RW-RW-RW-`

```
PERMIT "Directory"
```

Before	After
`RW-R--R--`	`RWXRWXRWX`

SRM Passwords and Locks

The SRM system offers three kinds of access capability for files and directories:

READ For a file, possessing this access capability allows you to execute statements that read the file (such as GET, ASSIGN, ENTER, etc.).

For a directory, possessing this access capability allows you to execute statements that read the file names in the directory, and to "pass through" the directory when the directory's name is included in a directory path. For example, in the SRM file specifier:

```
"/PROJECTS/Project_one<READpass>/JOHN/f1"
```

including the assigned password `<READpass>` allows passage through the directory `Project_one` to allow access to its subordinate directories and files.

WRITE For a file, possessing this access capability permits you to execute statements that write to the file (such as SAVE, OUTPUT, etc.).

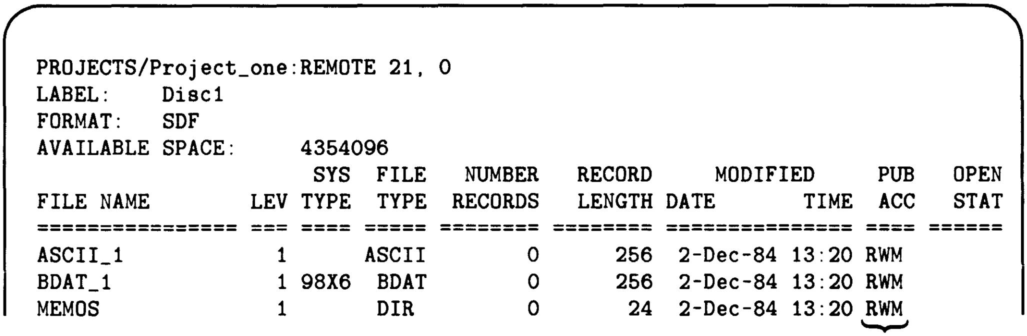
For a directory, possessing this access capability allows you to execute statements that add to or delete from the directory's contents (such as CREATE ASCII, CREATE DIR, PURGE, etc.).

MANAGER With the MANAGER access capability, public capabilities for a file or directory differ slightly from password-protected capabilities.

- Public MANAGER capability allows any SRM user to PROTECT, PURGE, or RENAME the file.
- The password-protected MANAGER capability provides MANAGER, READ, and WRITE access capabilities to users who include a valid password in the file or directory specifier.

Capabilities are either public (available to all workstations on the SRM) or protected (available only to users who know the appropriate password).

The current access capabilities for a file are shown in a catalog listing:



```
PROJECTS/Project_one:REMOTE 21, 0
LABEL:    Disc1
FORMAT:   SDF
AVAILABLE SPACE:      4354096
                      SYS  FILE   NUMBER   RECORD     MODIFIED      PUB   OPEN
FILE NAME         LEV TYPE  TYPE  RECORDS   LENGTH DATE        TIME  ACC   STAT
================ === ==== ===== ======== ======== ============== ==== ======
ASCII_1            1      ASCII        0      256  2-Dec-84 13:20 RWM
BDAT_1             1 98X6  BDAT        0      256  2-Dec-84 13:20 RWM
MEMOS              1        DIR        0       24  2-Dec-84 13:20 RWM
```

Access capabilities
currently public
(not password protected)

In the above example, the file ASCII_1 has none of the access capabilities public; that is, all access must include the password. The file BDAT_1 has the READ capability public, which means that anyone on the SRM system can read the file. The directory MEMOS has READ and WRITE capabilities still open to the public; anyone can create and purge files in the directory, as well as "search" through the directory (with statements like MASS STORAGE IS "MEMOS/SUB_DIR").

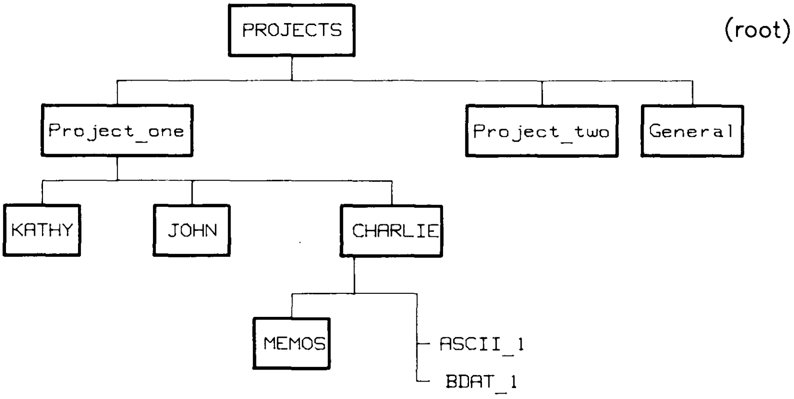
Capabilities are protected with the PROTECT statement, which associates password(s) with one or more access capabilities. One password can be used to protect one or more capabilities. Each file or directory can have several password/capability pairs assigned to it.

Once assigned, the password protecting an access capability must be included with the file or directory specifier to execute statements requiring that access. If you don't specify the correct password when it is required, the system will report an error and deny access to the file or directory.

When you create directories and files, their access capabilities are "public" (available to any user on the SRM). You may subsequently protect a directory or file against certain types of access by other SRM workstations, provided *all* of the following are true:

- You possess MANAGER access capability on the file or directory (MANAGER access to the file is public or you know the password protecting the capability).
- You possess READ access capability on the directory immediately superior to the file or directory you wish to protect.
- You protect the file or directory either while "in" its superior directory or by specifying the valid directory path to its superior directory.

For example, using the directory structure established for other examples in this section (see illustration) and assuming no passwords have been assigned to the files, you could:



1. Assign the password **passme** to protect the MANAGER and WRITE access capabilities on the directory **CHARLIE** with the sequence:

```
MSI "/PROJECTS/Project_one"

PROTECT "CHARLIE",("passme":MANAGER,WRITE)
```

   which executes the PROTECT statement after moving to the directory **Project_one** (immediately superior to **CHARLIE**). As a result of this PROTECT statement, the READ access capability on **CHARLIE** is still public, but any operations that require MANAGER or WRITE capabilities must include the password.

2. Remove all public access capabilities from the file ASCII_1 by assigning the password no_pub, using:

   ```
   PROTECT "CHARLIE/ASCII_1",("no_pub":MANAGER,WRITE,READ)
   ```

   or

   ```
   MSI "CHARLIE"
   PROTECT "ASCII_1",("no_pub":MANAGER,WRITE,READ)
   ```

   These statements assume you are in the directory, Project_one, as if you had executed the statements in the previous step. The second sequence of statements makes CHARLIE the new working directory, whereas in the first, you merely "pass through" CHARLIE to reach ASCII_1. With the READ access capability on CHARLIE still public, you do not need a password.

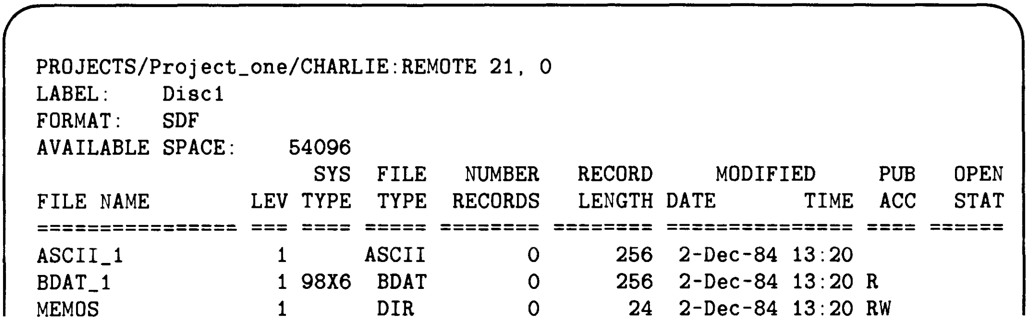
3. Protect the file, BDAT_1, so that data can be read from it but not written into it without using the password, write. If the current working directory were CHARLIE, you would type:

   ```
   PROTECT "BDAT_1",("write":MANAGER,WRITE)
   ```

4. Protect the MANAGER access capability of the directory MEMOS with the password, mgr_pass (so that everyone can read from and write to the directory, but a password is required to purge the directory or its contents) by typing:

   ```
   PROTECT "MEMOS",("mgr_pass":MANAGER)
   ```

If you protected the files and directory in CHARLIE as in the steps above, a catalog listing of CHARLIE would look something like this:



```
PROJECTS/Project_one/CHARLIE:REMOTE 21, 0
LABEL:    Disc1
FORMAT:   SDF
AVAILABLE SPACE:    54096
                      SYS  FILE   NUMBER   RECORD     MODIFIED     PUB   OPEN
FILE NAME        LEV TYPE  TYPE  RECORDS   LENGTH DATE       TIME  ACC   STAT
================ === ==== ===== ======== ======== =============== ==== ======
ASCII_1            1      ASCII        0      256  2-Dec-84 13:20
BDAT_1             1 98X6  BDAT        0      256  2-Dec-84 13:20 R
MEMOS              1       DIR         0       24  2-Dec-84 13:20 RW
```

The letters in the column labeled PUB ACC indicate access capabilities that are public (not protected with a password). For example, only the MANAGER (M) access capability on the directory MEMOS has been protected, leaving the READ (R) and WRITE (W) capabilities available to any SRM workstation user.

Specifying Passwords

When a password is required, you must include the correct password as part of the file or directory specifier in any command or statement that requires the protected access on the file or directory. The password must be enclosed between "<" and ">" and must immediately follow the name of the file or directory it protects.

For example, to get the file ASCII_1, you might execute:

```
GET "/PROJECTS/Project_one/CHARLIE/ASCII_1<no_pub>"
```

If the password were not included in the specifier, the system would respond with an error message and refuse to get the file.

Exclusive Access: Locking SRM Files

Although sharing SRM files saves disc space, allowing several users access to one copy of a file introduces the danger of users trying to access the file at the same time, which can cause unpredictable results. For instance, if one user tries to read part of a file while another user is writing to it, the file's contents may be inaccurate for the read.

To avoid problems, the SRM system adds two BASIC keywords, LOCK and UNLOCK, which you can use to secure files during critical operations. LOCK establishes exclusive access to a file, which means that the file can only be accessed from the workstation at which the LOCK was executed. You may wish to LOCK a file, for example, during any procedure that writes new information to the file.

To permit shared access to the file once again, UNLOCK must be executed from the same workstation, or the file must be closed. Only ASCII or BDAT files that have been opened by a user via ASSIGN may be locked explicitly by that user.

Locking and unlocking is usually done from within a program. For more information, refer to the descriptions of the ASSIGN, LOCK and UNLOCK keywords in the *BASIC Language Reference.*

Locking and Unlocking SRM Files

You can "lock" an SRM file with the LOCK statement, giving you sole access to that file. The same file can be locked several times in succession. Unlocking a file requires that you cancel all locks on that file. If you use the UNLOCK statement, you must cancel each LOCK with a corresponding UNLOCK. Using ASSIGN to re-open a locked file unlocks the file and you must execute another LOCK statement to lock the file again. Closing the file via ASSIGN @...TO * cancels all locks on the file.

In this example, a critical operation must be performed on the file named `File_a`, and you do not want other users accessing the file during that operation. The program might be as follows:

```
1000   ASSIGN @File TO "File_a:REMOTE"
1010   LOCK @File;CONDITIONAL Result_code
1020   IF Result_code THEN GOTO 1010  ! Try again
1030   !  Begin critical process
        .
        .
        .
2000   !  End critical process
2010   UNLOCK @File
```

The numeric variable called `Result_code` is used to determine the result of the LOCK operation. If the LOCK operation is successful, the variable contains 0. If the LOCK is not successful, the variable contains the numeric error code generated by attempting to lock the file.

Copying Files and Volumes

The COPY statement allows you to duplicate individual files or an entire disc volume. Any type of file may be copied; however, only LIF volumes may be copied as an entire volume.

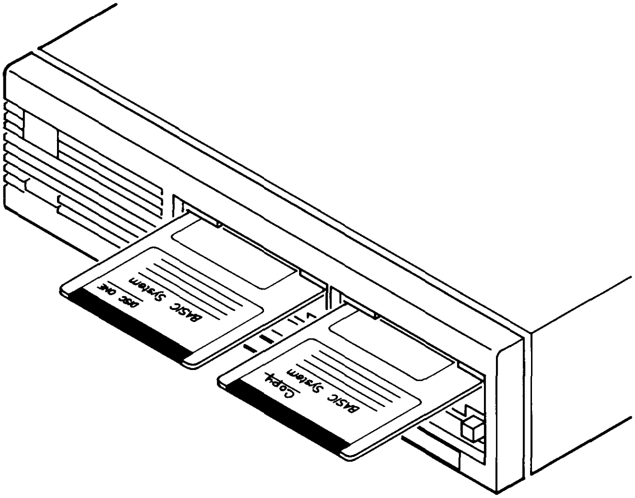
- **Copying a file** duplicates the existing file and places the new file name in the directory.

  ```
  COPY "ExistFile" TO "NewFile"
  ```

  A new file can be created either on the same volume or on another volume. If you copy a file to the same volume, the new file name must be different from the existing file name. If there is not enough room on the disc for the file to be copied, the system cancels the statement and reports an error.

- **Copying an entire volume** makes an *image* copy of the source volume on the destination volume. This type of copy is allowed only with LIF volumes.

  Note: This type of copy **destroys all** data the destination disc. If you want to copy multiple files, you should use one of the "back-up" utilities described in the "BASIC Utilities" chapter of this manual.



To perform this type of copy, simply identify the source and destination volumes.

```
COPY ":,700" TO ":,702"
```

Note: You can copy a larger volume to a smaller volume, if copying the files from the larger volume will not overflow the smaller one. However, copying a smaller volume to a larger volume will effectively change the size of the larger volume (making it the size of the smaller volume).

More Examples

The following statement copies "File1" from the current default mass volume to a new file called "File2" on the same volume:

```
COPY "File1" TO "File2"
```

The following statement copies the file named "MYPROG" from the current default volume into a file with the same name on the specified volume at interface select code 7, primary address 2, unit number 1:

```
   COPY "MYPROG" TO "MYPROG:,702,1"
```

The following statement copies "File1" from the current system mass storage to a drive at interface select code 7, primary address 0, unit number 1. Note that both files can be named "File1" if they are on *different* volumes.

```
  COPY "File1" TO "File1:,700,1"
```

As you may have guessed, COPY can be used to copy files from volumes of one format to another. For instance, you can copy LIF files to HFS or SRM volumes, SRM files to HFS or LIF volumes, and so forth.

Copying LIF Files

To copy a protected LIF file, you must specify the protect code: The following statement copies a file from a CS80 drive to the current system mass storage device. The old file "File1" has the protect code "xx", but the new file has no protect code:

```
  COPY "File1<xx>:,700,0" TO "DATA"
```

The following statement copies the entire disc from the right-hand drive to the left-hand drive of an dual-drive disc.

```
  COPY ":,700,1" TO ":,700,0"
```

You can copy an entire LIF volume in a single COPY statement. Make **absolutely sure** that you understand the consequences of this type of operation (see the preceding section for details).

Copying HFS Files

To copy an HFS file, you must have R (read) permission on the existing file and X (search) permission on all superior directories. You also need W (write) and X permissions on the destination directory into which the duplicate file is being copied. as well as X permission on all superior directories.

You cannot copy directories, although you can copy files from one directory to another. Similarly, you cannot copy an entire HFS volume in a single COPY statement. (You must copy an HFS volume file by file.)

Copying SRM Files

To copy an HFS file, you must have R (read) permission on the existing file. You must also have W (write) permission on the directory into which the duplicate file is being copied, as well as R permission on all superior directories.

You cannot copy directories, although you can copy files from one directory to another. Similarly, you cannot copy an entire SRM volume in a single COPY statement. (You must copy an SRM volume file by file.)

Renaming Files

The name of a file can be changed without disturbing the file's contents. This is done with the RENAME statement. For example, to change the name of a file (on the default volume) from "George" to "Frank", use the following statement:

```
RENAME "George" TO "Frank"
```

Special Additional Action with SRM Volumes

The RENAME statement can be used to change the name of a file, optionally changing its location in the hierarchy:

```
RENAME "/USERS/MARK/File22" TO "/USERS/MARK/OLD_FILES/File22"
```

Purging Files

You can purge a file from a directory by using the PURGE statement. Purging a file deletes the directory entry for the file and releases the reserved space in the data area.

A file entry can be removed from the disc directory with·the PURGE statement. This prevents any further access to the file. For example, the following statement removes the file "Old_stuff" from the current default volume:

```
PURGE "Old_stuff"
```

Once a file is purged, there is generally no way of retrieving the information it contains[1].

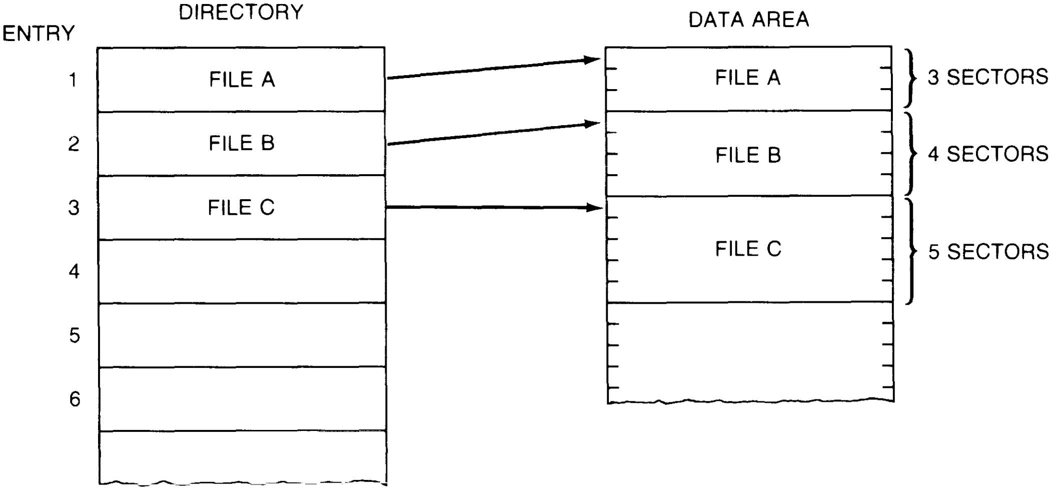
[1] The only exception to this rule is with LIF discs, on which files can be "up-purged" using the Mass Storage Program (MASS_STOR). See the "BASIC Utilities" chapter for instructions.

Effects of PURGE on LIF Directories

The order of file names and files' data areas are the same on LIF discs. Therefore, purging a file on a LIF directory creates two "gaps" on the disc:

- One in the data area
- One in the directory.

As an example, suppose that you have three consecutive files on a disc with the following names and sizes.

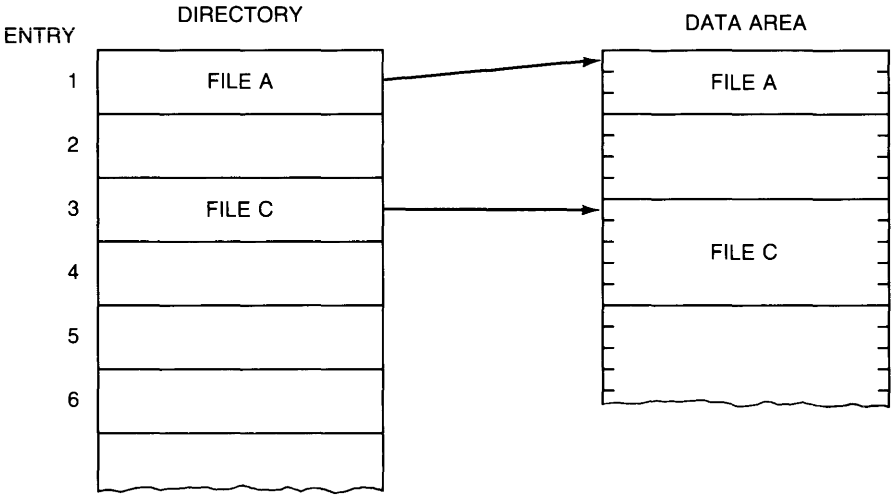


LIF directory entries are in the same order as the files in the data area. The third directory entry, for example, must correspond to the third file in the data area. Consequently, if you PURGE a LIF file, and then create a smaller file, you may lose disc space. The following examples illustrate this principle.

Executing the following statement:

```
PURGE "FileB"
```

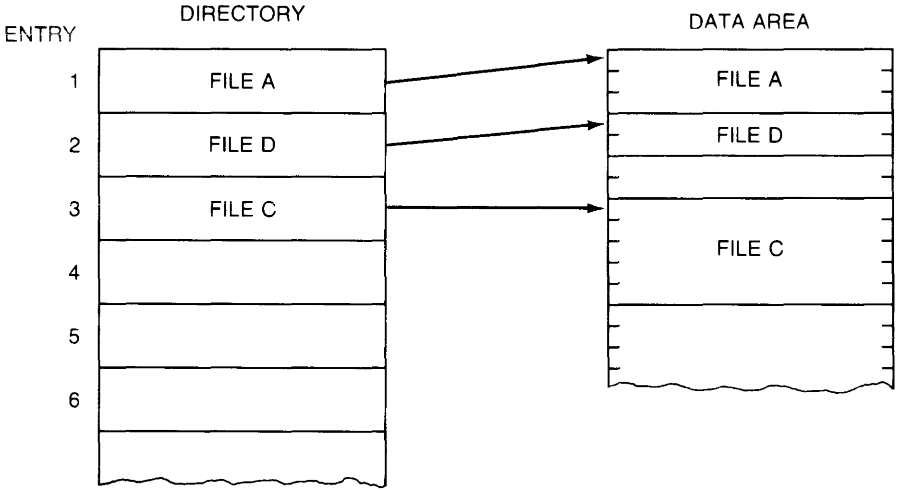
creates a 1-entry gap in the directory and a 4-sector gap in the data area.



When you create a file on a LIF volume, the system looks for the first gap in the data area with enough room to store the file. When it finds one, it puts the file into this gap. To continue the above example, suppose you create a 2-sector file with this statement:

```
CREATE ASCII "FileD",2
```

The system will place this file in the data-area gap and place the directory entry in the directory gap.



You now have a 2-sector gap in the data area but no gaps in the directory. If you create another file, the system will fill entry 4 in the directory and will reserve space in the data area past `FileC`. The two unused sectors will not be reclaimed unless you PURGE one of the adjacent files, `FileD` or `FileC`.

In this example, the 2 unused sectors were not a large problem. However, suppose that `FileB` was 800 sectors long, and `FileD` was 2 sectors. This scenario would result in 798 sectors on the disc being unusable!

The solution to this problem is to "REPACK" the disc using the "Mass Storage" (MASS_STOR) utility, available on the *BASIC Utilities* disc. See the "BASIC Utilities" chapter for instructions.

Purging HFS Files and Directories

The PURGE statement works the same for removing HFS files as for removing files from other mass storage. You may also remove directories using PURGE.

Here are the restrictions on using PURGE to remove HFS files:

- In order to use PURGE, you must have ownership of the file or directory. (See the preceding "HFS File Access Permissions" section for details.)
- PURGE works only with closed files and directories. (You cannot PURGE a file currently open with ASSIGN, or a directory which is the current working directory—specified in the last MASS STORAGE IS statement.)
- Directories must also be empty (not contain any files or directories).

Purging SRM Files and Directories

The PURGE statement works the same for removing remote files as for removing files from local mass storage. You may also remove directories using PURGE. PURGE works only with closed files and directories. Directories must also be empty (not contain any files or directories).

When specifying the SRM file to be purged, you must include all passwords protecting access capabilities required for the PURGE. For example, to purge the file `BDAT_1` from the directory `CHARLIE` (see previous examples), you could type:

```
PURGE ".<passme>/BDAT_1<write>"
```

In this example, `CHARLIE` is the current working directory, as denoted in the directory path by " . ". (Refer to the syntax for directory path in this chapter's "BASIC Language Reference" section.)

To purge a file, you must have the MANAGER access capability on that file and READ and WRITE access capabilities on the file's superior directory. Because passme protects the WRITE capability on CHARLIE and write protects the MANAGER capability on BDAT_1, both passwords must be included in the file specifier in the PURGE statement.

Although you do not normally need to specify the working directory in a directory path, you must include the password for the PURGE operation. The READ capability on CHARLIE is not password-protected.

To purge CHARLIE, you would first need to purge the remaining files and directory in CHARLIE. Because the MSI statement "opens" a directory (making it the current working directory), you must also "close" CHARLIE.

For example, if no files or directories remained in CHARLIE, you could purge CHARLIE by executing these two commands:

```
MSI ":REMOTE"
PURGE "PROJECTS/Project_one/CHARLIE<passme>
```

The first statement closes CHARLIE and establishes the root directory as the current working directory. Note that, because passme protects the MANAGER access capability on CHARLIE, you must include that password in the PURGE statement.

Volume Labels (LIF and HFS Volumes Only)

When you INITIALIZE a LIF disc, the BASIC system gives it a default “volume label”—”B9836” or something similar—that is displayed with the CAT statement:

```
         :CS80,700
 ----->  VOLUME LABEL: B9836
         FILE NAME PRO TYPE  REC/FILE BYTE/REC   ADDRESS

         MyProg        PROG        14      256        16
         VisiComp      ASCII       29      256        30
```

Volume labels are useful in cases when you want to identify a particular *disc media*, rather than a particular *disc drive*, for instance. (The example below shows this use.)

Reading Volume Labels

To determine the label on the media currently installed in a mass storage device, use the device's mass storage unit specifier:

```
10  READ LABEL Label$ FROM ":,700,1"
```

The statement puts the label into the string variable named Label$, which must be large enough to hold all of the characters in the label (volume labels are 6 characters or less).

Here is an example that searches two drives for a disc with the volume label “MY_VOL”:

```
     .
     .
     .
100  READ LABEL Vol_label$ FROM ":,700" !       Unit 0.
110  IF Vol_label$="MY_VOL" THEN Msus$=":,700"
120  !
130  READ LABEL Vol_label$ FROM ":,700,1" !     Unit 1.
140  IF Vol_label$="MY_VOL" THEN Msus$=":,700,1"
150  !
     .
     .
     .
```

Writing Volume Labels

To write the label "MY_VOL" onto the current default volume, you could execute either of these statements:

```
PRINT LABEL "MY_VOL" TO SYSTEM$("MSI")
    or
PRINT LABEL "MY_VOL"
```

To change the volume label on a different volume, you will need to specify the volume:

```
PRINT LABEL "MY_VOL" TO ":,700,1
```

Enabling Checkread Verification

Normally, the BASIC system writes data into files and does not verify that the data was written without error. The reliability of mass storage devices is generally high enough to justify this design decision. However, if you want the system to perform a read-after-write verification of data written into files, you can use this statement:

```
CHECKREAD ON
```

Data subsequently written into all files by the following statements is subject to this verification:

```
COPY                PRINT LABEL             RE-SAVE
CREATE ASCII        PROTECT                 STORE
CREATE BDAT         PURGE                   RE-STORE
OUTPUT              RENAME                  TRANSFER
                    SAVE
```

Note, however, that CHECKREAD does not affect files being written through the PRINTER IS or PLOTTER IS statements. For SRM systems, there is already a checkread operation performed automatically by the SRM controller; therefore, using CHECKREAD statement does nothing.

To disable this feature, execute:

```
CHECKREAD OFF
```

What to Do Next

Task/Topic	Chapter Number
Learn how to load and run programs.	9
Learn about how to enter, edit, secure, document, and store programs.	11
Learn about how to maintain your system.	12 thru 17
Learn how to use a BASIC utility program.	18
Learn about each key on your keyboard.	19 thru 21

Notes

Editing and Storing Programs 11

In this chapter you will learn how to enter, edit, and store BASIC programs.

A Brief Look at Entering and Storing Programs

One of the great joys of using this BASIC system is that it makes entering, editing, and storing programs an extremely simple task. This section introduces you to some fundamental concepts and skills involved in entering and storing BASIC programs.

Terminology

keyword a group of characters recognized by BASIC to represent some pre-defined action. Examples are:

```
CAT
LOAD
COPY
```

statement a keyword followed by any parameters and/or secondary keywords that:

- are required or allowed with that keyword
- and fit on one program line[1]

Examples are:

```
CAT ":,700"
LOAD "MyProg"
COPY "MyProg" TO "BackupFile"
```

command a statement that is executed from the keyboard.

program line a statement preceded by a line number (and optional line label) that is stored in a program. Examples are:

```
100  CAT ":,700"
250  COPY "MyProg" TO "BackupFile"
875 Line_label:  LOAD "MyProg"
```

[1] The maximum length of a command or program line is two CRT lines. When entering commands and programs from the keyboard, this is 160 characters on most models, but only and 100 characters on the Model 226. (Program lines longer than 100 characters can be created on a 310, for instance, and then transferred to a 226 using a mass storage operation such as LOAD or GET. The resulting lines are valid program lines on the 226. However, when viewed on the CRT in EDIT mode, these lines have an asterisk after the line number, with only the first 100 characters visible.)

Statements as Commands vs. Program Lines

In general, a statement can be used as *either*:

- A command—if the statement is executed from the keyboard; for example:

  ```
  PRINT "This is a keyboard command." Return or ENTER
  ```

- A program line—if the statement contains a leading line number; for instance:

  ```
  100  PRINT "This is a program line."
  ```

However, there are some statements that *cannot* be:

- Executed as commands (such as DIM and RETURN)

  ```
  100  DIM String_var$[100]
  ```

- Stored as program lines (such as DEL and SCRATCH)

  ```
  SCRATCH A
  ```

The general case is that statements are both programmable and keyboard executable, such as CALL and PRINT. The *BASIC Language Reference* shows whether or not each keyword is keyboard executable or programmable, or both.

Getting into EDIT Mode

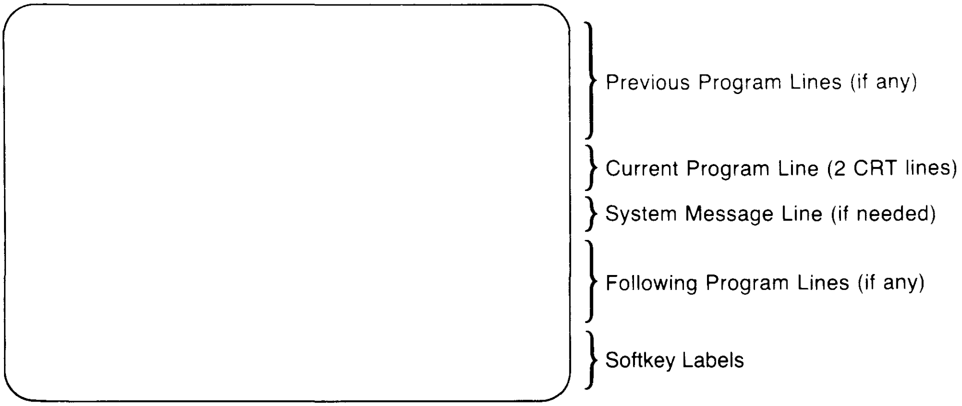
When you want to edit program lines, you will need to get into EDIT mode[1]. If a program is not running[2], you can get into edit mode by typing:

`EDIT` Return or ENTER

or by pressing the EDIT key followed by ENTER (HP 98203 keyboards only).

You can also use the **EDIT** softkey (f1 in the `User 1` menu of an ITF keyboard).

The system goes into EDIT mode, and the screen has this format:


Previous Program Lines (if any)
Current Program Line (2 CRT lines)
System Message Line (if needed)
Following Program Lines (if any)
Softkey Labels

[1] The EDIT binary must be loaded in order to use EDIT mode. It is possible, however, to enter a program by typing program lines (line number and statement) on the normal keyboard input line of the CRT and pressing the Return key. But it is usually much more desirable to use EDIT mode where you can see several program lines at one time.

[2] See the "Introduction to the System" chapter for instructions on determining whether or not a program is running.

In this mode, you can view a multi-line portion of the program. You can view different portions of the program by scrolling the display (see subsequent section called "Scrolling the Program" for details). If you want to edit a particular line, you must scroll the display so that the line you want to edit is in the middle of the screen.

If there is no program in memory when you enter EDIT mode, the cursor will appear on a line with the number 10, which is the default line number of the first program line.

```
10  _
```

You can then begin to enter program lines; the following section explains how to do that.

Note

When you enter EDIT mode, the typing-aid softkey definitions and labels are changed to a User menu, whose definitions that are more useful for editing operations. See the subsequent section called "A Closer Look at Editing" for further information.

Entering Program Lines

To enter a program line, type the desired characters at the keyboard. For practice, type in the lines shown below. The characters you will type are encircled to represent key presses; the 10 and 20 are in a dot-matrix font[1] to indicate that the system supplies the line numbers.

```
10 [P][R][I][N][T][ ]["][T][i][n][y][ ][p][r][o][g][.]["][Return]
20 [E][N][D][Return]
```

[1] Note that this is not the normal notation used in this text; usually, dot-matrix font is used to show characters in program lines and commands that can be typed *exactly* as shown.

Correcting Typing Mistakes

If you make any errors while typing, use the Back space or ◄ and ► keys to move the cursor to the erroneous character(s), and then re-type them correctly.

Storing the Line

Once the line is exactly as you want it, press Return or ENTER. (The cursor may be any place on the line when you store it; the system will read the entire line, regardless of the location of the cursor.)

The BASIC System Checks Syntax

Before storing a program line, the computer checks for syntax[1] errors, and also changes the lettercase of keywords and identifiers (see the following section, "Uppercase or Lowercase Letters?", for details).

Immediate syntax checking is one big advantage of writing programs on this BASIC system. A great many programming errors can be detected at program-entry time, which increases the chances of having a program run properly and cuts down debugging time. If the syntax of the line is correct, the line is stored, and the next line number appears in front of the cursor.

If the system detects an error in the input line, it displays an error message immediately below the line and places the cursor at the location it blames for the error.

```
  10 PRINT "Short program.

Error 949  Syntax error at cursor

  20 END
```

Keep in mind that there is an endless variety of human mistakes that might occur, and that BASIC is not very good at dealing with even slight ambiguities. As a result, you might not always agree with its diagnosis of the exact error or the error's location. However, an error message definitely indicates that something needs to be fixed. There is a complete list of error messages and their meanings in the "Errors" appendix of the *BASIC Language Reference* and *BASIC Condensed Reference* manuals.

[1] *Syntax* is a term used to describe the way in which keywords, parameters, etc. are put together to form a legal statement.

Uppercase or Lowercase Letters?

Program entry is simplified by the computer's ability to recognize the uppercase- and lowercase-letter requirements for most elements in a statement. An entire statement can be typed using all uppercase or all lowercase letters. If the statement's syntax is correct, and there are no "keyword conflicts" (see the explanation below), the system stores the program line. Upon LISTing or EDITing the program[1], however, the system uses these conventions:

- Keywords are *all* uppercase letters (CAT, LOAD, DISP, etc.).
- All variable names are listed with the first letter in uppercase and the rest of the letters, if any, in lowercase (Var1, String33$, etc.).

In other words, you don't usually have to bother with the Shift key when you enter a line, because the system will automatically change all letters to the proper lettercase. On the other hand, if there is a "keyword conflict," an error is reported. A keyword conflict occurs when you try to use a keyword for an identifier (variable name, line label, or subprogram name). You can use keywords for identifiers; simply change the lettercase of at least one letter in the identifier name (for example, `Cat` or `cAt`), and then press Return or ENTER again. A word containing a mixture of uppercase and lowercase letters is assumed to be an identifier.

The system's assumptions about keywords versus identifiers won't cause any problems if your line has the proper syntax. However, if you are guessing at a keyword or syntax, don't assume that you got the line right just because the computer stored it. For instance, suppose that you are trying to type a PRINT statement to print a blank line; however, you misspell the keyword PRINT:

```
100  PRINY
```

The system does not report an error, because the line could legitimately be interpreted as a call to a subprogram named "Priny".

Thus, in general, if the system puts lowercase letters in something that you thought was a keyword, then it wasn't really recognized as a keyword.

[1] The EDIT binary must be loaded in order to edit and list programs.

Keys Used for Editing the Current Line

In order to edit a line, it must be the "current line"—the line that the cursor is on. The next few paragraphs give a quick overview of the standard editing keys that you can use while editing the current line. (The "Keyboard Reference" chapter lists all key definitions for each type of keyboard.)

Editing Feature	Explanation
Normal cursor (blinking underline, _)	Whenever you type characters at the keyboard, they appear on the current line at the cursor, *overwriting* any existing characters.
◄, ►, and Back space	move the cursor one character in the indicated direction. If the cursor has reached either end of the line, it doesn't go any farther. Pressing Shift ► moves the cursor to the end of the line, and Shift ◄ moves the cursor to the beginning of the line.
Knob or mouse	also move the cursor.
INS CHR or Insert char	changes the cursor to the insert cursor (see below), and enters insert mode (any characters typed are placed before the current cursor position, and the cursor and subsequent characters are shifted one position to the right). This key toggles between the normal cursor and the insert cursor.
Insert cursor (inverse-video block, ▒)	indicates that the character entered is inserted *in front of* the character currently highlighted by the cursor.
DEL CHR or Delete char	deletes the character pointed to by the cursor. Subsequent characters on the line are shifted one position to the left.
CLR→END or Clear line	deletes all the characters from the cursor to the end of the current line.
CLR LN or Shift-Clear line	clears the entire current line.

Editing Feature	Explanation
SET TAB and CLR TAB or f5 and Shift-f5 (System menu)	SET TAB and CLR TAB perform the indicated action at the cursor position.
Tab and Shift-Tab	The Tab key moves the cursor to the next tab position, if there is one. Shift-Tab moves the cursor back to the previous tab position, if there is one.
ANY CHAR (SHIFT-STEP) or f7 (System menu)	Characters that don't appear on the keycaps can be typed by using this key. Assume that you are typing a program line and you want the vertical bar character. Press the ANY CHAR key. The following message appears below the current line: `ENTER 3 DIGITS, 000 THRU 255` For instance, the decimal ASCII code for a vertical bar is 124. Press the 1 2 4 number keys. A vertical bar appears at the cursor position, and the message goes away. If a key that is not part of a 3-digit number in the proper range is pressed during this operation, the ANY CHAR operation is aborted and the key performs its normal function. (By the way, the vertical bar character is available on the 98203 keyboard; press SHIFT-(on the numeric keypad.)
Softkeys k0 thru k9 or f1 thru f8 (in the User 1, 2, and 3 menus of ITF keyboards)	These keys produce characters and system-key presses, just as if you had typed them at the keyboard. See the "Using Typing-Aid Softkeys" section of the "Introduction to the BASIC System" chapter for details.

Keys Used for Scrolling the Program

All of EDIT mode's text-entry capabilities apply to the "current line"—the line in the middle of the screen with the cursor on it. That is, you must move a line to the current-line position before you can edit it[1]. The text on the screen is scrolled so that you are always editing the line in a "window" in the middle of the screen.

Editing Key	Explanation
▲	scrolls the program up one line, so that you will be editing the next program line. (Note that the cursor remains on the line in the middle of the screen.)
▼	scrolls the program down one line, so that you will be editing the preceding program line.
Shift-▲	scrolls the program all the way up, so that the end of the program is displayed and the next available line number shown on the current line.
Shift-▼	scrolls the program all the way down, so that the beginning of the program is displayed and the first program line is the current line.
Knob or mouse	scroll the program.

[1] The only exception to this is when you enter a new line with the same line number as an existing line. In that case, the new line replaces the old, even though the old line was not moved to the current-line position.

Inserting Lines

Lines can be easily inserted into a program. As an example, assume that you want to insert some lines between line 90 and line 100 in your program. Place line 100 in the current-line position, and press the Insert line (INS LN) key.

```
90   PRINT "Line 90."
100  PRINT "Line 100."  ←———————Make this the current line,
                                   then press Insert line.
```

The program display "opens" and a new line number appears between line 90 and line 100.

```
90   PRINT "Line 90."
91   _                  ←———————Begin typing; letters appear at cursor.
100  PRINT "Line 100."
```

Type and store the inserted lines in the normal manner. Appropriate line numbers will appear automatically.

The insert mode can be cancelled by pressing the Insert line (INS LN) key again, or by performing an operation that causes a new current line to appear (such as scrolling or insert line).

While inserting lines, the system maintains the established interval between line numbers, if possible.

- If the interval between lines in the preceding example was 5, the first line number appearing would be 95.
- When the normal interval between lines can no longer be maintained, an interval of 1 is used. Thus, after line 95 is stored, the next line number supplied is 96.
- When there are no line numbers available between the current line and the next line, enough of the program below the current line is renumbered to allow the insert operation to continue. In the example, this would happen after line 99 is stored. The original line 100 is renumbered to 101 and the number 100 appears in the current line.

Deleting and Recalling Lines

Lines can be deleted one at a time or in blocks. The Delete line (DEL LN) key deletes the current line, after which the lower part of the display is refreshed to "close" the space created by the deletion.

Before deletion:

```
90   PRINT "Line 90."
100  PRINT "Line 100." ←———————— Make this the current line,
110  PRINT "Line 110."             then press Delete line.
```

After deletion:

```
90   PRINT "Line 90."
110  PRINT "Line 110." ←———————— New "current line."
```

If you press the Delete line by mistake, you can recover the line by pressing RECALL (f8 in the System menu, or left-most unlabeled key above the numeric keypad, on an ITF keyboard), and then store it by pressing Return or ENTER. The system has a recall buffer that holds the last lines entered, deleted, or executed. You can cycle through these lines, most recent to less recent, by repeatedly pressing the RECALL key. SHIFT-RECALL cycles through from the current line to the more recent lines. (You can also clear this recall buffer by executing SCRATCH R; this is useful, for instance, when you want to keep others from seeing passwords that you may have typed at the keyboard while accessing protected files.)

When the keyword DEL is followed by a single line identifier, only a single line is deleted. The line identifier can be a line number or a line label. The Delete line key produces the same results, but has some advantages.

- You can see the line before you delete it.
- The Delete line key saves the line in the recall buffer (the DEL command does not).

Therefore, DEL is more useful for deleting blocks of lines (described in the subsequent section called "Deleting Multiple Lines.")

Copying Lines (By Changing Line Numbers)

Although the computer supplies a line number automatically, you are not forced to use that number if you don't want to. To change the line number, simply back up the cursor and type in the line number you want to use. You can do this to existing lines as a way of copying them to another part of the program. (Note that there is an easier way to copy program lines—by using the COPYLINES command—as described in the next section.)

When you change a line number and store the line, the program is automatically scrolled so that the line just stored is one line above the current-line position. In other words, when you copy a line to a new location, the new location is displayed.

Here are some points to keep in mind when changing the line numbers supplied by the system.

- Changing the line number of an existing line causes a copy operation, not a move. The line still exists in its original location.
- Existing lines are replaced by any line entered with their same line number.
- Be careful that you don't accidently replace a line because of a typing mistake in the line number.

Getting Out of EDIT Mode

There are many ways to terminate the EDIT mode. Your choice depends upon what yôu want to do next. If you simply want to return the CRT to its "normal" mode (input line on the bottom and printout area above), press [PAUSE] ([Stop]) or [CLR SCR] ([Clear display]). Any of these keys terminates EDIT mode and returns the screen to the normal format.

Another way to leave EDIT mode is to proceed with another operation. The most useful choices in this case are LIST, CAT, [RESET], [RUN] ([f3] in the System or a User menu of ITF keyboards), or [STEP] ([f1] in the System menu of ITF keyboards). EDIT mode is also terminated by a GET or LOAD operation, and by any operation that uses the display (like LIST and CAT).

Listing the Program

List the program by executing the following command[1] (which also gets you out of EDIT mode, if you have not already terminated it in one of the other ways described in the preceding section):

```
LIST  [Return] or [ENTER]
```

The system lists the program on the screen (or whichever device is the current PRINTER IS device).

```
10  DISP "Short program."
20  END
```

Storing the Program

You can store the above program, for instance, in the file named "MyProg" on the default volume by executing this statement:

```
STORE "MyProg"  [Return] or [ENTER]
```

You can also use the SAVE statement, which stores the line in an ASCII representation (rather than in an "internal" representation; see "A Closer Look at Storing Programs" for details.)

Running the Program

Now run the program by typing:

```
RUN  [Return] or [ENTER]
```

or pressing the [RUN] key ([f3] in the System or in a User menu of ITF keyboards).

The computer should display the following message on the CRT display (or current system display device):

```
Short program.
```

Further information about running programs in described in the "Loading and Running Programs" chapter.

The next sections describe additional editor and system features.

[1] The EDIT binary must be loaded in order to use LIST and EDIT.

A Closer Look at Editing

This section provides a closer look at the BASIC editor[1], showing you more of its powerful and easy-to-use features.

More Details about Getting into EDIT Mode

The EDIT command allows two parameters. The first is a line identifier and the second is the increment between line numbers.

For example, the following command tells the computer to place the program on the CRT so that line 140 is in the current-line position.

```
EDIT 140,20
```

Also, any lines that are added to the program get a line number 20 greater than the previous line.

If the increment parameter is not specified, the computer assumes a value of 10. Thus, the following command tells the computer to place the program on the CRT so that line 1000 is in the current-line position, and added lines get a line number 10 greater than the previous line.

```
EDIT 1000
```

When the line identifier is not supplied, the computer has some interesting ways of assuming a line number.

- If this is the first EDIT after a power-up, SCRATCH, SCRATCH A, or LOAD, the assumed line number is 10.
- If EDIT is performed immediately after a program has paused because of an error, the number of the line that generated the error is assumed.
- At any other time, EDIT assumes the number of the line that was being edited the last time you were in EDIT mode.

[1] The EDIT binary must be loaded in order to use the BASIC editor.

The line identifier also can be a line label. This makes it very easy to find a specific program segment without needing to remember its line number. For example, assume that you want to edit a sorting routine that begins with a line labeled `Go_sort`. Simply type:

```
EDIT GO_SORT
```

The line labeled `Go_sort` is placed in the middle of the display, and lines before and after this line (if any) are displayed above and below this line, respectively.

The EDIT command is not programmable, and you cannot use EDIT mode while a program is running.

In order to locate a program line in a subprogram context, you can use the FIND command. See the subsequent section called "Global Editing Operations" for details.

Typing-Aid Softkey Menu Changes (ITF Keyboards Only)

When you go from "normal" mode to EDIT mode on a system with an ITF keyboard, the softkey menu and labels change to the User 2 menu (if the PDEV binary is currently loaded).

While in EDIT mode on an ITF keyboard, however, you can switch softkey menus normally: use either the Menu key, or the appropriate statements (such as SYSTEM KEYS and USER 1 KEYS) to switch to other menus.

If you are in the User 2 menu when you exit EDIT mode, the system will return you to the menu that was in effect when you entered EDIT mode.

A Closer Look at Listing a Program

All or part of your program can be displayed or printed by executing a LIST statement[1]. The LIST statement allows parameters that specify both the range of lines to be listed and the device to which the listing should be sent.

If the keyword LIST is executed without any parameters, the assumed action is to list the entire program on the system printer.

```
LIST
```

The default system printer after a power-on or SCRATCH A is the CRT. (The system is defined by the PRINTER IS statement.)

Starting and ending line numbers can be specified in the LIST statement. For example, the following command lists lines 100 thru 200, inclusively.

```
LIST 100,200
```

The following example lists the last portion of the program, from line 1850 to the end.

```
LIST 1850
```

The line identifiers can also be labels. For instance, the following command lists the program from the line labeled "Rocket" to the end.

```
LIST Rocket
```

[1] The EDIT binary must be loaded in order to use the LIST statement and not get the message **(Requires EDIT binary)**.

Directing the listing to a device other than the CRT is easy, but involves concepts that have not been introduced yet. If you want a listing on a printer, you have two choices:

- Specify a different system printer, and then use the LIST statement. For example:

  ```
  PRINTER IS 701
  LIST
  ```

  The parameter 701 identifies the printer connected to the computer through the interface at select code 7 (the built-in HP-IB); the printer itself has an address setting of 01. You can also use the PRT function, which returns a value of 701.

  However, it is often desirable to keep the CRT display as the system printer and still get program listings on an external printer.

- Specifying the printer in the LIST statement. For example, the following command sends the entire program listing to an HP-IB printer (address 01) without changing the system printer selection.

  ```
  LIST #701
  ```

  When both the printer and the line range are specified, the printer number is specified first and terminated with a semicolon. For example, this command lists lines 200 thru 500 on the device connected to the interface at select code 12.

  ```
  LIST #12; 200,500
  ```

Global Editing Operations

The preceding sections showed how to edit single program lines. This section shows how to perform editing operations that may affect the entire program.

BASIC Command	Purpose and Example Command	Typing-Aid Softkey[1]
REN	renumbers the program (or a specified segment of the program) `REN 100,10`	f1 (k1)
INDENT	indents lines in a program to show the nesting of the branching constructs (such as FOR..NEXT and REPEAT..UNTIL). `INDENT 7,2`	f8 (k2)
FIND	searches the program for a specific textual pattern `FIND "a pattern"`	f6 (k3)
CHANGE	searches for a textual pattern, but allows you to optionally change it to a new pattern `CHANGE "old text" TO "NEW CHARACTERS"`	f7 (k4)
COPYLINES	copies (duplicate) program line(s) to another location in a program. `COPYLINES 10,300 TO 550`	f5
MOVELINES	moves program line(s) to another location in a program. `MOVELINES 450,522 TO 10`	f4
DEL	deletes program segments (ranges of program lines) `DEL 100,150`	Not on a default typing-aid softkey.

[1] *Note that the ITF softkeys (f1, f2, etc.) are in the User 2 menu.*

This section explains how to use these commands.

Renumbering a Program

After an editing session with many deletes and inserts, the appearance of your program can be improved by renumbering. This also helps make room for long inserts. Renumber programs with the REN command.

This example renumbers the entire program in memory, using a new beginning number of 10 and incremental line numbers of 10:

```
REN
```

Both the starting line number and the interval between lines can be specified. For example, the following example renumbers the entire program, using 100 for the first line number and an increment of 5.

```
REN 100,5
```

If the increment (second parameter) is not specified, 10 is assumed. For example, the command below renumbers the entire program, using 1000 for the first line number and an increment of 10.

```
REN 1000
```

As shown in the first example above, a value of 10 is assumed for starting-line number and line-number increment when no parameters are specified.

You can also renumber only a specified portion of a program. For example, the following command renumbers only line numbers in the range 1000 to 2000:

```
REN  1000,10 IN 1000,2000
```

Indenting a Program

INDENT is also a non-programmable command. It is used to scan an entire program and indent it so as to show the "nesting" of program segments[1] that define:

- Looping (such as FOR..NEXT and REPEAT..UNTIL)
- Conditional execution (such as IF..THEN and SELECT..CASE..END CASE)
- A separate program segment (such as SUB subprograms and DEF FN user-defined functions)

[1] A complete list of the statements that define these constructs is provided in the *BASIC Language Reference* description of the INDENT command.

The following program shows the indentation performed by this command:

```
INDENT 7,2
```

the first parameter (7) indicates the indentation of the "outermost" program segment, and the second parameter (2) shows how many additional spaces each subsequently nested segment is indented. Notice how easy it is to follow the logic flow?

```
10    FOR I=1 TO 5
20      REPEAT
30        INPUT "How old are you?",Age
40        Reasonable=1  ! Assume they're telling the truth...
50        IF Age<0 THEN
60          DISP "Aw, c'mon! You gotta be born.
70          Reasonable=0
80        ELSE
90          IF Age>120 THEN
100           DISP "Oh, p'shaw!  Are you sure?!"
110           Reasonable=0
120         ELSE
130           IF Age>100 THEN
140             DISP "You are pretty spry!"
150           ELSE
160             IF Age>80 THEN
170               DISP "Wow! Most people your age don't use computers much."
180             ELSE
190               DISP "Glad to meet you."
200             END IF
210           END IF
220         END IF
230       END IF
240       WAIT 4
250     UNTIL Reasonable
260     DISP "You were";Age*365.2422;" days old on your last birthday."
270     WAIT 3
280   NEXT I
290   END
```

Here is another example of indenting the same program, but with different parameters:

```
   INDENT 5,4

10  FOR I=1 TO 5
20      REPEAT
30          INPUT "How old are you?",Age
40          Reasonable=1  ! Assume they're telling the truth...
50          IF Age<0 THEN
60              DISP "Aw, c'mon! You gotta be born.
70              Reasonable=0
80          ELSE
90              IF Age>120 THEN
100                 DISP "Oh, p'shaw!  Are you sure?!"
110                 Reasonable=0
120             ELSE
130                 IF Age>100 THEN
140                     DISP "You are pretty spry!"
150                 ELSE
160                     IF Age>80 THEN
170                         DISP "Wow! Most people your age don't use computers
much."
180                     ELSE
190                         DISP "Glad to meet you."
200                     END IF
210                 END IF
220             END IF
230         END IF
240         WAIT 4
250     UNTIL Reasonable
260     DISP "You were";Age*365.2422;" days old on your last birthday."
270     WAIT 3
280 NEXT I
290 END
```

Indentation Bounds

When indentation parameters attempt to force program statements too far to the right, they are bounded by the width of the screen minus 8 characters. That is, a program line will never start to the right of 8 characters from the right-hand edge of the screen. For instance, on an 80-column screen, a program line will never start to the right of column 72. Instead, all lines which should be indented farther to the right of this column will begin in this column; indentation remains in this column until the nesting level gets back to a manageable point, at which time the line beginnings will begin to drop back to the left.

Removing Indentation

To remove all indenting, execute this command:

```
INDENT 7,0
```

Finding Textual Patterns

When programs are larger than a couple of screenfuls, it is handy to have the computer search for a variable name, numeric or string literal, comment, etc. The non-programmable FIND command allows you to do this.

The following example searches the current program, beginning at the line currently being edited, for the letters "A pattern":

```
FIND "A pattern"  [Return] or [ENTER]
```

These letters may be a variable name, a string or numeric literal, or a comment (or a portion of any of these).

If you want to begin the search in a different place, then specify the range of lines to be searched:

```
FIND "A pattern" IN 200,650  [Return] or [ENTER]
```

Upon executing this command, BASIC begins a search for these characters. The following message is shown in the message/results line (below the "keyboard input line", which is near the middle of the screen in EDIT mode):

```
Finding "A pattern"
```

If the pattern is *not found*, then the system displays the following message:

```
"A pattern" not found
```

If an occurrence of these letters *is found*, the system displays the program line containing the pattern and a confirmation:

```
300   PRINT "A pattern of circles is shown on the display."

Found "A pattern"
```

You can choose any of the following actions:

- Edit the line (optional): move the cursor, and change, add, or delete characters.
- Press Return or ENTER to store the edited (or unchanged) line.
- Scroll the program up or down (with the ↑ or ↓ cursor keys), which cancels the FIND mode.
- Press CONTINUE (f2) to leave the line unchanged and continue the search.

If you choose to remain in FIND mode, press Return or ENTER. After checking syntax of the line, the FIND command will begin searching for the next occurrence of the specified characters; if the modified line contains a syntax error, you may correct the error and press Return or ENTER again. Once the line is syntactically correct, the FIND command begins searching for the next occurrence of the specified string.

You will remain in FIND mode as long as the FIND command has additional program lines to search. The system reminds you that you are in this mode by displaying these prompts at the bottom, right-hand corner of the screen:

```
                                                    Command



                                                              *
```

If you want to abort the FIND command, then use the Break (CLR I/O) key to cancel the mode. The system will display:

Search aborted at *nnnnn*; **"A pattern" not found.**

in which *nnnnn* is the line number at which the FIND was aborted.

Search-and-Replace Operations

The CHANGE command is similar to FIND, except that you will specify both a search pattern and a replacement pattern.

The following example searches for the pattern “Old text” and replaces it with “New characters”:

```
CHANGE "Old text" TO "New characters"
```

Return or ENTER

Like with FIND, the system shows that it is busy searching for a pattern:

```
Finding "Old text"
```

Also like FIND, the CHANGE command pauses when it finds the first occurrence of the search pattern; however, CHANGE also replaces the old pattern with the new one, and awaits your confirmation/rejection of the change:

```
200   PRINT "New characters."

"Old text" to "New characters"?
```

- To confirm the change, press Return or ENTER
- To reject the change, press CONTINUE (f2 in the System menu of an ITF keyboard).

If you want only the occurrences of the pattern in a certain program segment to be changed, then use the following syntax:

```
CHANGE "old" TO "New" IN 1, 250
```

If you want all occurrences of the pattern changed, with no capability of interactively confirming/rejecting the changes, use the following syntax:

```
CHANGE "old" TO "New" ; ALL
```

You can also combine these two specifications to change all occurrences within a range of lines:

```
CHANGE "old" TO "New" IN 1, 250; ALL
```

Copying Program Segments

While programming, you may encounter a need to duplicate several lines of BASIC code in another location of the program. With this BASIC system, the COPYLINES command provides an easy way of doing this kind of operation.

The first example below copies lines 180 through 220 to a location beginning at line 5205:

```
COPYLINES 180,220 TO 5205  [Return] or [ENTER]
```

The following example copies lines 300 through 3005 to a location beginning at line 100:

```
COPYLINES 300,3005 TO 100  [Return] or [ENTER]
```

If a program line already exists at the specified destination for the moved lines, it will be renumbered (along with as many subsequent lines) so as to make the move correctly. For instance, if you were copying 20 lines to begin at line 100, but there is already a line 100 and a line 105, then the COPYLINES command will renumber these existing lines to 120 and 121 before moving the 20 lines to 100 through 119. Note that the lines being copied are renumbered *before* the copy operation is actually performed.

Moving Program Segments

Earlier sections of this chapter showed how to use the [RECALL] key to recall a line which was previously deleted. You could use this technique to move one or two lines from one location in a program to another. However, when you are moving several program lines at a time—to a spot several screens away from the original location—there is an easier way: use the MOVELINES command.

This example command moves lines 32 through 127, inclusive, to a spot beginning at line 453:

```
MOVELINES 32,127 TO 453  [Return] or [ENTER]
```

The following example moves lines 300 through 3005 to a location beginning at line 100:

```
MOVELINES 300,3005 TO 100  [Return] or [ENTER]
```

If a program line already exists at the specified destination for the moved lines, it will be renumbered (along with as many subsequent lines) so as to make the move correctly. For instance, if you were moving 10 lines to begin at line 240, but there is already a line 240 and a line 245, then the MOVELINES command will renumber these lines to 250 and 251 before moving the 10 lines to 240 through 249.

Moving Lines into a Subprogram

One of the more frequent uses you may find for the MOVELINES command is in moving program lines from a "main context" into a separate "subprogram context" (defined by SUB and SUBEND statements). However, you may have noticed that to do so you must go to a line *below* all of the existing lines in memory and enter the SUB statement.

```
2100   SUBEND
2110   SUB New_subprogram
2120   _
```

After typing in this subprogram heading, you can use MOVELINES to move program lines from the main program (or from another subprogram) into the new subprogram:

```
MOVELINES 350,499 TO 2120
```

Don't forget to delimit the end of the new context with a SUBEND statement!

```
2630   SUBEND
2640   _
```

Deleting Multiple Lines

The DEL command, introduced in a preceding section, can also be used to delete several lines in a single operation. Blocks of program lines can be deleted by using two line identifiers in the DEL command.

- The first number or label identifies the start of the block to be deleted.
- The second number or label identifies the end of the block to be deleted.

The line identifiers must appear in the same order they do in the program. Here are some examples.

The following command deletes lines 100 thru 200, inclusively.

```
DEL 100,200
```

This command deletes all the lines from the one labeled "Block2" to the end of the program.

```
DEL Block2,32766
```

This command would do nothing except generate an error:

```
DEL 250,10
```

If you have subprograms or user-defined functions in your program, they can only be deleted in certain ways (such as with DELSUB). Primarily, the SUB or DEF FN statement cannot be deleted without deleting the entire subprogram or function. This subject is explained fully in the "Subprograms" chapter of *BASIC Programming Techniques.*

The DEL command is not programmable and cannot be used while a program is running.

Making Programs Readable

When first learning how to program, most people view the use of comments, long variable names, descriptive printouts, and other documentation tools as merely extra typing that isn't really necessary in their short programs. As time passes, old programs are expanded, new programs are written, and more people use the program. Eventually, software support activities become necessary. Some obscure bug is found or some exciting enhancement is requested. The programmer picks up a copy of a program written a year ago and can't begin to remember what "X1" was or why you would ever want to divide it by "X2". Program documentation can make the difference between a supportable tool that adapts to the needs of the users and a support nightmare that never really does exactly what the current user wants. Keep in mind that the local software support person just might be you.

This BASIC language makes it easy to write self-documenting programs. In addition to BASIC's standard REM (remark) statement, additional documentation features are:

- Descriptive keywords (such as REPEAT..UNTIL, LOOP..END LOOP, and so forth)
- Descriptive variable names (up to 15 characters)
- Descriptive line labels (up to 15 characters)
- End-of-line comments.

Contrast Between Documented and Undocumented Programs

Although this section deals primarily with commenting methods, all of these features work together to make a readable program. The following example shows two versions of the same program. The first version is uncommented and uses "traditional" BASIC variable names. The second version uses the features of HP's BASIC language to make the program more easily understood. Which version would you rather work with?

```
100  PRINTER IS 1
110  A=.03
120  B=.02
130  X=0
140  Y=0
150  C=A+B
160  PRINT "  Item       Total      Total"
170  PRINT "  Price       Tax       Cost"
180  PRINT "------------------------------"
190  P=0
200  INPUT "Input item price",P
210  D=P*C
220  E=P+D
230  X=X+D
240  Y=Y+E
250  DISP "Tax =";D;"Item cost =";E
260  PRINT P,X,Y
270  GOTO 190
280  END
```

```
100  ! This program computes the sales tax for
110  ! a list of prices. Item prices are input
120  ! individually. The tax and total cost for
130  ! each item are displayed. The running
140  ! totals for tax and cost are printed on
150  ! the CRT. Modify line 220 to change the
160  ! the system printer.
170  !
180  ! Sales tax rates are assigned on lines 230
190  ! and 240. The rates used in this version
200  ! of the program were in effect 1/1/81.
210  !
220  PRINTER IS CRT         ! Use CRT for printout
230  State_tax=.03          ! Local tax rates
240  City_tax=.02
250  !
260  Total_tax=0            ! Initialize variables
270  Total_cost=0
280  Tax_rate=State_tax+City_tax
290  ! Print column headers
300  PRINT "  Item      Total      Total"
310  PRINT "  Price      Tax       Cost"
320  PRINT "------------------------------"
330  !
340 LOOP  ! Start of main "Get Price" loop.
350    Price=0       ! Don't change totals if no entry.
360    INPUT "Input item price",Price
370    Tax=Price*Tax_rate
380    Item_cost=Price+Tax
390    Total_tax=Total_tax+Tax   ! Accumulate totals.
400    Total_cost=Total_cost+Item_cost
410    DISP "Tax =";Tax;" Item cost =";Item_cost
420    PRINT Price,Total_tax,Total_cost
430  END LOOP        ! Repeat loop for next item.
440  END
```

There are two methods for including comments in your programs. The use of an exclamation point is demonstrated in the second example program. The exclamation point marks the boundary between an executable statement and comment text. There does not have to be an executable statement on a line containing a comment. Therefore, the exclamation point can be used to introduce a line of comments, to add comments to a statement, or simply to create a “blank” line to separate program segments. Exclamation points may be indented as necessary to help keep the comments neat.

The REM statement can also be used for comments. The exclamation point is neater and more flexible, but the REM statement provides compatability with other BASIC languages. The REM keyword must be the first entry after the line identifier and must be followed by at least one blank. Note also that the REM statement moves when a program is indented (with INDENT); however, "!" comments do not get indentation changed (unless forced by other text in that line).

Here are some examples of proper and improper REM statements and "!" comments.

```
       Right                               Wrong
-----------------------------  -------------------------------

10 REM Check Book Balance      20 REMinitialize array

40 Start2: ! Subtotal loop     50 X=PI*R^2  REM Area of circle
```

General Recommendations for Commenting Programs

Each programmer has an individual style in the use of comments. Therefore, the following is not a list of rules. It is simply some suggestions on the effective use of comments.

1. Include a **programs heading** that answers the following questions:
    a. Why was the program written?
    b. What does it do?
    c. Who would probably use it?
    d. Who is the author of the program?
    e. When was the date of the last revision?
    f. Who is currently in charge of supporting it?
    g. What modifications could/should be made by a normal user?
2. **Describe all variables**, especially global variables. A descriptive variable name may do the job, or a more detailed explanation may be needed.
3. Describe any **hardware or software configuration** required for the proper running of the program. This may even include an explanation of how to modify the program to accommodate alternate devices (when such changes are reasonable).
4. **Make major blocks and entry points visible.** Many tools are available for this, including descriptive labels, indenting, spacing, and comments describing program flow.

(continued on following page)

5. **Use comments freely** to describe the action of complex lines, equations, fancy manipulations, and "low-level" operations like CONTROL statements and escape code sequences. These heavily coded operations can be very important to the computer but very mysterious to the human trying to read the program.

Software Security

There may be times when you want to keep portions of your programs from being read or used by other programmers or users. With this BASIC system, and Series 200/300 computer hardware, you can either prevent a program from being read, or from being executed unless you give the authorization.

Preventing Programs from Being Listed

With this BASIC system, you can use the SECURE statement to prevent program line(s) from being listed. (Another way is to simply make sure that the EDIT binary is not loaded in the system or available to anyone who you don't want to look at *any* programs. However, that is not a highly restrictive method since the EDIT binary is available as a standard component of most Series 200/300 BASIC systems.)

The following example secures lines 30 through 60 from being listed (either with the editor or by using the LIST statement):

```
SECURE 30,60
```

Here is what the program might look like—either with the editor or as the output of a LIST statement:

```
10   ! Example of SECURE'd program.
20   ! Begin password check routine.
30*
40*
50*
60*
70   ! End of password check.
80     .
       .
       .
```

If you want the whole program to be secured, use this statement:

```
SECURE
```

Note

Once a program is secured, it **cannot** be un-secured. Therefore, you should keep an un-secured back-up copy of all programs.

Other Security Measures

There is also another method of preventing software from being used by anyone who may acquire a copy of it. You can write a routine that checks the serial number of a computer, or the serial number of an optional HP 46084 ID Module. Then your routine can determine whether or not to permit the rest of the program to be executed on this hardware configuration.

Reading an ID PROM

To read the serial number of a computer, use the following statement:

```
SYSTEM$("SERIAL NUMBER")
```

The function returns the contents of the ID PROM, if present, or the null string if no ID PROM is present.

Reading an ID Module's Contents

The same function:

```
SYSTEM$("SERIAL NUMBER")
```

also returns the **encoded** contents of an ID Module. In order to decode an ID Module's contents, use the "ID_MODULE" program supplied on the *Manual Examples* disc.

A Closer Look at Storing Programs

To record a program, you will use either the SAVE or the STORE statement. There is no "right" or "wrong" choice; your choice depends upon the kind of file you want.

- STORE records an internal representation of the program in a PROG file. The main advantage of a PROG file is a rapid retrieval rate.
- SAVE[1] records the actual text of the program in an ASCII file. The main advantage of an ASCII file is that it can be read as data by a BASIC program or many other LIF-compatible[2] devices (such as other HP computers and terminals).

The following table gives a brief summary of the differences between SAVE and STORE.

Characteristic	SAVE	STORE
File type created:	ASCII	PROG
Retrieved by:	GET	LOAD
Approximate storage speed:	900 bytes/s	13 000 bytes/s[3]
Approximate retrieval speed:	300 bytes/s[4]	14 000 bytes/s[3]
Can file be read as data?	Yes	No
LIF-compatible file?	Yes	No
Arbitrary program segments allowed?	Yes	No
Subprograms included?	No	Yes
Can use LOADSUB to retrieve a subprogram?	No	Yes

[1] Using SAVE requires the EDIT binary.

[2] LIF is the acronym for Logical Interchange Format, which is a disc format used by several HP divisions. (Note that the first letter of the file name must be a letter; in addition, some LIF-compatible devices restrict file names to uppercase letters and the decimal digits 0 through 9.)

[3] The speeds for LOAD and STORE are approximate for an interleave factor of 1 on an HP 9836 internal disc drive. Interleave factors greater than this will cause a corresponding decrease in speed.

[4] The retrieval speed for GET is very data-dependent. On an HP 9836 with a clock rate of 8 MHz, for instance, it can vary from 20 bytes/second to 600 bytes/second (and maybe beyond those limits) according to the contents of the file and the syntax checking required to enter the lines into program memory. See the "Efficient Use of the Computer's Resources" for a discussion of how to time various computer operations.

Using STORE

The following command creates a program file called "MyProgFile" on the current default volume:

```
STORE "MyProgFile".
```

If you should happen to get an error 54, that means that there is already a file on the disc with the name you are using. In this event, you have three choices.

- Pick a different name that doesn't already exist. To determine which file names are already being used, execute a CAT command.
- You may want to replace the existing file with a new one (like when you are updating program files with new, improved versions). To replace an existing file, use the RE-STORE statement. For example, the command to replace a program file called "BEAMS" is:

  ```
  RE-STORE "BEAMS"
  ```

  Note that the hyphen must be used in the RE-STORE statement. (RESTORE without a hyphen is used for an entirely different operation involving the pointer associated with DATA statements; see the "Data Storage and Retrieval" chapter of *BASIC Programming Techniques* for details.)
- PURGE the old file, then STORE the new one.

Using SAVE

The SAVE operation is similar to the STORE operation in that it stores the current program in a file; however, it has one additional feature. The SAVE statement allows line identifiers that specify what portion of the program you want to save. This is especially helpful when moving or appending program segments during major editing operations. Here are some examples of using the SAVE statement.

To save all of a program in an ASCII file called "WHALES", on the current default volume, execute the following command:

```
SAVE "WHALES"
```

The following command saves the last part of a program, from line 500 to the end, in an ASCII file called "LastPart".

```
SAVE "LastPart",500
```

When both the starting and ending lines are specified, any arbitrary portion of a program can be saved. Executing the following command saves that portion of a program that is between the lines labeled "Sort" and "Printout" (inclusive) in an ASCII file called "Sorter".

```
SAVE "Sorter",Sort,Printout
```

There is also a RE-SAVE statement that allows an existing file to be replaced by a newly created file with the same name. For example, to update an ASCII file called "Analysis" with a new version of the program, the following command would be used.

```
RE-SAVE "Analysis"
```

Creating vi-Compatible Files

If you want to create a file that you can edit with the HP-UX system's *vi* editor, here are the steps to take:

1. Use the CREATE statement to create a file of type HP-UX:

   `CREATE "ux_file",1` *(number of records is not important)*

2. Use the RE-SAVE statement to save the program in the file you just created:

   ```
   RE-SAVE "ux_file"
   ```

   The reason for the RE-SAVE (instead of just SAVE) is that the BASIC system would have created an ASCII type file with SAVE, while the RE-SAVE will maintain the file type (which is in this case a file of type HP-UX).

What to Do Next

Task/Topic	Chapter Number
Learn about how to load and run programs.	9
Learn about how to use and manage files.	10
Learn about how to maintain your system.	12 thru 17
Learn about utilities available with the BASIC system.	18
Learn about each key on your keyboard.	19 thru 21

Notes

Maintaining the BASIC System

This **part** of the *Installing, Using, and Maintaining the BASIC System* manual contains an overview and several chapters that describe how to maintain your BASIC system. Some tasks are done routinely— others are done as necessary.

The sections in the overview provide general information that you can consider before doing the maintenance described in subsequent chapters. In particular, see the section called "Conventions for Using Keys".

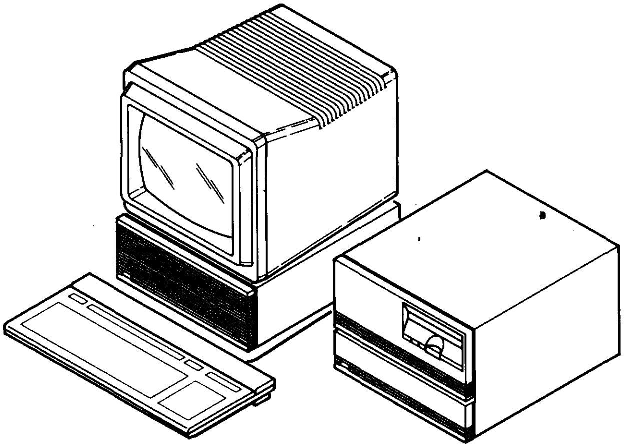
The table on the following page lists the chapters and major tasks with page numbers.

Overview 12

This chapter lists prerequisites to maintaining BASIC 5.0; explains some conventions for using keys; and discusses back-ups. In particular, note the conventions for using keys.

Prerequisites

Prior to any maintenance, you should have already installed and loaded BASIC and the binaries required for all of your mass storage devices according to the chapters in the part of this manual called "Installing BASIC".



While you do maintenance, it is helpful to have labeled devices such as disc drives with their **volume specifiers** (e.g. `:,1402` **or** `:,702,1` **or** `:CS80,700`) according to the chapter in this manual called "Verifying and Labeling Peripherals". You will see the abbreviation for volume specifier— **msvs**— in documentation, prompts, and messages; it means "mass storage volume specifier".

Conventions for Using Keys

In maintaining your system, you execute BASIC statements or use menu-driven utilities that require you to type certain keys. Here are the conventions for typing keys that apply in the chapters in this part of the *Installing, Using, and Maintaining the BASIC System* manual.

- **Running programs**: In BASIC, you can always run a program by typing:

  RUN Return or ENTER

  The ITF keyboard uses Return. The HP 98203A/B/C keyboard uses ENTER.

  The following chapters indicate that you should type Return and let you make the translation if you do not have an ITF keyboard. If you have executed a RUN statement, you can subsequently recall statements until you get the RUN statement and execute it (possibly with some editing) by typing Return. With an ITF keyboard, you can often run a program by typing the softkey for the RUN softkey label; and with a HP 98203A/B/C keyboard, you can type RUN.

- **Using Softkeys or Keys**: The HP 98203A/B/C keyboard has some keys such as CONTINUE that are often used in BASIC programs and that are not duplicated on the ITF keyboard. The utilities that maintain your BASIC system use many of these keys; and they typically give you two alternatives for typing. One is to have you type a key on the HP 98203A/B/C keyboard (e.g. type CONTINUE). If you have an ITF keybaord, the keys on the HP 98203A/B/C keyboard are duplicated as softkeys (e.g. type the softkey for the CONTINUE softkey label). In these cases, you need to look for the label and type the corresponding softkey. The following chapters simply indicate that you should continue (or select or proceed) and let you type the appropriate key or softkey. You get more explicit directions if there is a possibility for confusion.

Choosing the Back-Up Media

The following types of media can contain back-up copies of discs and files.

- **Flexible discs** are the **most common media** used for back-up copies. (They are your only choice if your only mass storage devices are flexible disc drives.) Before you begin to back up one or more discs, you need a blank, initialized flexible disc for each disc to be copied. You can get information about initializing discs in the "Preparing Flexible Discs" chapter.
- **Hard discs** are **not recommended as a back-up media**, unless you have two discs. In general, you use a hard disc as your primary "working" disc, and use either flexible discs or cartridge tapes as your back-up media.
- **Cartridge tapes** (DC600) are **useful in backing up hard discs**. They have large capacities (16 or 67 MegaBytes of storage space), and they save you the time and effort of swapping several flexible discs while backing up large quantities of data.
- **Paper** is a reliable way to keep **back-up copies of source programs**. (Use the LIST statement to make back-up copies of programs.) Paper copies of data files may not be as useful (unless they are files that contain only text). Having paper copies of system or binary files is not feasible or useful unless your native tongue is MC68000 machine language.

Backing Up Files

You should protect your software investment by making back-up copies of your BASIC Language System discs. Put the original discs in a safe place and use the backup copies for every-day work. Beyond backing up the BASIC discs, it is good practice to back-up your data and other discs on a periodic basis. This minimizes the loss of data, programs, and programming time for discs that are subsequently damaged.

There are two ways to perform volume and file back-up's:

- Have BASIC running and **use COPY statements** when you have LIF flexible discs, or you want to backup a few files.
- **Use the BACKUP utility** on the *BASIC Utilities* disc when you have an HFS hard disc, or you want to backup several files.

Chapter 13 explains how to use COPY statements, and chapter 14 explains how to use the BACKUP utility.

Using COPY for Back-Ups 13

In using COPY to make back-up copies:

- Use **Procedure 1** if you have *two* or more flexible disc drives.
- Use **Procedure 2** if you have *one* flexible disc drive.

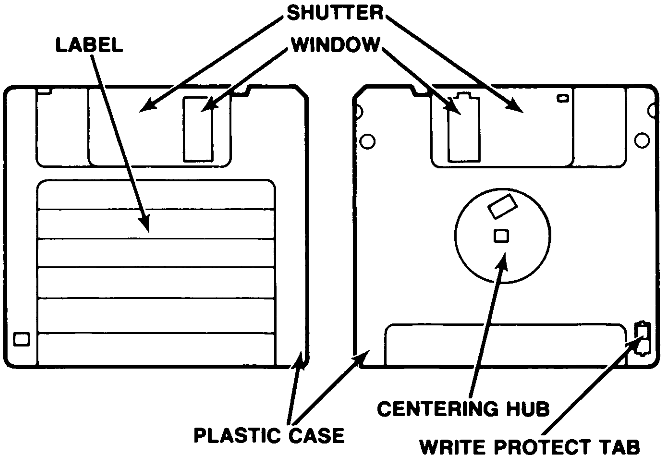
Go directly to the procedure you need.

Procedure 1: Copying With Two Drives

This procedure shows how to use two disc drives to copy an entire disc or an individual file. For the example of learning to copy a disc, you will make a back-up copy of your *BASIC System* disc. For the example of learning to copy an individual file, you will copy the SYSTEM_BA5 file on the *BASIC System* disc. The following steps show both cases.

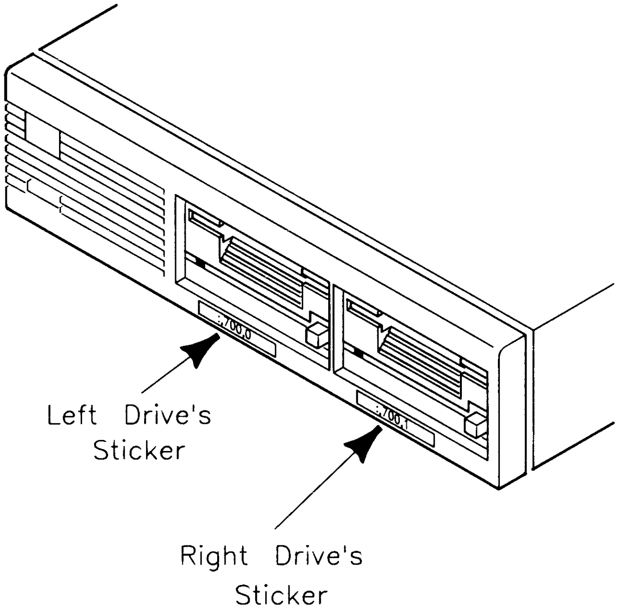
STEP 1: Write Protecting the Source Disc

Note the drawing and make sure your source disc is **write-protected**. This prevents inadvertantly destroying its information.



STEP 2: Inserting the Source and Destination Discs

Insert the source disc (the *BASIC System* disc in this example) into **one** of your flexible disc drives **and** insert the destination disc into the other flexible disc drive. **If you backup an entire disc, the existing contents are destroyed**. Try to have a designated destination disc so that an "old" backup is overwritten by a "new" backup. Examine the following drawing and note the use of volume specifiers. **Hereafter, we assume you will supply appropriate volume specifiers.**



Note in subsequent steps that for an entire disc, you will type statements such as:

```
COPY ":,700,0" TO ":,700,1" Return
```

and for a single file, you will type statements such as:

```
COPY "SYSTEM_BAS:,700,0" TO "SYSTEM_BAS:,700,1" Return
```

Relate the use of volume specifiers in the above statements to the labels on the stickers shown in the drawing. Hereafter, it is assumed you will use the apprpriate **msvs** in your statements or options.

STEP 3: Copying a Disc or File

In either case during a copy, wait until the the asterisk, *, disappears from lower-right display. (If you have KEY LABELS ON, `Command` is also shown with the *; and when the command is completed, `Idle` is shown.)

- To **copy an entire disc**, note the volume specifiers on the labels you stuck on your source and destination disc drives during installation. (See the chapter called "Verifying and Labeling Peripherals" if you need help with this).

  The general statement for copying a disc is:

  ```
  COPY ":,msvs" TO ":,msvs" Return
  ```

  where *msvs* is the source volume specifier (such as `700,0`) and the second *msvs* is the destination volume specifier (such as `700,1`). With this in mind, an appropriate COPY statement might be:

  ```
  COPY ":,700,0" TO ":,700,1" Return
  ```

- **Copying an individual file** is similar to copying a disc except that you also include, respectively, the name of the source file and the destination file ahead of the *:,msvs*. Therefore, to copy the SYSTEM_BA5 file from the *BASIC System* disc onto the back-up disc, using the same volume specifiers as were used above, type:

  ```
  COPY "SYSTEM_BA5:,700,0" TO "SYSTEM_BA5:,700,1" Return
  ```

STEP 4: Verifying the COPY

Get a list of files on the destination disc and verify the COPY by typing:

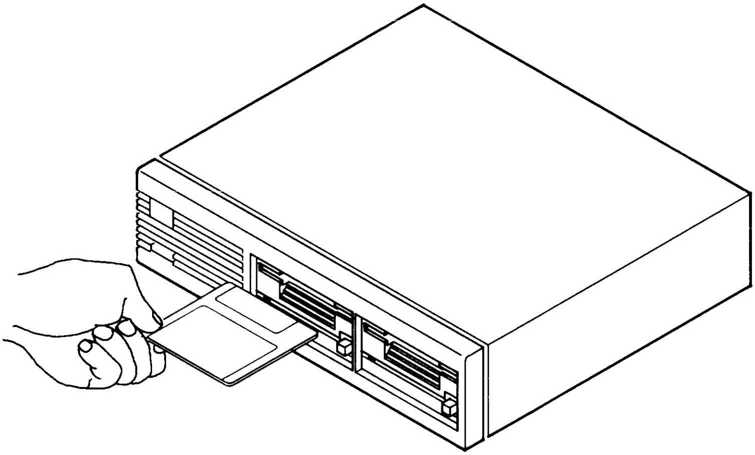
```
CAT ":,700,1" Return
```

You should get the list of files you copied. If you do not get them, check to see that you had the discs in the correct drives and repeat the COPY.

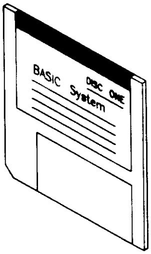
STEP 5: Repeating the COPY Procedure

Now that you are set up and are familiar with the procedure, you might want to copy some more discs or files.

- To **copy another individual file** on the current source and destination discs, repeat steps 3 and 4 for each file to be copied.
- To **copy another entire disc**, note the following picture and remove the source (*BASIC System*) disc and the back-up (destination) disc from their drives.



Note the following picture and print a label for the duplicate disc using the same information that appears on the original disc. Print "Copy" and the current date on the label. Peel the backing off of the label, stick the label on the disc, and repeat steps 1 through 6 for each disc to be copied.



Procedure 2: Copying With One Drive

This procedure shows how to use one disc drive to copy an entire disc or an individual file. The major difference from using two disc drives is that, in using one disc drive, you employ a **memory volume** as an "intermediate disc drive".

To copy a disc, you will make a back-up copy of your *BASIC System* disc. To copy an individual file, you will copy the SYSTEM_BA5 file on the *BASIC System* disc. The following steps explain the procedure for both a whole disc and a single file.

STEP 1: Creating the Memory Volume

Estimate the amount of memory you need for the COPY and create a **memory volume** in your computer's main memory by typing a statement like one of the following according to your estimate:

- For a typical 270-Kbyte disc, type:

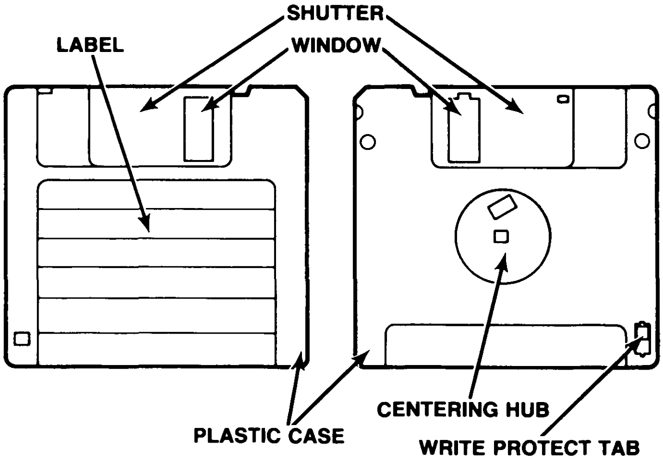
  `INITIALIZE ":MEMORY,0"` Return or

- For a disc that has 680-Kbytes, allowing enough 256-byte sectors to encompass 680-Kbytes and noting that the last parameter— 2720— is based on 2720 timés 256 which is 696 320 bytes (680-Kbytes), type:

  `INITIALIZE ":MEMORY,0,2720"` Return or

STEP 2: Write Protecting and Inserting the Source Disc

Note the drawing and make sure the source (original) disc is **write-protected**, which prevents destroying the information it contains.



Insert the write-protected source disc (the *BASIC System* disc for this example) into your flexible disc drive.

STEP 3: Copying to the Memory Volume

Look at the following alternatives and COPY to the memory volume. In either case, when the copy begins, wait until the the asterisk, *, disappears from lower-right display.

- To **copy an entire disc** of 270-Kbytes, note the volume specifier on the label you stuck on your source disc drive during installation. (See the chapter called "Verifying and Labeling Peripherals" if you need help with this).

  The general statement for copying a disc to memory is:

  ```
  COPY ":,msvs" TO ":MEMORY,0" Return
  ```

  where *msvs* is a source volume specifier such as 700,0. With this in mind, an appropriate COPY statement might be:

  ```
  COPY ":,700,0" TO ":MEMORY,0" Return
  ```

- **Copying an individual file** is similar to copying a disc except that you also include the name of the source file ahead of *:,msvs*. Therefore, to copy the SYSTEM_BA5 file from the *BASIC System* disc into memory using the same volume specifiers as were used above, type:

  ```
  COPY "SYSTEM_BA5:,700,0" TO ":MEMORY,0" Return
  ```

STEP 4: Copying from Memory to the Destination Disc

Remove the source disc (the *BASIC System* in this example) from the disc drive and insert the destination (back-up) disc. Copy the disc (or file) from memory to the destination disc according to the appropriate case:

- If you were **copying an entire disc** using the preceding assumptions, type:

  ```
  COPY ":MEMORY,0" TO ":,700,0" Return
  ```

- If you were **copying an individual file** using the preceeding assumptions, type: type:

  ```
  COPY "SYSTEM_BA5:MEMORY,0" TO "SYSTEM_BA5:,700,0" Return
  ```

You should now have a copy of a disc or file on the destination disc.

STEP 5: Verifying the COPY

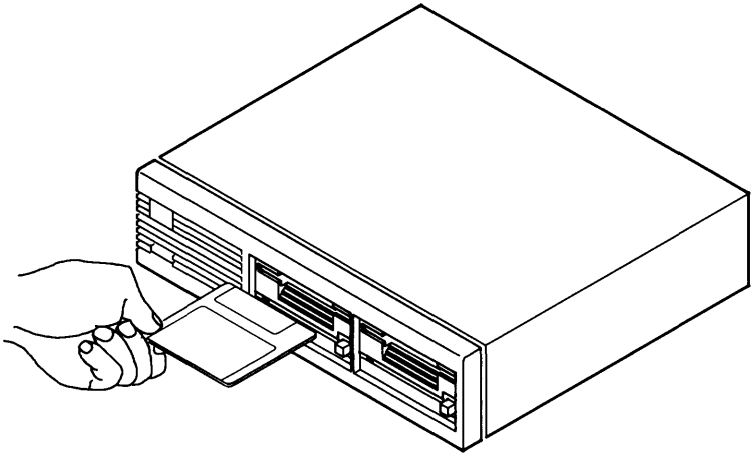
To determine that the copy is on the destination disc and verify the COPY, type:

```
CAT ":,700,0"
```

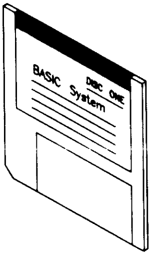
STEP 6: Repeating the COPY Procedure

Now that you are set up and familiar with the procedure, you might want to copy some more discs or files.

- To **copy individual file(s)**, repeat steps 3 through 5 for each file to be copied. (You might want to change source and/or destination discs between copies to suit your needs.)
- To **copy an entire disc**, remove the source (*BASIC System*) disc and the back-up (duplicate) disc from their drives according to the picture.



Print a label for the duplicate disc; use the same information that appears on the original disc. Print "Copy" and the current date on the label. Peel the backing off of the label, stick the label on the disc, and repeat steps 1 through 6 for each disc to be copied.



Using the BACKUP Utility 14

The BACKUP/RESTORE utility on the *BASIC Utilities* disc, provides a menu-driven means of backing up, restoring, or cataloging several discs or files. (Note that, if your utilities are on more than one disc, do a CAT to see which disc has a file named BACKUP.)

The BACKUP/RESTORE utility employs the format used by the HP-UX ***cpio*** command; and this means that you can **access the files on back-up media** in the BASIC, Pascal, **and** HP-UX systems.

Before you use the utility, note that if you are going to copy only one file or disc, executing a COPY statement in BASIC is an easier and quicker means of backing up a disc or file; see the earlier chapter called "Using COPY for Back-Ups".

Loading the BACKUP Utility

Insert the appropriate utilities disc in your flexible disc drive.

Load the utility into memory by typing the following statement, substituting an appropriate specifier such as `702` or `700,1` for *msvs*:

```
LOAD "BACKUP:,msvs" Return
```

(Note that, if you have an HP 98203A/B/C keyboard, you type ENTER instead of typing Return. Hereafter, you just see Return.)

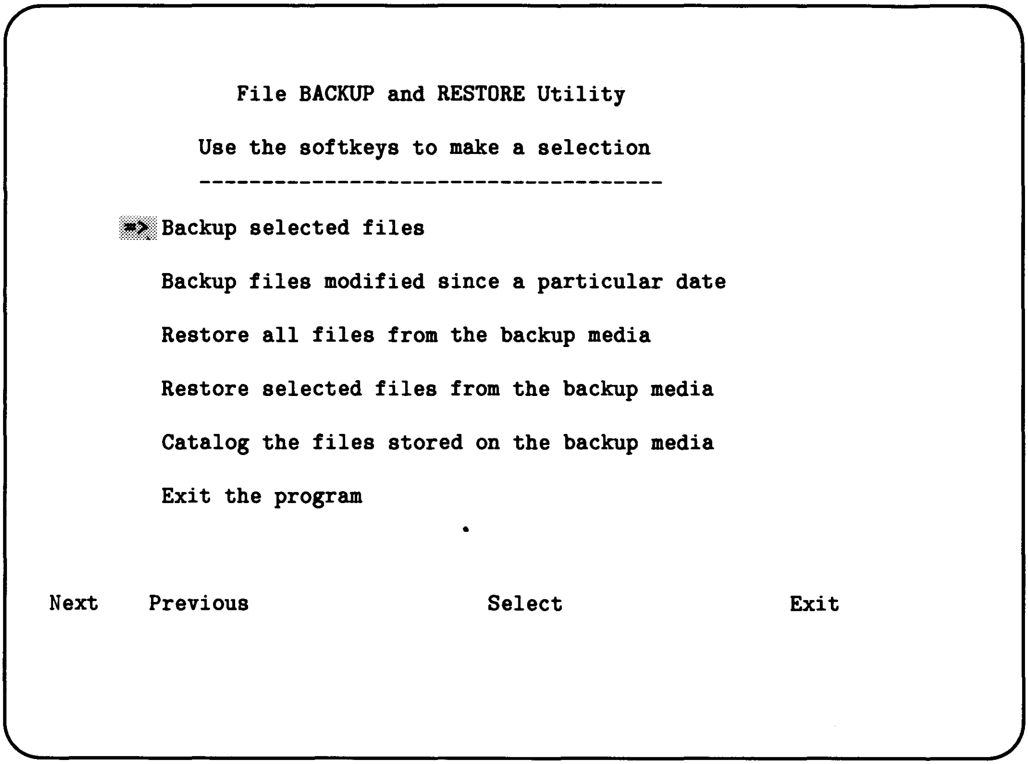
When the program has been loaded (the * disappears from the lower-right display), run the program by typing:

```
RUN Return
```

Note that you have several alternatives for running a program besides the above entry. With an HP 98203A/B/C keyboard, you can simply type RUN. With an ITF keyboard, you typically have a `RUN` softkey label in the System or User menus, and you can run a program by typing the corresponding softkey (one of the keys f1 through f8).

BACKUP/RESTORE Menu of Options

On running the BACKUP/RESTORE utility, you get the following menu of options. Read the next section called "Wildcard Names and Specifiers" to see how to interpret prompts and specify discs or files. Then, read the sections in the "Using BACKUP Utility Options" section according to your needs.



```
                    File BACKUP and RESTORE Utility

                 Use the softkeys to make a selection
                 -------------------------------------

          => Backup selected files

             Backup files modified since a particular date

             Restore all files from the backup media

             Restore selected files from the backup media

             Catalog the files stored on the backup media

             Exit the program


 Next    Previous                  Select                  Exit
```

Wildcard Names and Specifiers

Most of the options in this utility let you specify any of the combinations of directory paths, file names, and volume specifiers shown in the following table. The = character is known as a **wildcard** because it represents any combination of characters. (If you don't know what a wildcard is, look through the examples; you should be able to understand its definition from them.) The examples appear without quotes, but in general, you enclose them with double quotes.

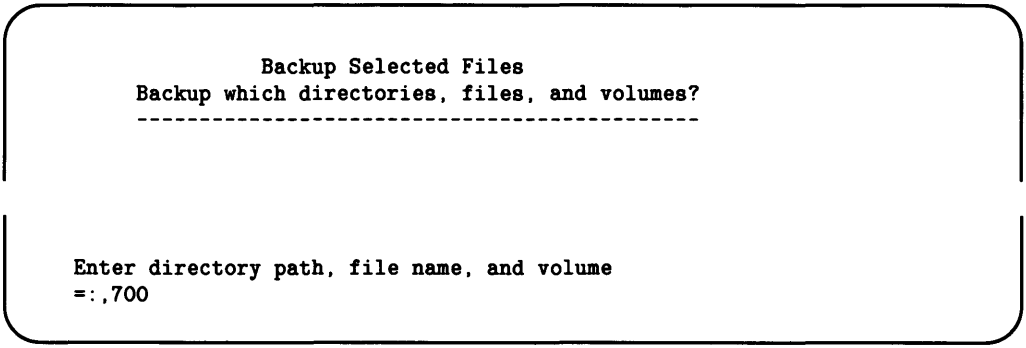
Item	Examples	File(s) Specified
:*volume*	`:,700`	the volume at select code 7, address 0, unit 0
	`:CS80,802,1`	the volume at select code 8, address 2, unit 1
file name:*volume*	`ONE_FILE:,700,1`	only the file `ONE_FILE` on volume `:,700,1`
	`=:,702`	all files on volume `:,702` (in current working directory)
directory path:*volume*	`/ONE_DIR:,700`	the directory `ONE_DIR` on volume `:,700`
	`/=:,802,1,1`	all files on volume `:,802,1,1`
path/file:*volume*	`/ONE/TWO/=:,700`	all files and directories in `/ONE/TWO` on volume `:,700`
	`TWO/=:,702`	all files in `TWO:,702` (this directory is subordinate to the current working directory)

Using BACKUP Utility Options

The following subsections explain the options and show examples of how to use them.

Backup selected files Option

Selecting this option prompts:



```
                  Backup Selected Files
        Backup which directories, files, and volumes?
        ---------------------------------------------


    Enter directory path, file name, and volume
    =:,700
```

Note the prompt, =:,700. The = character, known as a **wildcard**, was described in an earlier section. The default backup specification, =:,700, would backup all files (the =) on the current MSI'd device(the :,700). In general, you will type the specification you want, and that specification "types over" the =:,700. For example, to backup all files on the volume specified by :,702,1, you would type:

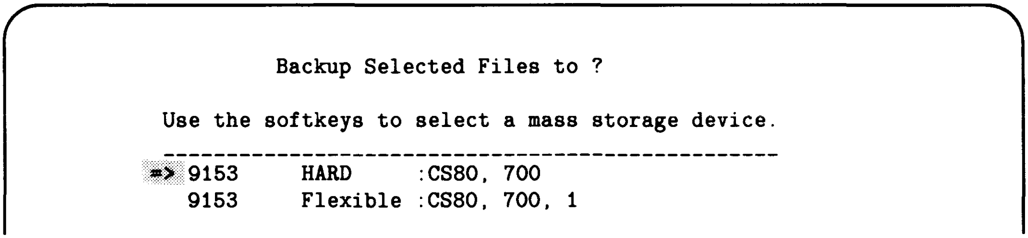
```
=:,702,1 Return
```

Note that when you type Return, the specification is displayed and you can type another specification for files you want to backup. Your second entry, for example, might be something like:

```
/WORKSTATIONS/CODE/ACE/New_file:,701,1 Return
```

While you are specifying files to backup, you can use the features provided by the **Recall** key and you can use the arrow keys to move among the displayed specifications. After you have specified all the files you want to backup, type the key for Proceed to get the following menu.

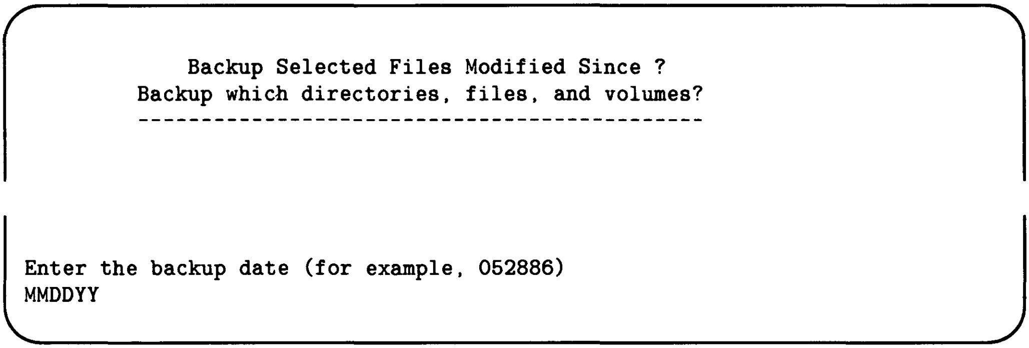
Noting the available mass storage devices, which will look something like what you see on the following display, use arrow keys (or softkeys for PREV or NEXT) to move the pointer to the desired backup media. Typing Return initiates the backup. **Be aware that the files on the backup media will be destroyed.** If the back up media overflows, the utility prompts you to install another back-up disc or tape. When the back-up is complete, the utility returns to the main menu, and you can select another option or exit the utility.



```
                Backup Selected Files to ?

           Use the softkeys to select a mass storage device.
            ------------------------------------------------
          => 9153     HARD     :CS80, 700
             9153     Flexible :CS80, 700, 1
```

Backup files modified since a particular date Option

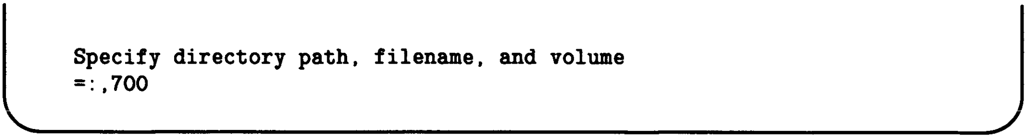
Selecting this option initially prompts you to enter a date as a sequence of two-digit numbers for month, day, and year (e.g. 052886 is May 28, 1986):



```
             Backup Selected Files Modified Since ?
         Backup which directories, files, and volumes?
         ---------------------------------------------



 Enter the backup date (for example, 052886)
 MMDDYY
```

Enter the desired date and type Return.

You are then prompted to specify directories, file names, and volumes for the files to be copied in the same manner as was shown in the previous section called "Backup selected files Option".



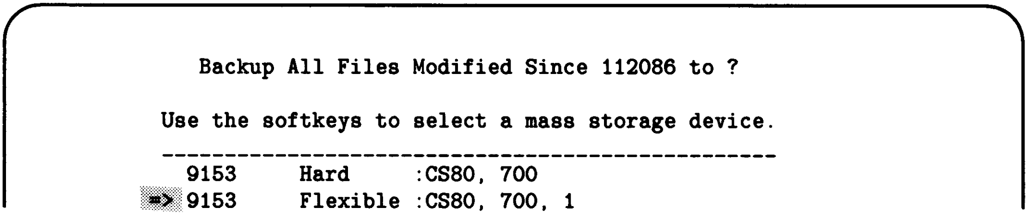
```
Specify directory path, filename, and volume
=:,700
```

For example, to back up all files on a LIF volume specified by `:,702`, replace the `=:,700` by typing:

`=:,702` Return

You can enter additional file specifications, and after you have entered the ones you want, type the key for **Proceed**.

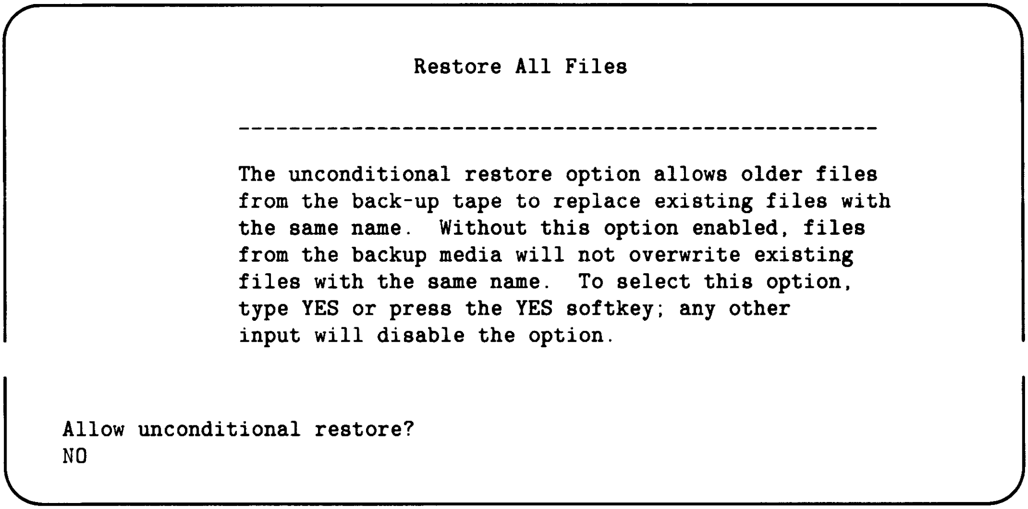
Then, you examine the following list and use the arrow keys (or softkeys for **PREV** or **NEXT**) to select the backup device. Typing Return completes the copy. If the back up media overflows, the utility prompts you to insert a different back-up disc or tape. When the back-up is complete, the utility returns to the main menu, and you can select another option or exit the utility.



```
      Backup All Files Modified Since 112086 to ?

   Use the softkeys to select a mass storage device.
   ------------------------------------------------
     9153     Hard     :CS80, 700
  => 9153     Flexible :CS80, 700, 1
```

Restore all files Option

Selecting this option provides some directions and prompts you to type "YES" to allow an unconditional restore or type something else to disable the unconditional option.



```
                              Restore All Files

                   ---------------------------------------------------

                   The unconditional restore option allows older files
                   from the back-up tape to replace existing files with
                   the same name.  Without this option enabled, files
                   from the backup media will not overwrite existing
                   files with the same name.  To select this option,
                   type YES or press the YES softkey; any other
                   input will disable the option.


     Allow unconditional restore?
     NO
```

The following **example** restores LIF files to a hierarchical volume. The example **assumes** you have **already** created the directories on the hierarchical volume. If not, create them now if you want to work through the example; and refer to "Hierarchial Directories" in the "Using and Managing Files and Directories" chapter if you need help.

Your answer to the above prompt indicates whether or not files on the back-up media are to **replace** files of the same name on the media being restored.

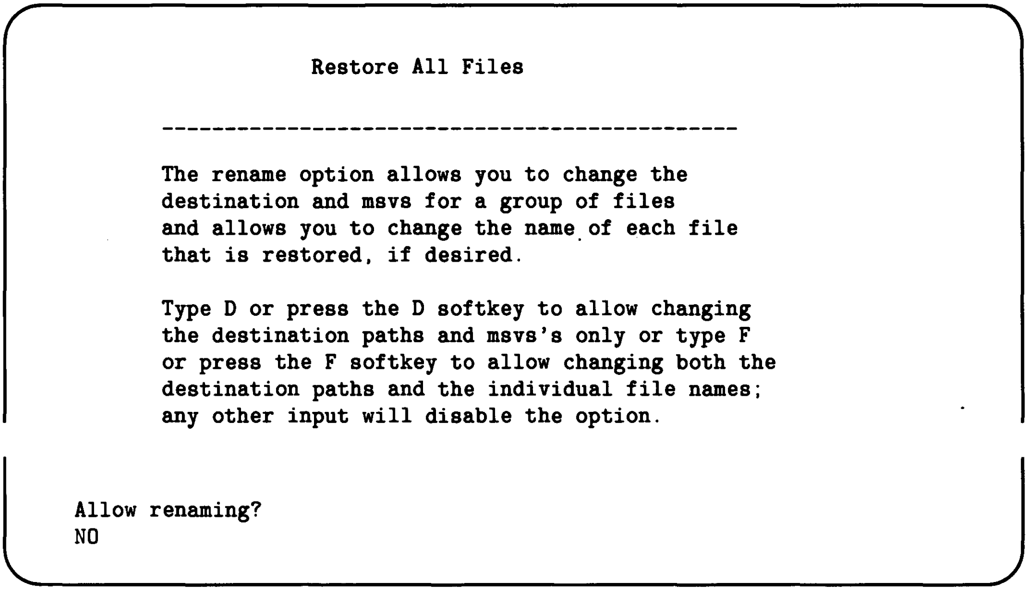
- `NO` indicates that files on the back-up media are **not** to replace any files with the **same name** on the media being restored.
- `YES` indicates that files on the back-up media **are** to replace any files with the **same name** on the media being restored (the existing files are overwritten, which destroys any data they contain).

For this example, assume you **do not** want to automatically restore all files (without a chance to confirm the replacement), so typing the softkey for `NO` provides the following information and prompt.

Your answer to the prompt shown below indicates whether or not you want the option of renaming each file and directory that is being restored from the back-up media:

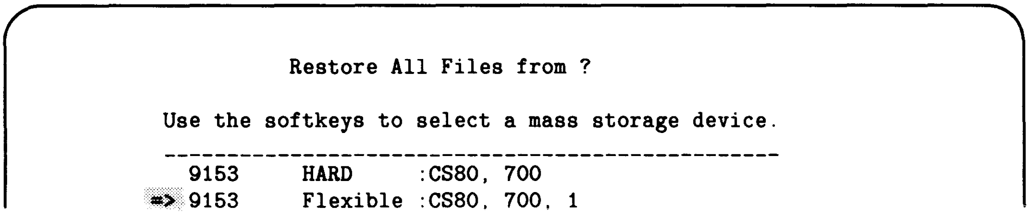
- `NO` indicates that files on the back-up media are **not** to be renamed (all files will retain current name and directory location).
- `YES` indicates that you want the **option of renaming** files on the back-up media as they are being restored (you will be prompted for each file's name and directory location).

This example restores the files to the same locations from which they came; so type the key for `NO`.



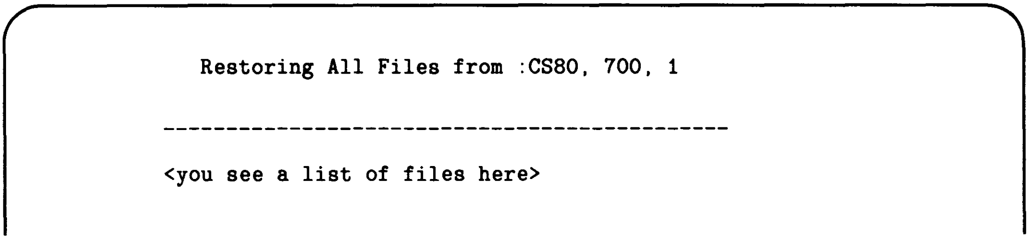
```
                     Restore All Files

          ----------------------------------------------

          The rename option allows you to change the
          destination and msvs for a group of files
          and allows you to change the name of each file
          that is restored, if desired.

          Type D or press the D softkey to allow changing
          the destination paths and msvs's only or type F
          or press the F softkey to allow changing both the
          destination paths and the individual file names;
          any other input will disable the option.


    Allow renaming?
    NO
```

As in other examples, after you set things up, the utility provides the following menu in which you use the arrow keys (or softkeys for PREV or NEXT) to select the backup device and type [Return] to initiate the backup.



```
             Restore All Files from ?

    Use the softkeys to select a mass storage device.
    -------------------------------------------------
      9153     HARD     :CS80, 700
   => 9153     Flexible :CS80, 700, 1
```

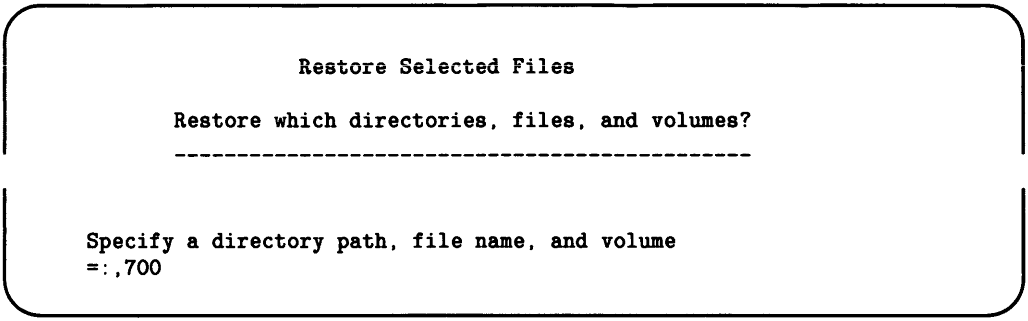
During the backup, you see the following display; and if the back up media overflows, the utility prompts you to install a different back-up disc or tape. Of course, your list of files depends on what you did. When the back-up is complete, the utility returns to the main menu where you can select another option or exit the utility.



```
       Restoring All Files from :CS80, 700, 1

    ---------------------------------------------

    <you see a list of files here>
```

Restore selected files Option

Selecting this option provides the following prompt which lets you specify the file name(s), wildcard(s), directory paths, and volume specifiers for all files you want to restore (see the next example).



```
                    Restore Selected Files

          Restore which directories, files, and volumes?
          ----------------------------------------------


    Specify a directory path, file name, and volume
    =:,700
```

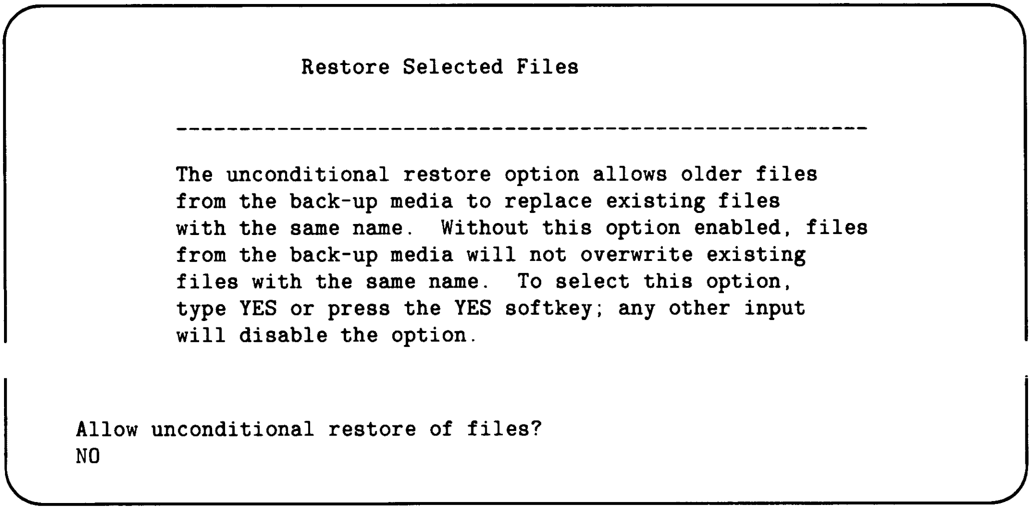
Typing a **specification** such as the following one restores the files from a hierarchical disc in the directories named: `/WORKSTATIONS`, `/WORKSTATIONS/BIN5.0`, and `/USERS/MARK`.

```
/WORKSTATIONS Return
/WORKSTATIONS/BIN5.0 Return
/USERS/MARK Return
```

When you have specified all files to be restored, typing the key for `Proceed` provides the following information and prompt that lets you indicate whether you want an unconditional restore.

- Entering `NO` indicates that files on the back-up media are **not** to replace any files with the **same name** on the media being restored (any back-up file's name that matches an existing file's name will **not** be restored).
- Entering `YES` indicates that files on the back-up media **are** to replace any files with the **same name** on the media being restored (you will be prompted to confirm or change each file's name and directory location).

This example assumes you want to automatically restore all files (**without a chance to confirm** the replacements), so type the key for `YES`.

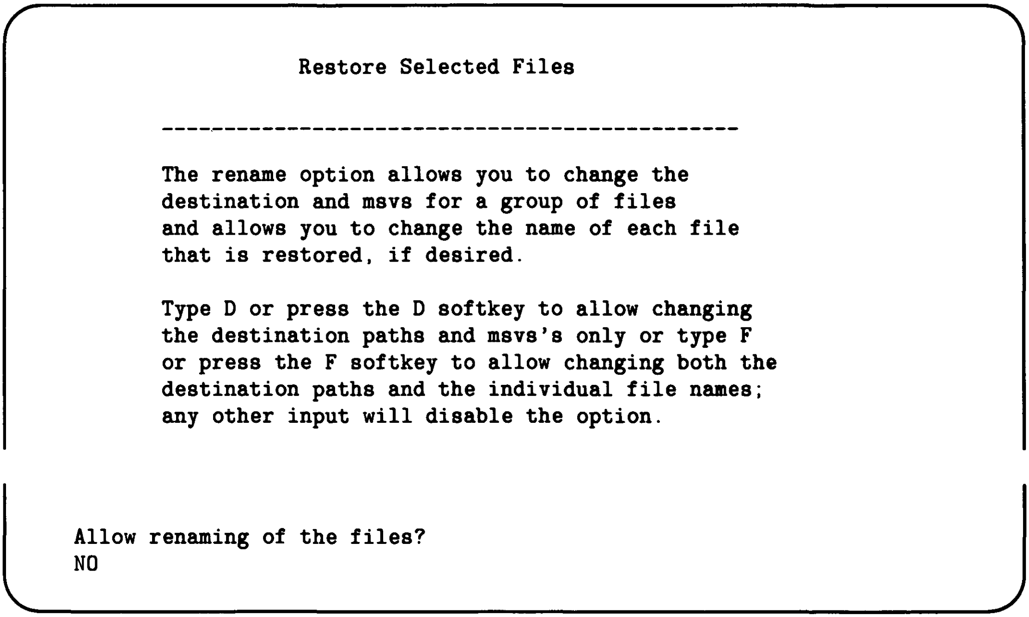


```
                        Restore Selected Files

             -------------------------------------------------------

             The unconditional restore option allows older files
             from the back-up media to replace existing files
             with the same name.  Without this option enabled, files
             from the back-up media will not overwrite existing
             files with the same name.  To select this option,
             type YES or press the YES softkey; any other input
             will disable the option.


     Allow unconditional restore of files?
     NO
```

You then get the following information and prompt which lets you indicate whether to allow renaming of files.

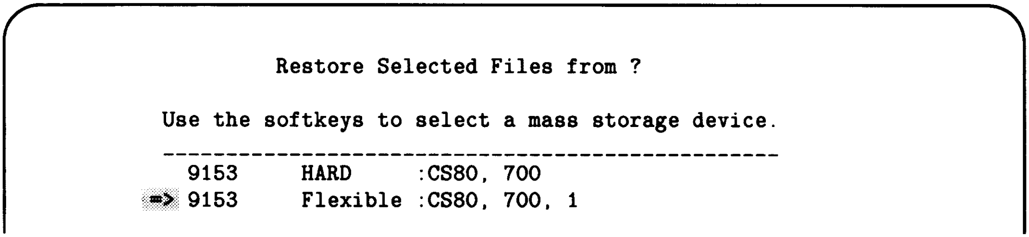
- Entering NO indicates that files and directories on the back-up media are **not** to be renamed (all file names and directory locations will remain the same).
- Entering YES indicates that you want the **option of renaming** files and directories on the back-up media as they are being restored (you will be given the chance to confirm or change the name and directory location of every file).

This example restores the files to the same locations from which they came; so type the key for NO.



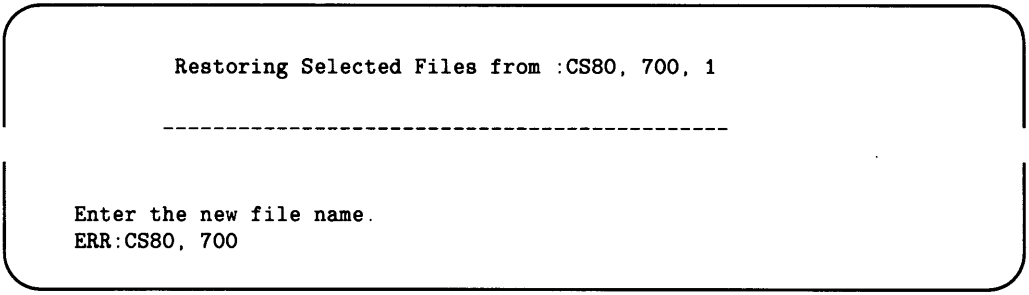
```
                        Restore Selected Files

             ----------------------------------------------

             The rename option allows you to change the
             destination and msvs for a group of files
             and allows you to change the name of each file
             that is restored, if desired.

             Type D or press the D softkey to allow changing
             the destination paths and msvs's only or type F
             or press the F softkey to allow changing both the
             destination paths and the individual file names;
             any other input will disable the option.


      Allow renaming of the files?
      NO
```

As with all previous examples, in the following list, you use the arrow keys (or softkeys for PREV or NEXT) to select the backup device and type Return to complete the restore.



```
                Restore Selected Files from ?

        Use the softkeys to select a mass storage device.
        -------------------------------------------------
          9153     HARD     :CS80, 700
       => 9153     Flexible :CS80, 700, 1
```

The utility copies the specified directories and/or files from the back-up media to the destination media while you see the following display in which the ERR:CS80,700 hypothetically indicates which file is being restored.

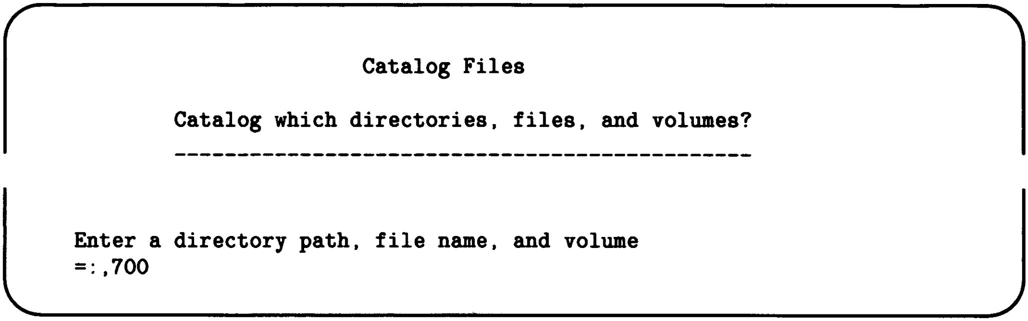


```
        Restoring Selected Files from :CS80, 700, 1

        ---------------------------------------------

    Enter the new file name.
    ERR:CS80, 700
```

If the back up media overflows, the utility prompts you to insert a different back-up disc or tape. When the back-up is complete, the utility returns to the main menu where you can select another option or exit the utility.

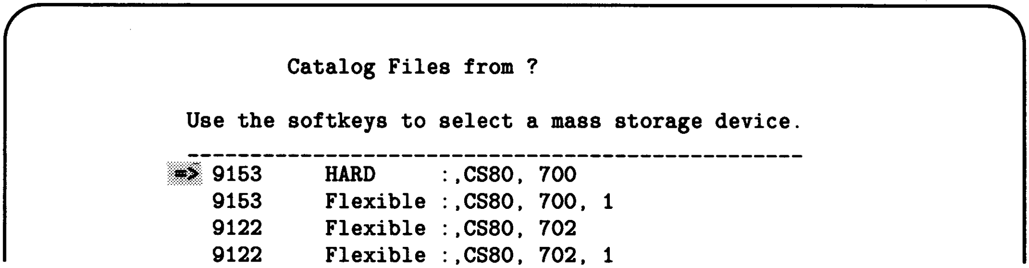
Catalog backup media Option

Selecting this option provides the following prompt for which you specify the files to catalog according to previous examples. You can specify any files as you did in previous examples along with any wildcards you want to use. Just typing [Return] accepts the `=:,700` and catalogs the current default disc. type the key for `Proceed` to indicate you have entered all of the back-up media you want to catalog.



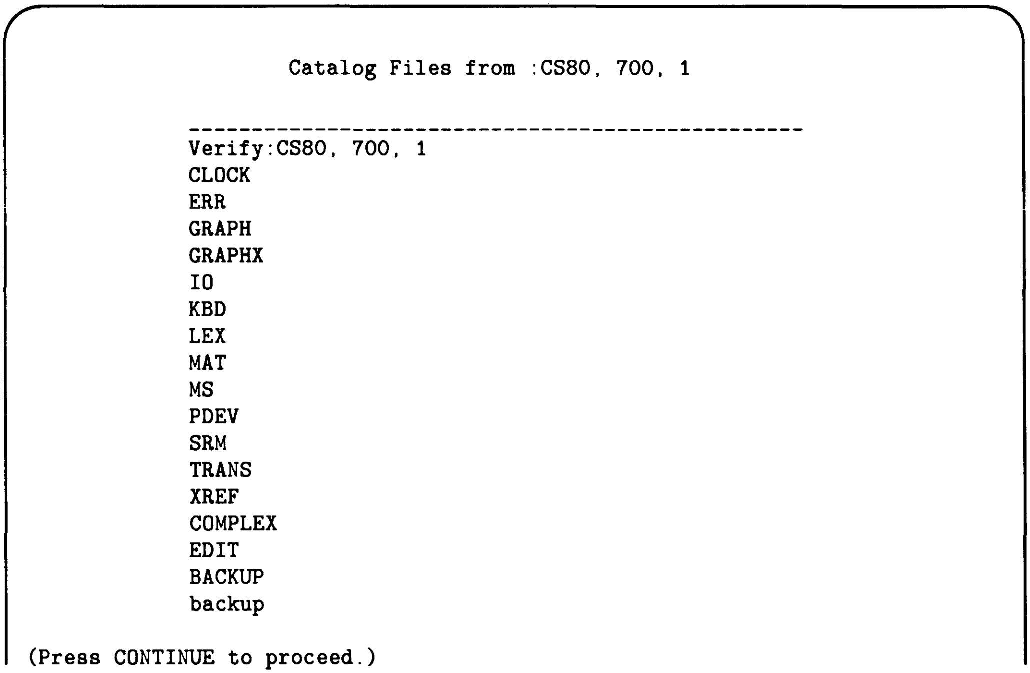
```
                         Catalog Files

            Catalog which directories, files, and volumes?
            ----------------------------------------------


     Enter a directory path, file name, and volume
     =:,700
```

The utility then prompts you for the media to be cataloged:



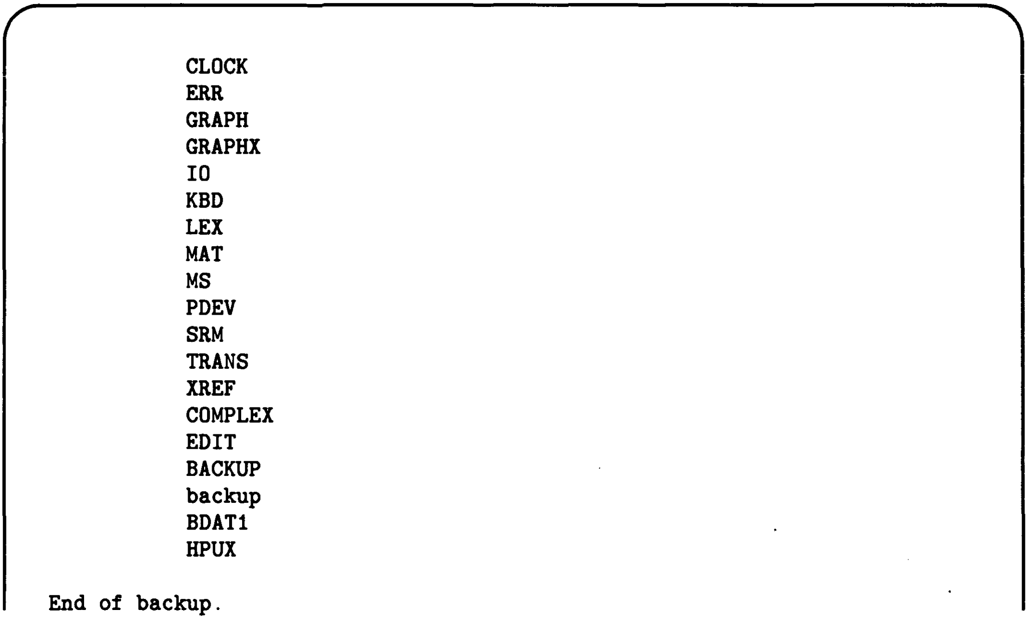
```
                   Catalog Files from ?

             Use the softkeys to select a mass storage device.
             -------------------------------------------------
            => 9153     HARD      :,CS80, 700
               9153     Flexible  :,CS80, 700, 1
               9122     Flexible  :,CS80, 702
               9122     Flexible  :,CS80, 702, 1
```

On selecting a media, the utility lists the files and directories on that media. (Note that only file names are listed, not directory paths, file types, time stamps, etc.) For example, you might see:



```
                       Catalog Files from :CS80, 700, 1

              ------------------------------------------------------
              Verify:CS80, 700, 1
              CLOCK
              ERR
              GRAPH
              GRAPHX
              IO
              KBD
              LEX
              MAT
              MS
              PDEV
              SRM
              TRANS
              XREF
              COMPLEX
              EDIT
              BACKUP
              backup

(Press CONTINUE to proceed.)
```

The bottom message indicates there is more than a "screenful" of files and directories on the specified back-up media. Type the key for CONTINUE to see the remaining files, noting that you type this key enough times to see all "screenfuls" of files. An additional list, which indicates that you have seen all the files, might look like this:



```
              CLOCK
              ERR
              GRAPH
              GRAPHX
              IO
              KBD
              LEX
              MAT
              MS
              PDEV
              SRM
              TRANS
              XREF
              COMPLEX
              EDIT
              BACKUP
              backup
              BDAT1
              HPUX

   End of backup.
```

Exit the BACKUP Utility Option

Choosing this option shows the following message, terminates the utility, and returns you to the ready state in BASIC.

```
     Utility terminated.
```

Although the utility has terminated, it is still in memory and you can run it again if you wish.

HFS Consistency Checks 15

The System Disc Utility (DISC_UTIL) on the *HFS Utilities* disc provides an option named "**Check Consistency of HFS Volume**" that lets you perform an Hierarchical File System Consistency Check (HFSCK). An HFSCK **examines** the structural integrity of a volume and **repairs** inconsistent HFS-formatted mass storage volumes. This section describes why, when, and how to run an HFSCK. The following table lists the sectins and page numbers.

Why Run an HFSCK?

HFS volumes can get into an inconsistent state—not irretrievable, but in need of some repair. This may happen, for instance: if an HFS disc is switched off during an I/O operation; or the system is powered down during a I/O statement that is accessing an HFS volume; or a program is executed in HP-UX which erroneously modifies an HFS volume. A hardware failure can corrupt a disc, as can inappropriate use of HP-UX from the same disc as you use BASIC. HFSCK is designed to perform this repair work (automatically, if possible; or with your help, if required).

How Often Should an HFSCK Be Run?

An HFSCK should be done on a **regular basis** if you are using HFS mass storage devices regularly. You should also do a check on getting the error message:

```
ERROR 180  HFS disc may be corrupt
```

For example, if you use the HP-UX Operating System when you are not running BASIC and you shutdown HP-UX improperly, you could get this error. The HP-UX *reboot* command sets the value of a certain byte to FS_CLEAN prior to shutdown, and if that byte has a different value during the subsequent boot process, BASIC will give the above error.

In addition, you should do a check when something unusual happens— such as the problems mentioned earlier. While there is no specific problem you can relate to file system corruption, **it is not a good idea to further modify a corrupted file system**.

An Essential Prerequisite

A directory named */lost+found* must exist in the root directory on the file system being examined **before** you do an HFS consistency check (do a CAT to check). This applies to hard **and** flexible discs. If you installed HP-UX and BASIC 5.0 on your hard disc, the directory probably exists on that hard disc; but it might not exist on other discs, especially flexible discs.

If there is no */lost+found* directory:

1. someone has purged it; or
2. the disc is not an HFS disc.

In any such case, to create the directory on the default msvs, type:

```
CREATE DIR "/lost+found" Return
```

To create the directory on a specified volume, replace *msvs* with an appropriate volume specifier (e.g. 702) and type:

```
CREATE DIR "/lost+found:,msvs" Return
```

An HFS Consistency Check places any "problem" files in this directory. Orphaned files and directories (allocated but unreferenced) are, with the user's concurrence, reconnected by placing them in the */lost+found* directory. The inode number is the assigned name.

Later, after you run the check, examine the files in the directory by executing the following statement, substituting an appropriate volume specifier such as `700` or `702,1` for *msvs*:

```
CAT "/lost+found:,msvs" Return
```

If you find any files that are out of place or not needed, copy them to where they belong or delete them.

Clear out the directory before you run another HFS consistency check. To create empty slots in */lost+found*, type the following statement, using an appropriate volume specifier for *msvs*:

```
MSI "/lost+found:,msvs" Return
```

and then type and RUN the following program:

```
10 FOR I = 1 TO 254      ! Program to create empty slots in /lost+found
20 A$ = "A" & VAL$(I)
30 CREATE ASCII A$,1     ! Create ASCII files
40 NEXT I
50 FOR I = 1 TO 254
60 A$ = "A" & VAL$(I)
70 PURGE A$              ! Purge the files
80 NEXT I
90 END
```

How to Do an HFSCK

If you have never done an HFSCK (Hierarchial File System Consistency Check), you might want to skim through the entire section. Then, come back here and work through the check.

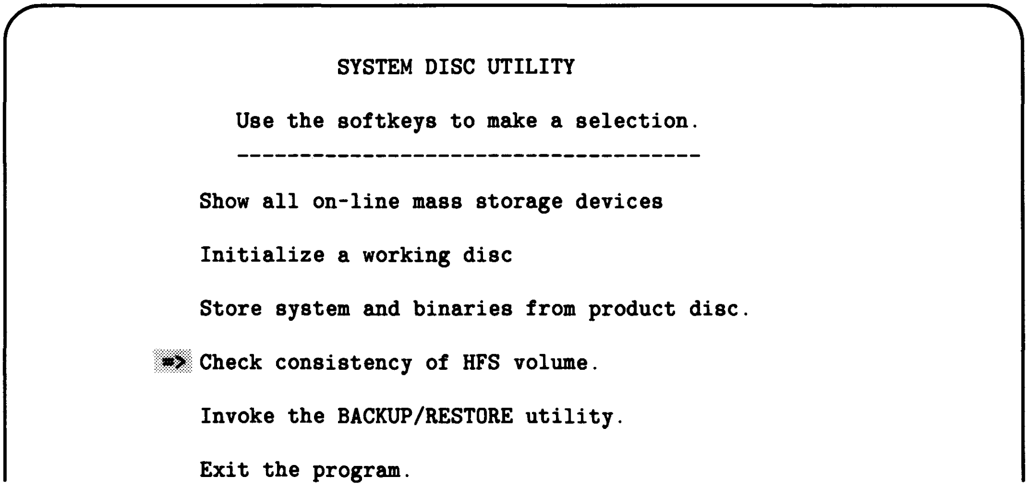
Have the BASIC system running. Load the System Disc Utility by inserting the *HFS Utilities* disc into your flexible disc drive. Replace *msvs* with the appropriate volume specifier (e.g. `1400,1` or `700` or`702,1`) and type:

```
LOAD "DISC_UTIL:,msvs" Return
```

After the LOAD operation is complete, get the `SYSTEM DISC UTILITY` menu by typing:

```
RUN Return
```

You need to see:



```
                    SYSTEM DISC UTILITY

            Use the softkeys to make a selection.
            -------------------------------------

          Show all on-line mass storage devices

          Initialize a working disc

          Store system and binaries from product disc.

       => Check consistency of HFS volume.

          Invoke the BACKUP/RESTORE utility.

          Exit the program.
```

Choosing the "Check Consistency" Option

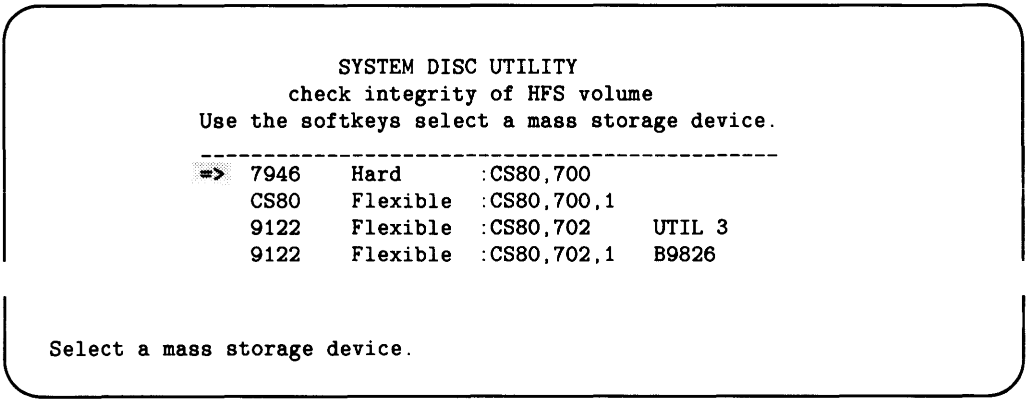
Note the pointer, `=>`. You can move it with the down/up arrow keys or with the softkeys for `Next` or `Prev`. To select an option, which will initiate an HFSCK in this case, move the pointer to:

```
Check consistency of HFS volume.
```

and type Return

Choosing the Volume to Check

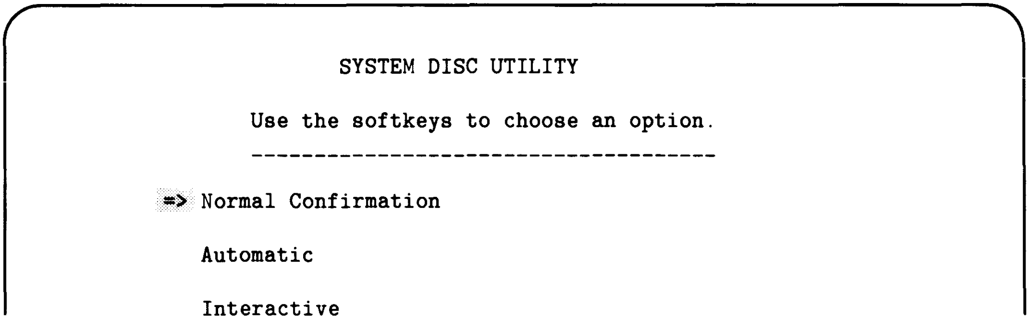
You get the following menu which lets you select the volume for which you want to do an HFS consistency check. Move the pointer to the desired volume and type Return.



```
                    SYSTEM DISC UTILITY
                  check integrity of HFS volume
              Use the softkeys select a mass storage device.

              ----------------------------------------------
              => 7946    Hard      :CS80,700
                 CS80    Flexible  :CS80,700,1
                 9122    Flexible  :CS80,702     UTIL 3
                 9122    Flexible  :CS80,702,1   B9826


  Select a mass storage device.
```

Choosing the Mode

The following menu lets you select the mode to be used in running an HFS consistency check. Note the modes and read the descriptions of them on the next page **before** you select a mode.



```
                    SYSTEM DISC UTILITY

                Use the softkeys to choose an option.
                -------------------------------------

            => Normal Confirmation

               Automatic

               Interactive
```

Normal Confirmation

This mode attempts to repair the disc by automatically fixing the unambiguous problems and by asking for confirmation with other problems. It is the **recommended mode**. In this mode, some of the questions are automatically answered with a "yes". The mode attempts to repair an inconsistency, if possible, and counts in the superblock and cylinder groups are automatically fixed.

Automatic

This mode fixes the following inconsistencies, but never removes any data. For things it can fix, you see a message that identifies the corrective action taken.

- Unreferenced inodes
- Missing blocks in the free list
- Blocks in the free list also in files
- Wrong counts in the superblock

Other problems cause the check to terminate; and to fix the problem you must do another HFSCK in either Normal or Interactive mode.

Interactive

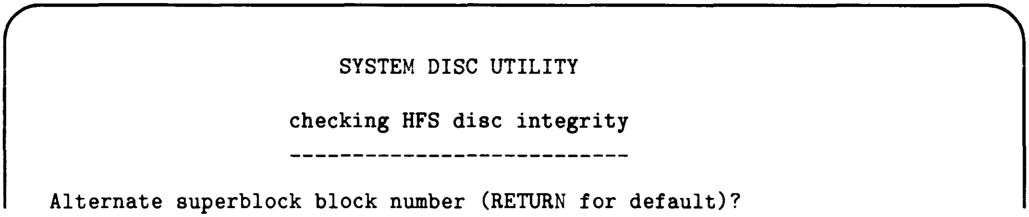
This mode lets you interact with the system, and in doing this, you make decisions about options which you should understand before you make the decisions. **Unless you have a clear understanding of how the file system operates, or unless you ran a check in Normal or Automatic mode and were told to run it in Interactive mode, do not select this mode**.

If you **do** select interactive mode, **and** you are asked during the check to make a desicion about how to handle an inconsistency, **and** you need a working understanding of the decision you need to make; **each** description of inconsistencies in the "Consistency Check Confirmation Requests" section points you to and appropriate part of the "File System Implementation" chapter. This way, you can get information when you need it.

With these things in mind, select the mode you want to initiate an HFS consistency check. What happens during each phase is described later in the "Consistency Check Confirmation Requests" section; and the next section describes the display that appears when you select a mode.

Specifying an Alternate Superblock

On selecting a mode and before the check begins, you see the following display. Type Return to continue, noting that the bottom prompt appears only in Interactive mode. Type an appropriate block number for this prompt only if you are an expert user and know what you are doing. (You need to have noted the superblock numbers when you formatted the HFS volume to be able to enter a correct number. You can always enter 16, but that superblock may also be corrupted.)



```
                           SYSTEM DISC UTILITY

                        checking HFS disc integrity
                        ---------------------------

  Alternate superblock block number (RETURN for default)?
```

Phases of the Consistency Check

Each phase of the utility invokes a different file system pass. Successive phases check blocks and sizes, directory path names, connectivity, reference counts, and the free-block map.

What If the Check Was Successful?

If the file system is in good order **and** you did not use Automatic mode, you can **expect to see** something like the following display. Note that the number of files in the file system and the number of used and free blocks appear in the last line.

```
                           SYSTEM DISC UTILITY

                        ---------------------------

   Alternate superblock block number (RETURN for default)?
   ** Phase 1 - Check Blocks and Sizes
   ** Phase 2 - Check Pathnames
   ** Phase 3 - Check Connectivity
   ** Phase 4 - Check Reference Counts
   ** Phase 5 - Check Cyl groups
   1407 files, 11320 used, 29823 free (247 frags, 3697 blocks)
```

If you use Automatic mode, expect to see the following display:

```
                           SYSTEM DISC UTILITY

                        ---------------------------

   :CS80, 700: 1407 files, 11320 used, 29823 free (247 frags, 3697 blocks)
```

At this point, continuing returns you to the main menu of options.

Determining Free (Unused) Disc Space

Whatever the mode, the final line containing numbers tells you about disc space. In the case at hand, 29 823 times the fragment size (1024 bytes if you use the BASIC 5.0 MKHFS to create the file system) is your amount of free space; this is 29 823 * 1024 = 30 538 752 bytes. A percentage of this space called the "free space threshold" is reserved. If the free space threshold is 2%, then 29 927 976 of the available 30 538 752 bytes can be used.

Consistency Check Confirmation Requests

You do not need to read this section unless, during an HFSCK, you get an inconsistency.

An HFS consistency check occurs in phases; and according to these phases, this section examines:

- The possible confirmation requests that can arise due to inconsistencies during a phase of the checking process;
- The available responses;
- The consequences of each response; and
- The documentation to read to get additional information, if necessary.

If you choose **Normal Confirmation** mode, some of the following questions are automatically answered with "YES", and the utility then attempts to repair the inconsistency. **If you choose Interactive mode and you give an answer other than "YES" to a confirmation request, the utility might not be able to repair the file system**. Unless you have a clear understanding of how the file system operates, or you were told to use Interactive mode, do not select Interactive mode.

There are some inconsistencies that can occur during an initialization phase or that can occur in more than one phase. These inconsistencies are discussed in the later section called "Initialization Phase Errors".

If you have an inconsistency you want to fix and you do not have sufficient information to deal with the response, see the documentation suggested with the inconsistency in the later discussions of phases.

Automatic Yes for Normal Confirmation

When you examine an error message and see a [Y] at the beginning of a message, the [Y] means that the question is automatically answered "yes" in Normal Confirmation mode. For example:

```
[Y] BAD DIRECT ADDRESS, SHOULD BE ZERO: inode.di_db[xxx] = yyy
CORRECT?
```

will automatically be answered "Yes" in Normal Confirmation and Automatic modes, and you do not need to do anything.

Initialization Phase Errors

Before the file system check is performed, there is an initialization phase in which tables are set up and certain files are opened. You can get any of several error conditions during this and some other phases. All of them are fatal when you choose Automatic mode, and they can be fatal in other modes.

In general, this phase finds error conditions resulting from memory requests, opening of files, status of files, file system size checks, and creation of the scratch file. The the next several subsections discuss these error conditions.

CANNOT ALOC NNN BYTES FOR "XXX"

XXX is either *blockmap*, *freemap*, *statemap*, or *lncntp*; and this means that HFSCK's request for memory has failed. The check terminates.

This should not happen, but if it does, you should contact your appropriate HP Sales and Service Office for assistance.

* Can't open ...

The listed file system cannot be opened. Check for missing drivers.

Other Messages:

In this category, you can get any of:

```
MAGIC NUMBER WRONG
NCG OUT OF RANGE
CPG OUT OF RANGE
NCYL DOES NOT GIVE WITH NCG*CPG
SIZE PREPOSTEROUSLY LARGE
TRASHED VALUES IN SUPER BLOCK
```

and the following message:

file_system: `BAD SUPERBLOCK`: *superblock_address*

Such an error condition means the superblock has been corrupted. An alternate superblock must be used. The check terminates.

Use the Interactive Mode of HFSCK to specify the location of an alternate superblock to supply the needed information. You can specify 16 or any other superblock number you recorded during formatting of the HFS volume.

* CAN NOT SEEK: BLK "bn" (CONTINUE)

HFSCK's request for moving the the specified block number in the file system failed. This should not happen, and you should contact your local HP Sales and Service Office for assistance.

You can attempt to continue, and the possible responses to the CONTINUE prompt are:

YES Attempt to continue to run the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. A second run of HFSCK should be made to re-check the file system.

NO Terminate the program.

* CAN NOT READ:BLK ... (CONTINUE)

HFSCK's attempt to read a specified block number in the file system failed. This can happen when you interrupt a check before it finishes. Contact your local HP Sales and Service Office for assistance.

You can attempt to continue, and the possible responses to the CONTINUE prompt are:

YES Attempt to continue to run the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. A second run of HFSCK should be made to re-check the file system.

NO Terminate the program.

* CAN NOT WRITE: BLK ... (CONTINUE)

HFSCK's attempt to write a specified block number in the file system failed. The disc is probably physically write-protected. Remove the write protection from the disc and re-run HFSCK. The possible responses to the CONTINUE prompt are:

YES Attempt to continue to run the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. A second run of HFSCK should be made to re-check the file system.

NO Terminate the program.

Phase 1 - Check Blocks and Sizes

This phase checks the block numbers and file size in each inode. The block numbers must be sensible and agree with the size. No block may be in more than one file. The messages appear on a line, followed by an indented explanation and your alternatives for dealing with them.

```
CG>> :BAD MAGIC NUMBER
```

The magic number of cylinder group is wrong, which usually indicates that the cylinder-group maps have been destroyed. The check is terminated in Automatic mode, and you should run the check again in another mode. If you are in Normal or Interactive modes, the cylinder group is marked as needing to be reconstructed during a later phase, and the check does not die at this point.

```
HOLD BAD BLOCK?
```

An inode has an illegal type field of the sort created by the UNIX[1] (but not HP-UX) *badblk* utility.

Y : Convert this to a regular file containing the one bad block.
N : Clear the inode.

For information, see "Bad Blocks" in "File System Implementation".

```
[Y] BAD DIRECT ADDRESS, SHOULD BE ZERO: inode.di_db[xxx] = yyy
CORRECT?
```

An inode has a direct address beyond the limit imposed by the file size. Recommend "Y".

Y : Zero the address.
N : Leave it as it is.

For information, see "Inodes" and "Indirect Blocks" in "File System Implementation".

```
BAD INDIRECT ADDRESS: IND BLOCK xxx[yyy] = zzz
CORRECT?
```

An inode has an indirect address beyond the limit imposed by the file size. Recommend "Y".

Y: Zero the address.
N: Leave it as it is.

For information, see "Indirect Blocks" in "File System Implementation".

```
COULD NOT CHECK INDIRECT BLOCKS
CLEAR?
```

HFSCK could not check the inode's indirect blocks. Recommend "N". (Rose: some help here.).

Y: Clear the inode.
N: Leave it as is.

For information, see "Indirect Blocks" in "File System Implementation".

```
LINK TABLE OVERFLOW.
CONTINUE?
```

There are more than 500 files with negative or zero link count. Recommend "Y".

Y: Keep going. HFSCK may be unable to produce a clean file system,
so you should run it again.
N: Exit HFSCK.

For information, see "Link Count" in "File System Implementation".

```
[Y] INCORRECT BLOCK COUNT I=xxx (yyy should be zzz)
CORRECT?
```

The block count in inode xxx is yyy instead of the zzz required by the inode size. Recommend "Y".

Y: Change the block count to agree with the size.
N: Leave it alone.

For information, see sections after "Link Count" in "File System Implementation".

```
UNKNOWN FILE TYPE I=xxx ...
CLEAR?
```

HFSCK does not understand the inode contents. This arises in several situations:

- the inode type is not recognizable.
- the size is negative.
- there are indirect disc addresses beyond the limit implied by the size.

Recommend "Y" in all cases.

Y: Clear the inode.
N: Leave it alone.

For information, see "Inodes" in "File System Implementation".

```
PARTIALLY ALLOCATED INODE I=xxx
CLEAR?
```

An inode marked unallocated has some disk addresses. Recommend "Y".

Y: Clear the inode.
N: Leave it alone.

For information, see "Format and Type" in "File System Implementation".

```
xxx BAD I=yyy
```

Inode yyy contains an unreasonable disk address xxx. The address is too large for the file system, or indicates a block in an area where data blocks shouldn't be. There is no confirmation; this is just an error report.

```
EXCESSIVE BAD BLKS I=xxx
CONTINUE?
```

There are more than 10 bad block numbers in the file. Recommend "Y".

Y: Skip this pass. HFSCK may be unable to produce a clean file system, so run it again.
N: Exit HFSCK.

For information, see "Bad Blocks" in "File System Implementation".

```
xxx DUP I=yyy
```

Inode yyy contains a disk address xxx found in another inode. There is no confirmation; this is just an error report.

```
EXCESSIVE DUP BLKS I=xxx
CONTINUE?
```

There are more than 10 duplicate block numbers in the file. Recommend "Y".

Y: Skip this pass. HFSCK may be unable to produce a clean file system, so run it again.
N: Exit HFSCK.

For information, see "Duplicate Blocks" in "File System Implementation".

```
DUP TABLE OVERFLOW.
CONTINUE?
```

There are more than 100 duplicated blocks. Recommend "Y".

Y: Keep going. HFSCK may be unable to produce a clean file system, so run it again.
N: Exit HFSCK.

For information, see "Duplicate Blocks" in "File System Implementation".

Phase 1b - Rescan for more DUPS

This phase only occurs if phase 1 found some duplicate blocks. This will result in some more "xxx DUP I=yyy" messages being output. HFSCK must rescan because when it discovers that a block is duplicated, it has forgotten which inode contained the first reference to the block.

Phase 2 - Check Pathnames

HFSCK traverses the directory tree, checking the directories for consistent entries, presence of '.' and '..', and possibly bad inumbers. In this phase, all errors are fatal when you choose Automatic mode.

```
ROOT INODE NOT DIRECTORY
FIX?
```

The inode for "/" is not a directory. Recommend "Y".

Y: Change its type and try to interpret the contents as a directory.
N: Exit HFSCK.

For information, see "Inodes" and "Data Blocks" in "File System Implementation".

```
DUPS/BAD IN ROOT INODE
CONTINUE?
```

The inode for "/" contains bad or duplicated blocks. Recommend "Y".

Y: Try to keep going.
N: Exit HFSCK.

For information, see "Duplicate Blocks" in "File System Implementation".

```
ZERO LENGTH DIRECTORY I=xxx OWNER=yyy ...
REMOVE?
```

The directory with inode xxx has size 0. Recommend "Y".

Y: Clear the inode.
N: Leave it alone.

For information, see "Inode Size" in "File System Implementation".

```
DIRECTORY TOO SHORT I=xxx OWNER=yyy ...
FIX?
```

The directory size is less than the minimum (big enough to hold '.' and '..'). Recommend "Y".

Y: Change the size to the minimum.
N: Leave it alone.

For information, see "Inode Size" and "Data Blocks" in "File System Implementation".

```
DIRECTORY CORRUPTED I=xxx OWNER=yyy ...
FIX?
```

An entry for the directory with inode xxx is not consistent, or the directory size is not a multiple of the size of a directory entry. Recommend “Y”.

Y: Correct the entry.
N: Leave it alone.

For information, see “Inode Size” in “File System Implementation”.

```
BAD INODE NUMBER FOR '.' I=xxx OWNER=yyy ...
FIX?
```

The entry for '.' does not contain the inumber of the directory. Recommend “Y”.

Y: Correct the entry.
N: Leave it alone.

For information, see “Data Blocks” in “File System Implementation”.

```
MISSING '.' I=xxx OWNER=yyy ...
FIX?
```

There is no entry for '.' in the directory. Recommend “Y”.

Y: Add entry for '.'.
N: Leave it alone.

For information, see “Data Blocks” in “File System Implementation”.

```
BAD INODE NUMBER FOR '..' I=xxx OWNER=yyy ...
FIX?
```

The inumber for the '..' entry is not the same as the inumber of the parent of xxx. Recommend “Y”.

Y: Correct the entry.
N: Leave it alone.

For information, see “Data Blocks” in “File System Implementation”.

```
MISSING '..' I=xxx OWNER=yyy ...
FIX?
```

There is no entry for '..'.

Y: Add entry for '..'.
N: Leave it alone.

For information, see "Data Blocks" in "File System Implementation".

```
EXTRA '.' ENTRY I=xxx OWNER=yyy ...
FIX?
```

The directory has more than one entry for '.'. Recommend "Y".

Y: Remove all but the first entry for '.'.
N: Leave it alone.

For information, see "Data Blocks" in "File System Implementation".

```
EXTRA '..' ENTRY I=xxx OWNER=yyy ...
FIX?
```

The directory has more than one entry for '..'. Recommend "Y".

Y: Remove all but the first entry for '..'.
N: Leave it alone.

For information, see "Data Blocks" in "File System Implementation".

```
I OUT OF RANGE I=xxx OWNER=yyy ...
REMOVE?
```

An entry in the directory has an impossible inumber. Recommend "Y".

Y: Remove the entry.
N: Leave it alone.

For information, see "Inodes" in "File System Implementation".

```
UNALLOCATED I=xxx OWNER=yyy ...
REMOVE?
```

An entry in the directory has an inumber for an inode that is unallocated. Recommend "Y".

Y: Remove the entry.
N: Leave it alone.

For information, see "Format and Type" in "File System Implementation".

```
DUP/BAD I=xxx OWNER=yyy ...
REMOVE?
```

An entry in the directory has an inumber for an inode with duplicate or bad blocks. Recommend "N".

Y: Remove the entry. This can be very dangerous! The inode
    may have duplicate blocks (same block as in another file),
    but it may be the OTHER inode that is bad, not this one.
N: Leave it alone.

For information, see "Duplicate Blocks" and "Bad Blocks" in "File System Implementation".

Phase 3 - Check Connectivity

This phase finds directory trees not connected to "/", and reconnects them to */lost+found.*

```
[Y] UNREF DIR I=xxx OWNER=yyy ...
RECONNECT?
```

The directory is not connected to "/". Recommend "Y".

Y: Reconnect it to */lost+found* if possible. Note that HFSCK will
not automatically increase the size of the */lost+found* directory.
It will only insert the orphan directory if there is an empty slot.
N: Leave it alone.

For information, see "Link Count" in "File System Implementation".

Phase 4 - Check Reference Counts

This phase checks the link counts on all inodes and adjusts them if necessary. If an inode is unreferenced, it clears it or links it into */lost+found.*

```
[Y] UNREF FILE I=xxx OWNER=xxx ...
[Y] UNREF PIPE I=xxx OWNER=xxx ...
RECONNECT?
```

The file has no references in any directories. Recommend "Y".

Y: Reconnect it to */lost+found* if possible. Note that HFSCK will
not automatically increase the size of the */lost+found* directory.
It will only insert an orphan file if there is an empty slot.
N: Leave it alone.

For information, see "Link Count" in "File System Implementation".

```
[Y] UNREF DIR I=xxx OWNER=yyy ...
[Y] UNREF FILE I=xxx OWNER=yyy ...
[Y] UNREF PIPE I=xxx OWNER=yyy ...
[Y] BAD/DUP DIR I=xxx OWNER=yyy ...
[Y] BAD/DUP FILE I=xxx OWNER=yyy ...
[Y] BAD/DUP PIPE I=xxx OWNER=yyy ...
CLEAR?
```

The inode is not referenced, and it is not a regular file, otherwise it has size 0 **or** you did not want to reconnect it into */lost+found* **or** there is no room in */lost+found.* Recommend "Y".

Y: Clear the inode.
N: Leave it alone.

For information, see "Link Count" in "File System Implementation".

```
[Y] LINK COUNT FILE I=xxx OWNER=yyy ...
[Y] LINK COUNT DIR I=xxx OWNER=yyy ...
[Y] LINK COUNT filename I=xxx OWNER=yyy ...
COUNT a SHOULD BE b
ADJUST?
```

The link count in the inode is incorrect. Recommend "Y".

Y: Correct it.
N: Leave it alone.

For information, see "Link Count" in "File System Implementation".

```
[Y] FREE INODE COUNT WRONG IN SUPERBLOCK
FIX?
```

The free inode count is wrong. Recommend "Y".

Y: Fix it.
N: Leave it alone.

For information, see "Free-Block Checking" and "Link Count" in "File System Implementation".

Phase 5 - Check Cyl groups

This phase examines the cylinder groups and corrects bad information, if found.

```
[Y] FREE BLK COUNT(S) WRONG IN SUPERBLOCK
FIX?
```

The free block totals in the superblock are wrong. Recommend "Y".

Y: Fix it.
N: Leave it alone.

For information, see "Super Block Consistency" and "Free-Block Checking" in "File System Implementation".

```
[Y] BAD CYLINDER GROUPS
FIX?
```

The cylinder group information does not match reality. Recommend "Y".

Y: Enter Phase 6.
N: Leave them alone.

For information, see "Updating the HFS File System" and "Bad Blocks" in "File System Implementation".

```
EXCESSIVE BAD BLKS IN BIT MAPS
CONTINUE?
```

There are too many bad blocks in the cylinder group bit maps. Recommend "Y".

Y: Enter Phase 6.
N: Exit HFSCK.

For information, see "Allocation of Disc Space" and "Bad Blocks" in "File System Implementation".

Phase 6 - Salvage Cylinder Groups

This phase reconstructs the cylinder group information.

Post Phases Cleanup

After a file system has been checked, and possibly repaired, some cleanup functions are performed. You can get some messages at this time and no action is required on your part.

You might get the message:

```
        ***** FILE SYSTEM WAS MODIFIED *****
```

which simply indicates what was done.

In addition, you will get the message:

```
    f files, b used, r free (y frags z blocks)
```

This message was explained earlier in the section called "Phases of the Consistency Check".

HFS File System Implementation 16

The earlier chapter called "Mass Storage Concepts" described the file system at the user level. This chapter extends that information and provides information about the implementation of a file system that is unique to a HFS (Hierarchial File System). Among other things, this information is useful when you create or check an HFS file system.

Before you jump into the details, it is helpful in understanding the HFS file system implementation to remember that:

- Block size is 8 Kbytes (8192 bytes);
- Fragment size is 1 Kbyte (1024 bytes);
- Cylinders/group is 16;
- Percentage of free space is 10%;
- Swap space is 0; and
- Bytes per inode is 2 Kbytes (2048 bytes).

These default values are mentioned in many places, and you will perform tasks that use them.

For your convenience in finding information, the following table shows the sections and their page numbers.

Disc Layout

Each disc used for a file system begins with an 8 Kbyte volume header area. The rest of the disc holds the file system.

Unlike an HP-UX file system, **all** file systems created with the BASIC 5.0 MKHFS CSUB utility (used by the "Initialize a working disc" option on the *HFS Utilities* disc) have zero (0) swap space. Each file system begins with the primary copy of the superblock and consists of one or more cylinder groups (see Figure 16-1). If you have a disc that supports "hard" partitions (e.g. the HP9133H), then you can address each partition separately (the volume bit in the minor address indicates the partition) and create a file system for that partition.

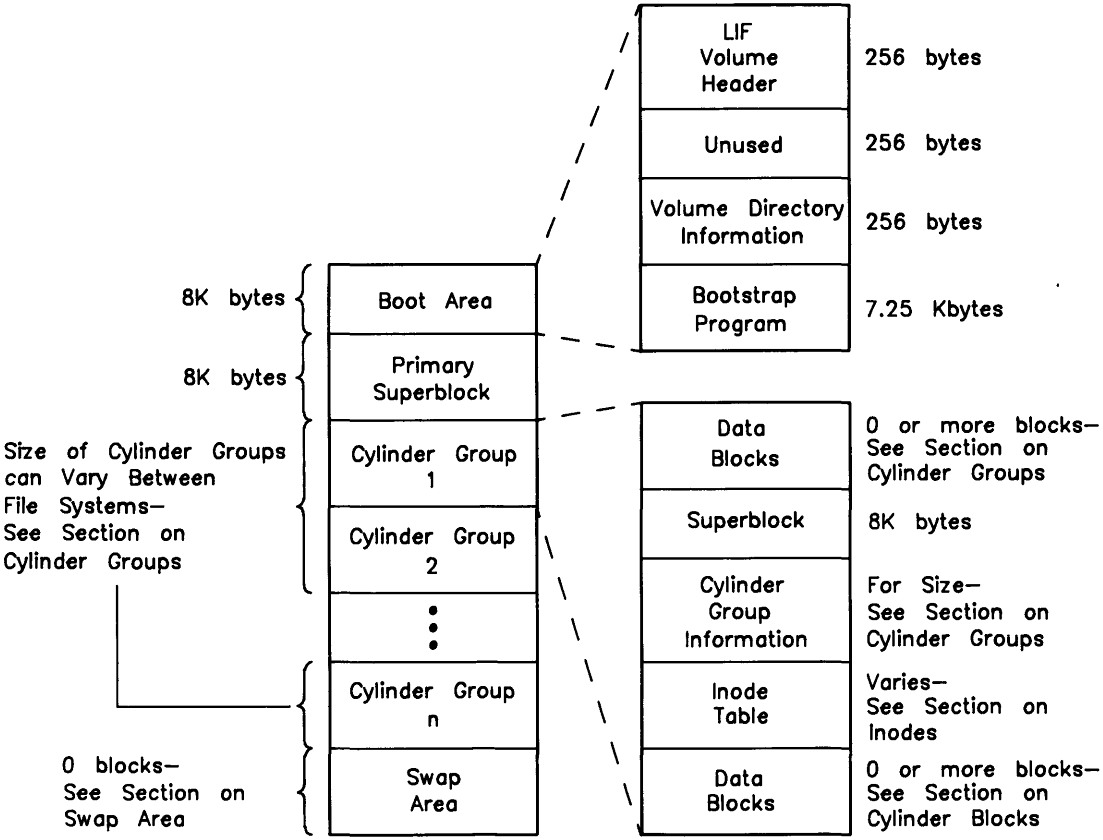

LIF Volume Header
256 bytes
Unused
256 bytes
Volume Directory Information
256 bytes
8K bytes
Boot Area
Bootstrap Program
7.25 Kbytes
8K bytes
Primary Superblock
Size of Cylinder Groups can Vary Between File Systems— See Section on Cylinder Groups
Cylinder Group 1
Data Blocks
0 or more blocks— See Section on Cylinder Groups
Cylinder Group 2
Superblock
8K bytes
Cylinder Group Information
For Size— See Section on Cylinder Groups
Cylinder Group n
Inode Table
Varies— See Section on Inodes
0 blocks— See Section on Swap Area
Swap Area
Data Blocks
0 or more blocks— See Section on Cylinder Blocks

Figure 16-1: File System Layout on Disc

Boot Area

A **boot area** is automatically reserved on a volume when you create a file system on a hard or flexible disc. Information in the boot area is used only if the volume is used for booting (boot disc), but the space is reserved on all volumes. The boot area resides on the first 8K bytes of a volume, and contains the volume header, volume directory information, and a bootstrap program used when an operating system is loaded. The boot area is reserved exclusively for use by the boot ROM.

Each boot disc must have a **volume header** in the boot area to identify the volume format. The format of the boot area is Logical Interchange Format (LIF). The volume header is checked by the boot ROM in its examination of bootable mass storage media when the computer is powered up. The **Volume Directory Information** contains no names. It is empty until the user does a `STORE SYSTEM` to the disc.

The latter several Kbytes on the boot area contain the **bootstrap program**. The boot ROM loads and passes control to the bootstrap program which in turn loads and passes control to the system to be booted.

Cylinder Groups

Each cylinder group contains a copy of the superblock, a cylinder group information structure, an inode table, and data blocks. The superblock, cylinder group information, and inode table are located in each cylinder group at varying offsets so that any single track, cylinder, or platter can be lost without losing all copies of the superblock. If a superblock is lost, the file system can be repaired by using HFSCK with an alternate superblock. Any extra space before or after the superblock, cylinder group information, and inode table is filled with data blocks. Since BASIC 5.0 does not have an equivalent to the HP-UX *fsdb* command, and if major reconstruction in necessary, the user should call the appropriate HP support person. If one is available, the reconstruction can be done in the HP-UX operating system by an HP-UX expert (see the last subsection called "Uncorrectable File System Corruption to see recommendations).

There is a primary **superblock** at the beginning of a file system, and a copy of the superblock is in each cylinder group. The superblock contains static information known at file system creation: block size, fragment size, and disc characteristics. The primary superblock also keeps track of file system update information in its **summary information** area. In the HFS file system, 8 Kbytes are reserved for each copy of the superblock.

The **cylinder group information** contains the following dynamic parameters of the cylinder group:

- Number of inodes and data blocks.
- Pointers to the last used block, fragment, and inode.
- Number of available fragments.
- Used inode map.
- Free block map.

A bit map in the cylinder group information keeps track of available data blocks and fragments. Data blocks are divided into 1 Kbyte fragments. Data block and fragment allocation are described in the later subsection called “Data Storage”.

The size of the cylinder group information data structure is between 1 fragment and 1 block. The size depends on the number of data blocks per cylinder group.

The **inode table** contains per-file information (see Figure 16-2). A static number of inodes is allocated for each cylinder group when the file system is created. HFS uses a default such that there are more inodes per cylinder group than will be needed for average usage.

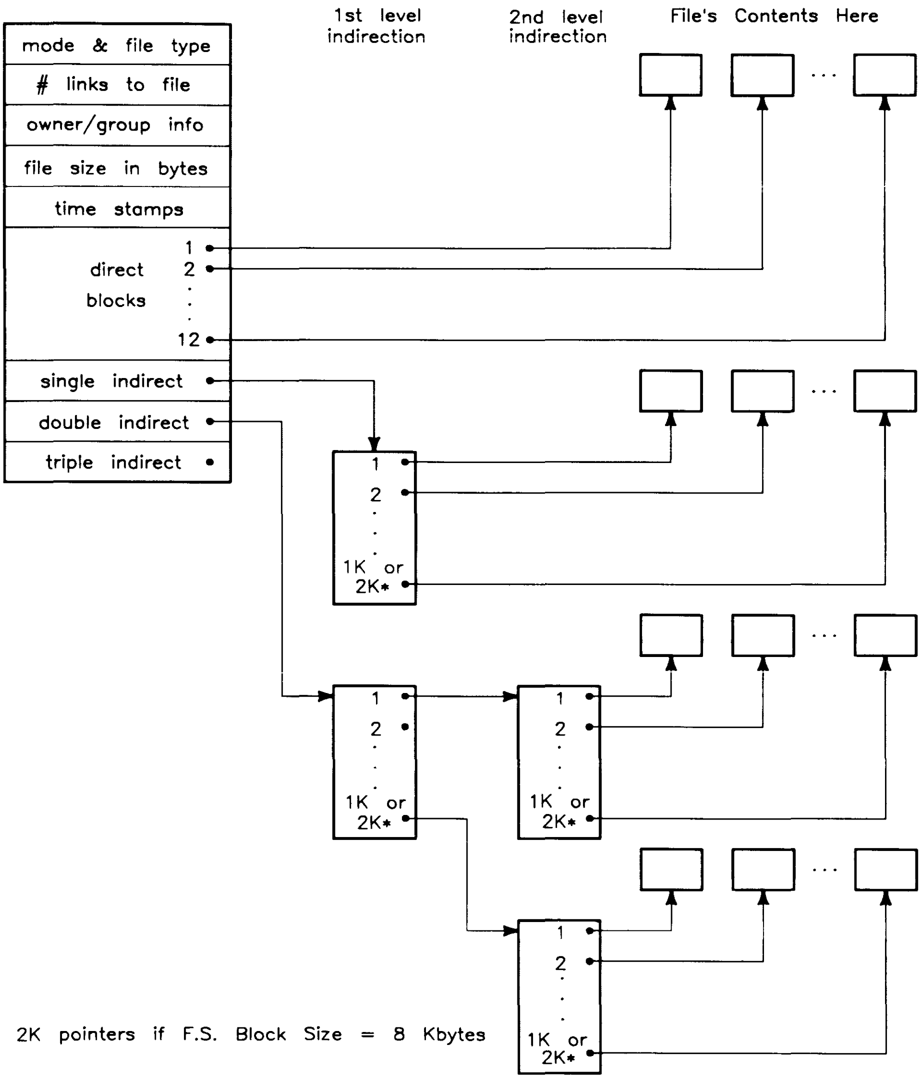


Figure 16-2: Regular File Mapping Scheme and the Inode Structure

A BASIC 5.0 HFS file system uses blocks of 8 Kbytes, and for the rest of the discussion on inode pointers the size of a block is referred to as *fs_bsize*. You can replace this variable with 8 Kbytes (8192 bytes) whenever necessary.

The first 12 pointers in an inode point directly to the first 12 blocks or fragments containing the file's data. If a file is larger than 12 blocks (greater than 12 × *fs_bsize*), indirect reference is made to the file's data. A group of indirect pointers is contained in one data block. Each pointer is 4 bytes long, so there can be 2048 pointers (8192/4) in each block of indirect pointers.

The 13th block address points to a block containing 2048 additional pointers to data blocks (from now on the number of indirect pointers in a block will be called *num_ip*). Thus, the 13th (single indirect) block address handles files up to 16 875 520 bytes in an 8 Kbyte block file system ($fs\_bsize \times (12 + num\_ip)$) . If the file is larger than this, the 14th inode block address points to *num_ip* indirect blocks, each of which contains pointers to an additional *num_ip* actual data blocks. If the file cannot be contained in this space, the 15th inode block address points to *num_ip* double-indirect blocks. With the 15th (triple-indirect) block address, the size of a file is limited by the size of your disc drive ($fs\_bsize \times (12 + num\_ip + num\_ip^2 + num\_ip^3)$). Your disc drive probably doesn't have this much space and files cannot cross disc drive boundaries. The point is that an HFS volume can handle a large file.

Inode pointers hold the address of a fragment. The address can be interpreted as referencing a whole block or as referencing 1 or more fragments, depending on the number of bytes stored at the address. Whether a block or a fragment is used depends on the following information in the inode: the file size, file system block size, and the pointer's index number. A partial block (1 or more fragments) is allocated only at the end of a file; so if there are 3 pointers to data, pointers 1 and 2 point to full blocks; and pointer 3 can point to a partial block.

Figure 16-3 shows an example of a 20 Kbyte file stored in 8 Kbyte blocks with 1 Kbyte fragments.

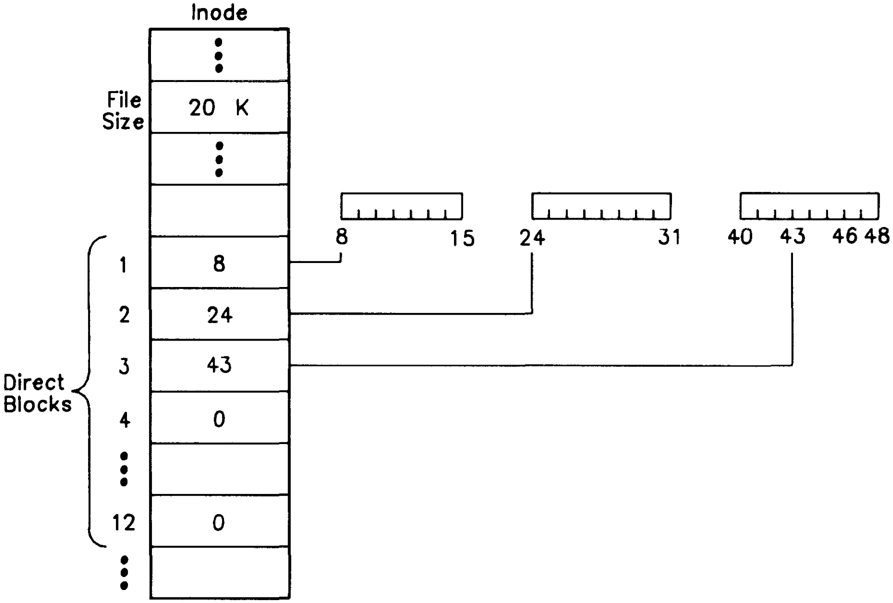


Figure 16-3: Inode Addressing Example

The number of blocks needed is 20 ÷ 8 (file size ÷ block size). This is 2 full blocks with a remainder of 4 fragments. Therefore, the first and second pointers point to full blocks, and the third pointer points to the remaining 4 fragments.

All indirect blocks are referenced only as full blocks; no pieces of the file are addressed at the fragment level beyond the 12 direct pointers. If the file described by the inode is a directory, then the pointers point to regular file system data blocks, but the blocks contain specifically formatted data.

The following information lets you determine the amount of space used by the inode table: number of bytes per cylinder group, average number of data bytes per inode; inode size (always 128); and block size. In a file system with 8 Kbyte blocks, 2 Mbyte cylinder groups, and 2048 data bytes per inode; there are 1000 inodes per cylinder group (2 Mbytes ÷ 2048 bytes). The 1000 inodes × 128 bytes per inode gives 128 000 bytes for the entire inode table **and** 128 000 ÷ 8192 (block size) gives 15.625 blocks needed for the table.

Since a partial block is not allocated for the inode table, the system rounds up to 16 blocks and "inode fills" the 16th block; an additional 24 inodes are added to fill the last block so no space is wasted.

Data Storage

In each cylinder group, the areas before and after the super-block, cylinder group information, and inode table contain the blocks used to store data for ordinary files and directories (see Figure 12-1). Indirect blocks filled with pointers to data blocks also reside in this part of the cylinder group. Free space is allocated primarily in block sizes. Blocks are 8 Kbytes. This size is constant and is set at file system creation.

Having a large block size significantly reduces the number of disc accesses for large files, and this increases file system throughput. On the other hand, having a large block size for small files creates wasted space. To circumvent the wasted space problem, a block can be divided into 1 Kbyte fragments. The 1 Kbyte fragment size is constant and is set when you create a file system.

Allocation of Disc Space

Free space availability is determined from a bitmap associated with each cylinder group. The bitmap contains one bit for each fragment. To determine if a block is available, consecutive fragments are examined. A piece of the bit map from a file system using 1024 byte fragments and 8192 byte blocks is shown in Figure 12-4.

```
--------------------------------------------------------------------------
bitmap                00000000      00000011       11111100       11111111
Fragment numbers        0-7            8-15          16-23          24-31
Block numbers            0              1              2              3
--------------------------------------------------------------------------
```

Figure 12-4: Example Free Block Bitmap in an 8192/1024 File System

The free fragments in this example are fragment numbers 14-21 and 24-31, indicated by *1*s in the bitmap. The allocated fragments are fragment numbers 0-13 and 22-23, indicated by *0*s in the bitmap. Fragments in adjacent blocks cannot be used to create a full block; only 8 contiguous fragments starting on a block boundary can be used to allocate a full block. In this example, fragments 24-31 can be coalesced to form a full block, but fragments 14-21 cannot be. Also, if a partial block is allocated, the fragments must be consecutive and not cross a block boundary. For example, if 3 fragments are needed, fragments 16-18 can be allocated, but fragments 14-16 cannot be.

In an already existing file, each time additional data needs to be written to the file, the system checks to see if file size must increase. If file size must increase, one of three conditions exists:

1. There is enough space in the existing block or fragment. In this case the additional data is written into the already allocated space.
2. The file contains only whole blocks, and there is not enough room in the last block to hold the additional data. If more than a full block of data needs to be written a new block is allocated and the first additional block of data is written there. This process is repeated until less than a block of additional data needs to be written. When this happens, a block containing enough contiguous fragments is located and the remaining amount of additional data is written there.
3. The file contains fragments, but not enough fragments to hold the additional data. If the size of the existing data in fragments, plus the additional data, exceeds the size of a full block, a new block is allocated. Both the existing and the additional data are written to the new block following the process in condition 2 above. If the size of the existing and additional data is less than a full block, a block with enough contiguous fragments (or a full block) is located and allocated.

When a block or fragment has been located, the address is recorded in the inode and the free block bitmap is updated.

A certain percentage of free space must always be available in the file system to localize a file's blocks (accessing a file is quicker if the whole file is localized). To ensure the effectiveness of the disc allocation strategies, the disc should not be kept completely full.

Each file system maintains a parameter—*minfree*— that gives the minimum acceptable percentage of file system blocks that can be free. This percentage is specified at file system creation. BASIC 5.0 sets this value to zero (0), and it cannot be changed by BASIC 5.0.

PWS allows the user to specify the value of *minfree* when the disc is formatted with MKHFS. With HP-UX, you can change the percentage of free space at any time with the **-m** option of the *tunefs* command; and you can use the **-t** option of the *dt* command to see the current value. Changing *minfree* to a non-zero value decreases the ammount of free disc space available to the user by that percentage.

Changing *minfree* has some implications for BASIC users. BASIC allows user access to all data blocks in a file system with this caveat— performance degradations can occur when the file system become more than 90% full. As the reserve of free blocks approaches zero (0), the file system throughput rate tends to be cut in half because the file system can no longer localize the blocks for a file. If performance is impaired due to overfilling the disc, it can be restored by removing enough files to obtain a free space of 10%. In addition, the access speeds for files that were created during periods of marginal free space can be restored by recreating them after enough space if made available.

Some tools are available to the user. The System Utilities Disc (DISC_UTIL) provides a utility for backing up and restoring files. Just backup the files you want to recreate and then restore them. In addition, you can use the MEM_UTILS program to assist in removing files to free up disc space and recreating them to improve the access speeds for those files.

Allocation Methods

Allocation is performed on: a global level to determine placement of new directories and files; and a local level to determine the actual placement of data in blocks.

At the global level, an attempt is made to put all files from a single directory in the same cylinder group. When a new directory is created it is put in the cylinder group that has the greatest number of free inodes and the smallest number of directories.

Global policy specifies that once the file size reaches MAXBPG (the maximum number of blocks of data per cylinder group), HFS allocates blocks from another cylinder group. This enforces the grouping of all files within one directory into a single cylinder group by causing the less common larger files to be spread over several cylinder groups.

The global allocation routines call local allocation routines with requests for specific data blocks. The global information is not always aware of the status of every data block. It is the local allocation routines that decide which blocks to allocate according to the following order:

1. Allocate an asked for block.
2. Allocate a block on the same cylinder that is rotationally closest to the requested block.
3. Allocate any block within the same cylinder group.
4. Use a quadratic hash to find a new cylinder group and allocate a block somewhere in the new cylinder group.
5. A brute force search to find an available block.

Updating the HFS File System

Every time a file is modified, the system performs a series of file system updates. These updates are designed to ensure a consistent file system.

The **superblock** of a file system is written to the disc when the file system has been modified.

An **inode** contains information specific to the file it describes. An inode is written to the file system as soon as the file is written.

Data blocks, which includes: directories, indirect blocks, and files are written to the file system whenever they have been modified. All writes are immediate.

The **cylinder group** information is updated whenever the cylinder group has been written.

Maintaining Your File System

It takes time to create file systems. Therefore, it is important to maintain them to ensure their integrity. Correcting problems before they become catastrophic can save remaking an entire system.

All file system maintenance tasks are important. Here is a reminder of what needs to be done:

- Do a file system consistency check (HFSCK).

  Whenever you suspect there is a problem with your file system, run HFSCK.

- Check and understand disc usage.

  Unused and large files use space on your file system. You should check and remove or archive large, unused files when you run low on space.

- Back up your system.

  Your system should be backed up. Follow the guidelines and procedures in the earlier section called "Backing Up Volumes and Files".

Corruption of the File System

A file system can be corrupted in varied and subtle ways: improper shutdown of HP-UX before you boot BASIC; interrupting disc accesses, reads, and writes; hardware failures.

Switching from HP-UX to BASIC is perhaps the most subtle way to corrupt a file system. Always execute ***reboot -h*** to shut down HP-UX. Then, you can start up BASIC.

In general, you should regularly check a file system for inconsistencies and immediately repair the ones you find. **Allowing a corrupted file system to be further modified can be disastrous!**

Hardware Failure

Your Hewlett-Packard computer system is highly reliable, but it is good to remember that any piece of hardware can fail at any time. This isn't a prediction of gloom; it is merely suggesting that you take precaution. By following the preventative maintenance outlined in your installation guides and in this manual, you should be able to avert serious problems. Failures can be as subtle as a bad block on a disc pack, or as blatant as a non-functional disc controller.

Detection and Correction of Corruption

A Hierarchial File System Check (HFSCK) examines the structural integrity of a file system by checking on the data that is intrinsically redundant in a file system. The redundant data is either read from the file system or computed from known values. When an inconsistency is discovered, HFSCK reports or fixes the inconsistency depending on the mode of operation. This section looks at how an HFSCK works. The specific inconsistencies are examined in the section called "HFS Consistency Check".

SuperBlock Consistency

The summary information associated with the super-block is prone to error because every change to the file system's blocks or inodes modifies the summary information. The super-block and its associated parts are most often corrupted when the computer is improperly or inadvertently halted.

The super-block can be checked for the following inconsistencies:

- File-System size—this rarely happens.
- Free-Block count—this is fairly common.
- Free inode count—this is fairly common.

If HFSCK detects corruption in the static parameters of the primary (default) superblock (rarely happens), HFSCK requests the user to provide the location of an alternate superblock. The alternate superblock addresses were displayed during file system creation, and if you did not record them, be aware that it is a good idea to do so when you create another file system. An alternate superblock is always found at block number 16. If this superblock is also corrupted, you must supply the address of another.

File-System Size

The superblock is checked for inconsistencies involving file system size, number of inodes, free block count, and the free inode count. The file system size must be larger than the number of blocks used by the superblock and the number of blocks used by the list of inodes. While there is no way to actually check these values, HFSCK can check for them being within reasonable bounds. All other checks of the file system depend on the correctness of these values.

Free-Block Checking

A check is made to see that all the blocks in the file system were found and all the blocks marked as free in the free-block map are not claimed by any files. When all the blocks have been accounted for, a check is made to see if the number of blocks in the free-block map plus the number of blocks claimed by the inodes equals the total number of blocks in the file system. If anything is wrong with the free-block maps, HFSCK rebuilds them, excluding all blocks in the list of allocated blocks.

The summary information contains a count of the total number of free blocks within the file system. HFSCK compares this count to the number of blocks it found free within the file system. If they don't agree, HFSCK replaces the count in the summary information by the actual free-block count.

Inode Checking

The summary information contains a count of the total number of free inodes within the file system. HFSCK compares this count to the number of inodes it found free within the file system. If they don't agree, HFSCK replaces the count in the summary information by the actual free inode count.

Inodes

An individual inode is not as likely to be corrupted as the summary information. However, because of the great number of active inodes, a few inodes can become corrupted.

The list of inodes is checked sequentially starting with inode 2 (inode 0 marks unused inodes and inode 1 is reserved for future use) and going to the last inode in the file system. Each inode can be checked for the following inconsistencies:

- format and type
- link count
- duplicate blocks
- bad blocks
- inode size
- block count

Format and Type

Inodes can be a regular file **or** a directory. Inodes can be found in one of three states: unallocated; allocated; or neither unallocated nor allocated. The last state indicates an incorrectly formatted inode. An inode can get in this state when bad data is written into the inode list through, for example, a hardware failure. The only possible corrective action is for HFSCK to clear the defective inode.

Link Count

A count of the total number of directory entries linked to the inode is contained in each inode. HFSCK verifies the link count **stored** in each inode by traversing the total directory structure (starting from the root directory) and calculating an **actual** link count for each inode.

If the stored link count is non-zero and the actual link count is zero, it means that no directory entry appears for the inode. HFSCK can link the disconnected file to the *lost+found* directory. If the stored and actual link counts are non-zero and unequal, a directory entry might have been added or removed without the inode being updated. HFSCK can replace the stored link count by the actual link count. BASIC 5.0 does not provide for linking files, but you can do this in Pascal and HP-UX.

Duplicate Blocks

Each inode contains a list of all the blocks claimed by the inode. For large files, it contains pointers to lists (indirect blocks).

HFSCK compares each block number claimed by an inode to a list of already allocated blocks. If a block number is already claimed by another inode, the block number is added to a list of duplicate blocks. Otherwise, the list of allocated blocks is updated to include the block number. If there are any duplicate blocks, HFSCK makes a partial second pass of the inode list to find the inode of the duplicated block. This condition can occur by using a file system with blocks claimed by both the free-block list and by other parts of the file system.

The user is prompted to clear both inodes. Often clearing only one inode solves the problem, but the data in the other inode is suspect.

Bad Blocks

Each inode contains a list (or pointer to lists) of all the blocks claimed by the inode.

HFSCK checks each block number claimed by an inode for a value outside the range of the file system (lower than that of the first data block, or greater than the last block in the file system). If the block number is outside this range, the block number is a bad block number, and HFSCK prompts the user to clear the inode.

Inode Size

Each inode contains a sixty-four bit (eight-byte) size field. The size of this field indicates the number of characters in the file associated with the inode. The size can be checked for inconsistencies (e.g. directory sizes that are not a multiple of thirty-two characters, or the number of blocks actually used not matching that indicated by the inode size).

A directory inode within the file system has the mode word set to directory. The directory size must be a multiple of thirty-two because a directory entry contains thirty-two bytes of information. HFSCK warns of such directory misalignment. This is only a warning because not enough information can be gathered to correct the misalignment.

A rough check of the consistency of the size field of any inode can be performed by computing from the size field the number of blocks that should be associated with the inode and comparing it to the actual number of blocks claimed by the inode. HFSCK calculates the number of blocks that should be claimed by an inode by dividing the number of characters in the file by the number of characters per block and rounding up. HFSCK then counts actual direct and indirect blocks associated with the inode. If the actual number of blocks does not match the computed number of blocks, HFSCK warns of a possible file-size error. This is only a warning because does not fill in blocks in sparse data files.

Indirect Blocks

Indirect blocks are owned by an inode. Therefore, inconsistencies in indirect blocks directly affect the inode that owns it. Inconsistencies that can be checked include: blocks already claimed by another inode; and block numbers outside the range of the file system. Detection and correction of the inconsistencies associated with indirect blocks follows the same scheme used for direct block, and is done iteratively.

Data Blocks

The two types of data blocks are:

- ordinary data blocks which contain the information stored in a file. HFSCK does not attempt to check the validity of the contents of an ordinary data block; and
- directory data blocks which contain directory entries.

Each directory data block can be checked for the following inconsistencies:

- Directory inode numbers pointing to unallocated inodes.
- Directory inode numbers greater than the number of inodes in the file system.
- Incorrect directory inode numbers for "." and ".." (current and parent directories respectively).
- Directories which are disconnected from the file system hierarchy.

If a directory entry inode number points to an unallocated inode, then HFSCK can remove that directory entry.

If a directory entry inode number is pointing beyond the end of the inode list, HFSCK can remove that directory entry. This condition occurs if bad data is written into a directory data block.

The directory inode number entry for "." should be the first entry in the directory data block. Its value should be equal to the inode number for the directory data block.

The directory inode number entry for ".." should be the second entry in the directory data block. Its value should be equal to the inode number for the parent of the directory entry (or the inode number of the directory data block if the directory is the root directory).

If the directory inode numbers for "." and ".." are incorrect, HFSCK can replace them by the correct values.

HFSCK checks the general connectivity of the file system. If directories are found not to be linked into the file system, HFSCK links the directory back into the file system in the *lost+found* directory.

Uncorrectable File System Corruption

HFSCK might not be able to proceed in certain instances (e.g. all copies of the super block have been lost). If this happens to you, the HP-UX *fsdb* command is provided for such situations. The *fsdb* command is a file system debugger that **should only be used by an HP-UX file system expert, since it can easily destroy the entire file system**. See the *fsdb(1M)* entry in the *HP-UX Reference Vol. 1A: Section 1 (A through L)* manual to see what the command does.

Other Maintenance Tasks 17

The earlier chapters in this part of the *Installing, Using, and Maintaining the BASIC System* manual focused on maintaining your file systems. Maintenance of your file systems is crutial to using the BASIC system, but there are several other things you should do.

The following list mentions some things users take for granted or tend to forget. The information is minimal, but it should be sufficient to point you to useful information and imply additional things you can do.

- Clean the disc and tape recording heads (see your disc and tape drive's maintenance instructions).
- Clean the filters in the computer and in peripheral devices (see your computer's and peripheral device's maintenance instructions).
- Clean your CRT screen using an appropriate cleaning solution and cloth or tissue.
- Obtain a fresh supply of pens for your plotter. Check your supplies of paper and obtain what you need.
- Clean your printer. Obtain fresh ribbons or cartridges. Check your supply of paper and any other materials or chemicals you might need.
- Check the connections and cables that link all parts of your system to see that they are operating correctly and are not about to come apart.
- Rotate your tires and otherwise examine your system to see that it is performing nicely.

What to Do Next

While maintaining your BASIC system is something you generally do after you have set up and used the system, maintenance can be a point from which you read about new or unusual things to do with BASIC 5.0. The chances are good that you are quite familiar with using BASIC in your present situation, and the question being raised here is: What can you do to continue to grow in your ability to use BASIC 5.0?

One totally new feature of BASIC 5.0 is the ways in which you can use an HFS file system. That system is quite different from what was previously available to Hewlett Packard BASIC, and this might be a good time to further examine the advantages of using an HFS file system. The HFS file system is discussed in this manual in the chapters called "HFS File System Implementation", "Mass Storage Concepts", and "Using Files and Directories".

Utilities

This **part** of the *Installing, Using, and Maintaining the BASIC System* manual describes how to use the various utilities.

In previous versions of BASIC, the utilities may have been described in more than one manual. This time, the descriptions were combined; and consequently, what follows is one rather large chapter that explains all the utilities. The idea was to help you by placing the information in one location.

To use any of the BASIC utilities, read the introductory material in Chapter 18. Then, you can see what is available by reading the brief descriptions of the utilities. To use a particular utility, read the detailed description. To actually use a utility, you might need to examine your media. If you have single-sided media, you can do a CAT of the utilities discs to see which utilities are on on a particular disc. If you have double-sided media, all the utilities fit onto one disc.

BASIC Utilities Library 18

This chapter describes the utilities provided with the BASIC 5.0 system. These utilities work with previous versions of HP Series 200/300 BASIC, allowing for some restrictions and exceptions. The following table shows the topics and page numbers.

Terminology

A **utility** is a program or subprogram that performs a specific, routine task. Utility programs do such things as: examine the status of an interface, dump a file to a printer, or create a backup copy of a file. Think of a utility as a program that does a task in a manner that goes beyond what is done by a conventional BASIC statement such as a CAT, COPY, or LIST.

A **library** is a collection of programs; and in this case, the utilities library is a collection of approximately 20 HP BASIC utilities that are housed on a flexible disc or discs. If your system uses double-sided flexible discs, the entire library fits on one disc named *BASIC Utilities.* If you have single-sided media, the library is contained on three discs; and in this case, you can use a CAT statement to find out which disc contains a particular utility. There are some differences in the locations of utilities on discs among previous versions of BASIC.

Prerequisites and Overview

You should read this material before you use any utility.

Necessary Skills

The material in this chapter is for an experienced HP BASIC programmer. At the least, you should be able to do the following things:

- Configure a hardware system.
- Boot the BASIC system and load binaries.
- Initialize discs and copy programs.
- Write programs or use applications programs.
- Handle SUBs and CSUBs within a conventional program.
- Interpret what the computer is doing, or ought to be doing

It is helpful to have performed the tasks described in the *BASIC Programming Techniques* manual and the *Installing, Using and Maintaining the BASIC System* manual.

Notes for BASIC 5.0 and HFS Files

Several BASIC utilities use PHYREAD and/or PHYWRITE; and these utilities usually require a LIF format for directories. In a LIF format, files are located in contiguous blocks. This is different from HFS, where files can be fragmented in a way that is transparent to the user. Some of the utilities described later have been made to detect an MSI'd device that is formatted for HFS, and to terminate or otherwise behave accordingly.

To make some utilities work with HFS discs, there is a subprogram in a file called HFSDISC that will be loaded by some utilities when necessary; and this will require that the appropriate utilities disc be present for the first part of program execution. (As in other cases, do a CAT of your utilities discs if you need to find the file.) This is not different from the situation you will read about for utilities that call the CAT, INFO, or CREATE subroutines. HFSDISC includes a function FNHfsdisc— which will examine the MSI'd device, and inform the calling program that it is trying to talk to an HFS disc.

What this means, in later discussions of utilities, is that some utilities will give erroneous results if used with HFS discs; and this may not always be obvious. The next few paragraphs describe some requirements or limitations. These are mentioned again, by utility, in the later "Brief Descriptions of Utilities" section.

The following utilities **do not depend** on having a LIF formatted disc, and therefore, they will also work with HFS discs:

- INITIALIZE
- DUMP

The following utilities were changed to utilize the feature mentioned above:

- MASS_STOR
- FBACKUP
- CBACKUP
- VERIFY_LIF

The LISTER utility was not changed to understand HFS, and consequently, you cannot use it with HFS discs at this time.

The following subroutines have not been changed and do not "understand" HFS discs.

- CAT
- INFO

The CREATE subroutine uses PHYWRITE, does not understand HFS, and could cause a serious loss of data if used on HFS files.

Organization of Topics

The organization of topics shown in the previous table was arbitrary because there is no real or logical basis for discussing one utility ahead of another. Your situation and inclination dictate their use; so you can note what you want to do; and then go to the appropriate section to get the information you need. In particular, after some discussion of basic information, you get a brief description of each utility. This was done because there are many utilities, and by mentioning all of them in one place, you can orient yourself to what is available. Following this, each utility is described, and you can selectively refer to the ones you want to use.

Conventions for Using Keys

Because utilities are developed over time, the user interface can vary among programs; and this means that knowing which keys to type when you use a utility program is important.

Besides the interface, the keys on the keyboard in your hardware system can vary. Most Series 200 computers use the HP 98203A/B/C keyboard, and most Series 300 computers use the ITF keyboard. This latter keyboard is also used with the Model 217 and Model 237 computers.

Here are the major differences between the keyboards and how you deal with them.

- The Return and Enter keys on an ITF keyboard are equivalent, respectively, to the ENTER and EXECUTE keys on the HP 98203A/B/C keyboards. Any reference to typing Return means type ENTER if you do not have an ITF keyboard. Similarly, any reference to typing Enter means type EXECUTE if you do not have an ITF keyboard.
- The HP 98203A/B/C keyboard has some keys not found on the ITF keyboard. These keys are typically available on the ITF keyboard as softkeys. Find the softkey label for a key on the HP 98203A/B/C keyboard and type the corresponding softkey. For example, CONTINUE on the HP 98203A/B/C keyboard is typically available as a `CONTINUE` or `Continue` softkey label on the ITF keyboard and you type a corresponding softkey such as f3.
- The ITF keyboard can have several menus of softkey labels: System, User 1, User 2, and User 3. If you have an ITF keyboard, most utilities employ the softkeys in the System and User 1 menus; and some use the User 2 or User 3 menus. Typing Shift Menu moves among User menus and typing System provides the System menu. When you use a utility, you can always move among the menus to see what functions are available.

System Requirements for Utilities

A Series 200/300 computer, monitor, flexible disc drive, and the BASIC language system is the minimal system that can effectively use utility programs or subprograms.

The use of some utilities requires additional components such as an interface card or printer. Some utilities apply only to older systems, since the utility's capability was added to the system or to a related peripheral device.

Utilities most often perform particular tasks (in contrast to application programs which perform a range of tasks within a major task). For example, one utility examines the status of your GPIO interface. Since it does not do anything else, having a GPIO interface is a requirement for using the utility. Another utility dumps graphics to a printer, and this obviously means that you need a printer. Be aware of potential system requirements when you think about how you might use a utility.

Besides hardware, you have to load certain binaries to use some utilities. An attempt was made to mention required binaries, but your system may have additional requirements.

CSUBs Can Be Utilities

Most utilities are BASIC PROG files. However, some are CSUBs (Compiled SUBprogram: written in Pascal and generated for use with BASIC using the CSUB Utility). If you have a situation where speed is critical, and if you have the programming expertise, you can create a special CSUB that can be accessed via BASIC. Here are the guidelines.

1. Create CSUB statements that perform your task using a special CSUB utility from the Pascal language system. The directions for how you do this are in the *CSUB Utility* manual; a package that is acquired separately from BASIC.
2. Create a BASIC program which loads the CSUB subprogram via a LOADSUB statement. The subprogram does not have to be loaded within a program, but it is handy to do it this way. After the subprogram is loaded, it can be invoked via a CALL statement just like normal SUB subprograms.

Do not dwell on this unless you know what type of utility you need to develop and have the necessary programming expertise. If you are curious and wish to learn how to create a CSUB, the process is described in the *CSUB Utility* manual. The *BASIC Language Reference* manual provides additional information.

Why Have a Utilities Library?

Having become familiar with what a utility is, you might ask: Why use a utility program? Although it may have been better to ask the question **before** you plowed through the information, here are two illustrations:

1. Suppose you are working with the HP-IB interface. You have run some programs and are a bit confused about what is happening. You could take a moment to run the HPIB_STAT utility to get a listing of the contents of the status registers for either the internal or external HP-IB interface. This information could help you see what is happening.
2. Suppose you have a situation where your discs must meet HP LIF standards (Hewlett-Packard Logical Interface Format) and you are uncertain about the status of several discs. You could use the VERIFY_LIF utility to determine whether a disc meets the standard.

These illustrations suggest that utility programs can provide useful information or perform a task easily that would be difficult by conventional methods.

The Two Levels of Documentation

Should you decide to use a particular utility, there are two sources of information. The "Brief Description of Utilities" section contains short descriptions of the utilities. You can skim this section to see what is available. The later "Detailed Description of Utilities" section contains complete descriptions of how to use the utilities. Refer to this section according to the utility you want to use.

Brief Descriptions of Utilities

The next several subsections provide brief descriptions of the utilities.

(CAT) Catalog Subprogram

Do **not use** CAT with HFS discs. This subprogram reads the directory from a disc in the default **msvs** into memory as an array. Then, you can access eight pieces of information: file name, file type, starting address, file length, time of creation, volume data, protect code designation, and defined record size. The subprogram also determines the number of files on the disc.

You have to create a program (PROG file) which CALLs the catalog subprogram.

The function of the CAT utility has been added to the BASIC System via `CAT TO String_array$(*)`, which requires the MS binary. See the "Data Storage and Retrieval" chapter of *BASIC Programming Techniques* and the *BASIC Language Reference* for additional information.

(MASS_STOR) Mass Storage Program

This program is a collection of subprograms. MASS_STOR **does make** use of HFSDISC. Each subprogram is called by typing a softkey. The following list is complete, but not in a particular order.

- The Extended CAT subprogram lists an extended directory catalog which includes purged files still in the directory.
- The FILE SIZER subprogram lets you adjust the size of data files.
- The ZAP subprogram purges all files on a LIF disc.
- The Change Volume Label subprogram lets you change the volume label on a LIF disc. (Volume labels can also be changed with the PRINT LABEL statement; see the *BASIC Language Reference* for additional information.)
- The PURGE subprogram purges a file.
- The unPURGE subprogram lets you recover from accidental purges.

[1] With double-sided media, the *BASIC Utilities* disc contains the contents of both the *BASIC Utilities 1* and *BASIC Utilities 2* discs.

- The REPACK subprogram packs files on a disc to the front and frees available disc space.
- The Change MSVS subprogram lets you change the current msvs.

Use of MASS_STOR varies slightly depending on which keyboard you have.

(DUMP) Formatted Record Dump

This program dumps a specified record from a disc in: integer, hex, octal/ASCII, hex/ASCII, LIF directory, or LIF or HFS system formats.

The softkey menus vary slightly depending on which keyboard you have.

(INFO) Directory Information Subprogram

Do **not use** INFO with HFS discs. This subprogram searches a disc directory for the presence of a file name which you specify. If the file is found, the following information is displayed:

- The first and last physical record addresses
- The number of defined records
- The defined record length
- The file type

If none of the above results are displayed, the specified file was not found.

You have to create a program that CALLs the utility subprogram.

(CREATE) Create File Subprogram

Do **not use** CREATE with HFS discs. This subprogram goes beyond the BASIC CREATE keyword.

1. You supply a file name, number of defined records, defined record size, and file type.
2. The subprogram creates a file entry in the disc directory and points to where the file's contents can be inserted before, after, or in between other files.
3. The new file's address is returned to the calling program if the file's contents will fit on the disc. Otherwise, you are informed that the file will not fit.

You have to create a program that CALLs the subprogram.

(INITIALIZE) Extended Mass Storage Media Initialize

This program lets you specify a legal volume label for a LIF or an HFS disc. Then, it lets you initialize the disc and specify its directory length.

(VERIFY_LIF) Verify Logical Interchange Format

VERIFY_LIF **does make** use of HFSDISC. This program reads record zero on the mass storage media and checks to see whether it meets the HP LIF standards. The status of your media is displayed.

(CBACKUP) Complete Disc Backup

CBACKUP **does make** use of HFSDISC. This program, intended for use with one disc drive, lets you backup the contents of a flexible disc which meets HP LIF standards. The disc size can be 3.5, 5.25, or 8 inch.

Do not use the CBACKUP program with a hard disc because this program assumes you can interchange the media. Also, **do not** use this program with the SRM system, because SRM **does not** support PHYREC.

(FBACKUP) Selective File Backup

FBACKUP **does make** use of HFSDISC. When you have one disc drive, use this program to backup particular files on a 3.5, 5.25, or 8 inch flexible disc which meets HP LIF standards.

Do not use the FBACKUP program with a hard disc because this program assumes you can interchange the media. Also, **do not** use this program with the SRM system, because SRM **does not** support PHYREC.

(TAPEBACKUP) CS/80 Tape Backup

This program lets you backup the contents of some HP CS/80 discs onto a DC600 or DC150 tape cartridge, and conversely.

The nature of a CS/80 disc drive complicates the procedure somewhat and requires that you coordinate information from several sources. You need to be familiar with your particular CS/80 disc drive and the rest of your computer system. You should coordinate the procedure for using this utility with information about your CS/80 disc and the DC600 tape cartridge.

(PHYREC) Physical Record CSUB

PHYREC is a special subprogram called a CSUB which was created by using the CSUB Utility. PHYREC lets you perform bit-by-bit copies of an integer array in memory to a file on a mass storage media, and vice versa.

The PHYREC utility contains two CSUBs:

```
10   CSUB Phyread(Sector, INTEGER Int_array(*))
```

and

```
20   CSUB Phywrite(Sector, INTEGER Int_array(*))
```

(LISTER) List Files

Do **not use** LISTER with HFS discs. This program provides options for listing BASIC programs.

1. First, it lets you enter the name of one file or enter the names of several files to be listed. The file must have been saved or output as ASCII strings.
2. Then, the values of several parameters are displayed (printer select code, pagination, perforation, lines per page, spacing, omit page numbers, first page number, print range, trailer, edit text, width, number of listings). You can alter these values so that the listing suits your needs.
3. Then, you can do some limited editing of comments.
4. Finally, your program or programs are listed.

A printer is required for effective use. The program is designed to list BASIC programs.

(82905DUMP) Dump Graphics Subprogram

This subprogram accepts a device select code, and then dumps a graphics display to an HP 82905B printer.

You have to create a program that CALLs the subprogram. The subprogram does not work with bit-mapped display.

(GDUMP_R) Rotated Dump Graphics Utility

This CSUB utility dumps graphics raster images to a printer. It is like the DUMP GRAPHICS statement, except that it rotates the image 90°.

(BPLOT) Raster Store and Load Utility

This utility provides the capabilities of storing and loading rectangular "blocks" of raster data (using a numeric array).

(LEX_AID) Creating a Lexical Order Table

This program simplifies the process of creating user-defined lexical tables.

You should coordinate the use of this program with information from the string manipulation chapter in the *BASIC Programming Techniques* manual. Do not use this utility unless you know what to do and why you need lexical tables.

The Status Utility Programs

Four programs are available which display a formatted listing of the contents of the status registers of either an interface or an ASCII or BDAT file. The programs are:

1. Status of HP-IB Interface (HPIB_STAT)
2. Status of RS232 Interface (RS232_STAT)
3. Status of GPIO Interface (GPIO_STAT)
4. Status of Data Files (FILE_STAT)

These programs display information about what is happening in the registers of an interface or file.

The interface or file must be present for you to get information about the contents of its status registers. Absence of an interface or a file is displayed.

(MEM_UTILS) Memory Utilities

This program adds five memory resident, soft-key accessed utility subprograms to BASIC 5.0. The subprograms are stored in a memory volume, loaded and called by a typing aid, and delete themselves from memory when they are done. You get these features: an interactive catalog (full 80 columns); appending volume copy; an interactive volume purge; copying a text file to a printer; and editing of a one page memo.

(FONT_ED) Font Editor

This utility provides a way to read, decode, and edit the bit patterns of the fonts supplied with the BASIC system. If you want to create your own fonts, the fonts supplied with the system can be a good "starting" point. The detailed explanation of the Font Editor in the next section describes how to use the utility. Note that the soft font display architecture and the corresponding BASIC language capabilities are described in the "Custom Character Fonts" section of the "Communicating With the Operator" chapter in the *BASIC Programming Techniques* manual; and you will need to be familiar with that material to effectively use the Font Editor utility.

(DISC_UTIL) Disc Utility

This utility is a collection of utilities that let you do such things as: show online mass storage devices; initialize a disc; store the BASIC system and binaries; check the consistency of an HFS volume; and call the BACKUP/RESTORE utility. These utilities are all described in the *Installing, Using and Maintaining the BASIC System* manual.

(MKHFS) Make Hierarchial File System

This utility is available on *BASIC 5.0 System Utilities* disc. It allows you to create an HFS (Hierarchial File System). The utility is not described in detail in this manual. See the part called "Maintaining Your BASIC System" in the *Installing, Using and Maintaining the BASIC System* manual to get more information.

(HFSCK) Hierarchial File System Check

This utility is available on *BASIC 5.0 System Utilities* disc. It allows you to check for inconsistencies in an HFS (Hierarchial File System) volume. The utility is described in the part called "Maintaining Your BASIC System" in the *Installing, Using and Maintaining the BASIC System* manual.

(BACKUP/RESTORE) Backup and Restore Utility

This utility, which is part of DISC_UTIL, lets you backup and restore BASIC files. It is described in the "Using the BACKUP Utility" chapter in the *Installing, Using and Maintaining the BASIC System* manual.

(VERIFY) Verify and Labeling Peripherals

This utility is available on the *BASIC 5.0 System Utilities* disc. It lets you verify which peripheral devices are connected to your system and gives you time to write down their volume specifiers. The utility is described in the "Verifying and Labeling Peripherals" chapter in the *Installing, Using and Maintaining the BASIC System* manual.

Detailed Descriptions of Utilities

The next several sections describe how to use the utilities. Be aware that some utilities require certain binaries, interfaces and/or devices. Some utilities require familiarity with HP BASIC. An attempt was made to present complete descriptions, but you might run into something unusual. Try to work through problems by relating information in this manual to what a utility seems to be doing.

Note that most utilities attempt to trap errors, but take care to supply correct entries and type the correct keys. Indiscriminate typing can produce unexpected or detrimental results.

In the descriptions, the term **execute** might not literally mean: type [EXECUTE] or [EXECUTE]. Instead, it can mean: do whatever is implied by the context of a situation. In most cases, you type something and type [Return] or [ENTER]. At other times, you execute a command such as `LOAD "MYFILE"`, and then execute another command such as `RUN`. Be sensitive to the context of a utility.

(CAT) Catalog Subprogram

The function of the CAT utility has been added to the BASIC.system via **CAT TO String_array$(*)**, which requires the MS binary. See the "Data Storage and Retrieval" chapter of *BASIC Programming Techniques* and the *BASIC Language Reference* for additional information.

Access this subprogram from within a main program. If you do **not** know how to do this, study the "Subprograms" chapter in the *BASIC Programming Techniques* manual. Fundamental information about storage of data on a media is provided in the "PHYREC" section (end of this chapter).

Do not interrupt the program while it is running. You need a printer to obtain hard copy information. The PDEV binary is required (if you use LOADSUB FROM) in addition to those required to operate your computer system.

The following procedure illustrates **one** way to access the subprogram.

Execute the **SCRATCH** command to ensure that no program is in memory. Then, write a small main program that calls the CAT subprogram. Just how you create the main or calling program can vary. The following steps illustrate the major things to do.

1. Declare parameters to be passed to the subprogram. These are usually input parameters, but can be selected output parameters. **Dstart**, an input parameter, is the address of the sector at which file entries begin. **Dleng**, an input parameter, is the length of the directory. **Num_files**, an output parameter, is the number of files in the directory.

   ```
   10     INTEGER Dstart,Dleng,Num_files
   ```

2. Initialize parameters that are passed to the subprogram. **Dstart** is set to 2, the usual address for the sector in which file entries begin. **Dleng** is set to 14, the usual directory length of a 5.25-inch flexible disc. (This value might need to be greater for a hard disc.) Read the "PHYREC" section (end of this chapter) to get more complete information.

   ```
   20     Dstart=2
   30     Dleng=14
   ```

   The **Num_files** output parameter is initialized in the subprogram, but you should declare it in the main program if you want to read its value after the subprogram has been executed.

3. Dimension the array that receives directory information.

```
40    DIM A$(895)[18]
```

The dimensioning is done according to the expression: `Dleng*64-1`. This is 895 because `Dleng=14` in the above example. The `[18]` specifies a string length of 18.

4. At this point, CALL the subprogram. The calling syntax is:

```
50    CALL Cat(A$(*),Dstart,Dleng,Num_files)
```

5. Output information derived from the called subprogram to the CRT, a printer, or a file. This lets you see the information or have it available to use later. For example, the statement:

```
60    PRINT "Number of files is ";Num_files
```

displays the number of files on the disc. Recall that the default print device is the CRT.

With regard to other output, the information read into the array, `A$(*)`, is available in segments of eight returned values per file. The first segment is `A$(0)` through `A$(7)`. The next is `A$(8)` through `A$(15)`, and so on. The eight pieces of information are:

```
1st- File name
2nd- File type
3rd- Starting address
4th- File length in physical records (in defined
     records for BDAT files)
5th- Time of creation (not currently used)
6th- Volume information (not currently used)
7th- Protect code indicator
8th- Defined record size
```

The following program segment, inserted in the main program after you call the subprogram, prints file information:

```
70     FOR K=0 TO Num_files * 8-1   ! Print 8 values/file
80       PRINT A$(K)                ! Print array value
90       WAIT .4                    ! Let user see values
100    NEXT K                       ! Continue loop
110    PRINT "That's all.  You are back in the BASIC system."
```

When you run the program, watch the displayed values. You will see the eight items mentioned above. The 1st value is the file name. The 2nd is the file type and so on. The 8th value is the record size (probably 256).

Printing the values of `A$(*)` this way serves no real purpose other than to let you see what happened. You would probably use the returned values for other purposes.

6. Terminate the context of the main (calling) program. Use:

```
   120    END
```

Your main program can do other things, but these statements illustrate a means of calling the CAT subprogram and using data it provides.

Before you run the main program (calling program), insert the disc which contains the CAT subprogram into the default disc drive and load the CAT subprogram into memory by executing:

```
LOADSUB FROM "CAT"
```

This appends the CAT subprogram to your calling program and adjusts the line numbers accordingly. Alternately, you could use:

```
LOADSUB ALL FROM "CAT"
```

To check the appending, execute `EDIT`. Study the program listing to see that the CAT subprogram begins after your main program ended.

At this point, remove the disc which contains the CAT subprogram and insert the disc for which you want data. Then, type [RUN] or the RUN softkey [SYSTEM menu on ITF keyboard]. The program accesses your disc and then prints the returned values.

Experiment with other calling programs to develop skill in writing programs that call subprograms.

(MASS_STOR) Mass Storage Utility

Skim through the entire section before you use the utility.

Avoid Using SRM

Do not use this utility with the SRM system. It will not work.

Access this program by executing:

```
LOAD "MASS_STOR"
```

An asterisk, *, is displayed (lower-right corner) while the program loads. Type RUN when the asterisk disappears.

Use of the MASS_STOR utility involves use of nested menus which are displayed via softkey labels. The softkey labels change often. Therefore, you need to keep track of what you are doing and which softkey labels are displayed.

In a similar vein, MASS_STOR is a program that executes subprograms which you call by typing softkeys. Then, you call subprogram functions by typing softkeys. The implication is that the softkey labels change often. You need to keep track of what you are doing and which softkey labels are displayed.

Computer systems which use the HP 98203A/B/C keyboards use softkeys [k0] through [k9]. The ITF keyboards (such as the HP 46020A) use softkeys [f1] through [f8] in three menus (System, User 1, and User 2).

In general, softkey [k0] ([f1] in the User 1 menu) causes the displayed menu and the current MSVS to be printed. Exit MASS_STOR by typing [k9] ([f7] in the User 1 menu, or [f4] in the User 2 menu). Softkey [f8] toggles between User 1 and User 2 menus.

The left column in the following list shows the displayed softkey label of a function provided by the MASS_STOR program. This information is duplicated in a displayed menu when you run the program. The second column indicates: which softkey you type; what the utility does; and how you use the utility. After you invoke a utility, most of its functions are called by typing a softkey. The labels are self explanatory. The few exceptions are explained. Go slowly and notice alternatives the first time you use a utility.

Softkey Label **Softkey and Procedure**

`FILE SIZER` k1 (f3 in the User 2 menu) lets you change the size of ASCII or BDAT files. When prompted, insert the disc to be sized. Then enter the file name you wish to change. Abort by typing ENTER (or Return) with no data. Then, you can select: SPECIFY (enter a new file length), FIT DATA (moves end of file to end of last sector with data), or EXIT (returns to main menu). Follow the directions.

`ZAP a disc` k2 (f2 in the User 2 menu) clears the directory of a disc by placing a −1 in the file type field of the first directory entry to denote the logical end of the directory (similar to INITIALIZE). When prompted, enter the MSVS (CONTINUE accepts default). Then insert the correct disc, and type the softkey for `YES` to ZAP the disc. **Note: This will destroy all files and data on the disc.** (You can also type the softkey for `NO` to terminate this function.)

`PURGE a file` k3 (f5 in the User 1 menu) purges a file on the current MSVS, Just follow the prompts.

`UnPURGE a file` k8 (f6 in the User 1 menu) unpurges a file on the current MSVS. An extended CAT is displayed. Follow the prompts. Choose the correct type of file (ASCII of BDAT). Note what has and has not been purged during the extended CAT display. Arrows point to files previously purged.

`REPACK a disc` k4 (f1 in the User 2 menu) repacks a disc to give you contiguous available space at the end of the disc. Pay attention to available softkey options. You are prompted for the MSVS. Type it, and type ENTER (or Return); entering no MSVS accepts the default. Then when prompted, insert the disc to be repacked. Type CONTINUE (or the `Continue` softkey on the System menu). Then, note the options via softkeys (User 1 menu) and proceed. Arrows in the extended directory catalog point to purged files that are lost during repacking.

`Change MSVS` k5 (f2 in the User 1 menu) lets you specify a new MSVS. The current MSVS is displayed. Type the softkey for the displayed option you want. (Toggle between User 1 and User 2 to see all options.) After selecting an MSVS, you can alter the select code and address if the displayed values are not acceptable.

`Extended CAT` k6 (f4 in the User 1 menu) displays an extended catalog of a disc directory.

`Chng Vol label` k7 (f3 in the User 1 menu) lets you change the volume label of a disc. The current label is displayed. When prompted, you can choose whether or not to change this label. If you select `YES`, then enter a new label (up to 6 uppercase characters or numerals, beginning with an alpha character). If you enter nothing, a volume label containing six spaces is placed on the disc. **Note:** The PRINT LABEL and READ LABEL statements, in the MS binary, also provide this function.

(DUMP) Formatted Record Dump

This utility dumps mass storage records in a variety of formats. To understand this section you may need to have a knowledge of LIF (Logical Interchange Format) discs.

Access this program by executing:

```
LOAD "DUMP"
```

When the asterisk disappears, type **RUN**. Then, insert the disc which contains a record to be dumped and type CONTINUE or the **Continue** softkey [SYSTEM menu of an ITF keyboard]. If your disc is **not** in the drive designated by the current **msvs**, enter an appropriate msvs before you continue.

Several alternatives for dumping a record are selected via softkeys. On an ITF keyboard, toggle between softkey menus to see all options. The default dump is hexadecimal. The options are:

RECORD# This lets you select the record you wish to dump.

Hex Dumps a record in hexadecimal format.

Hex/ascii Dumps a record in hexadecimal/ASCII format, where control characters are masked out and replaced by periods.

LIF sys Dumps the LIF volume label in the correct format.

CAT Displays a catalog of the disc.

N+1, N-1 Allows you to increment or decrement the current record number and dumps a record according to the current format.

Integer Dumps a record in an integer format.

Oct/ascii Dumps a record in an octal/ASCII format. Control characters are masked out and replaced by periods.

LIF dir Causes subsequent dumps to have a LIF directory format.

Examples of dumped records were not provided because use of this utility assumes you know what you are doing.

Dealing with the Pause

The program pauses after the fourth entry is printed. Type CONTINUE or type the `Continue` softkey [SYSTEM menu on an ITF keyboard] to print the rest of the record.

Type the `EXIT` softkey to return to the BASIC system.

(INFO) Directory Information Subprogram

This is another subprogram for which you must create a calling (main) program similar to the one created for calling the CAT subprogram (see the preceding "Catalog Subprogram (CAT)" section). Read through the entire section before you do anything.

Your calling program should do the following:

1. Begin with a CAT statement so the directory filename is displayed. Then, input the filename for the directory for which you want information. Use an INPUT statement.

   ```
   10    CAT
   20    INPUT "Enter file name ",File$
   ```

2. Declare several parameters as integers. These parameters allow you to pass values to the subprogram or pass values back to the main program (more on this below). Use:

   ```
   30    INTEGER Dstart,Dleng,Strt,Stp,Lrecs,L_rec_l,Ftype,Files
   ```

3. Initialize the input parameters `Dstart` and `Dleng`. See CAT subprogram section for reasons. Use:

   ```
   40    Dstart=2
   50    Dleng=14
   ```

4. Dimension the array `A$(*)`, which is used when the INFO subprogram calls the CAT subprogram. (Note that `A$(*)` and `Files` are optional parameters that may appear at the end of the parameter list in the call to the INFO subprogram. See the preceding "Catalog Subprogram (CAT)" section for reasons.) Use:

   ```
   60    DIM A$(895)[18]
   ```

5. Call the INFO subprogram. Use:

   ```
   70   CALL Info(File$,Dstart,Dleng,Strt,Lrecs,L_rec_l,Ftype)
   ```

6. At this point, print as many of the output parameters (described below) as you wish (the ones declared by the INTEGER and DIM statements above). Include them in your calling program between the CALL and END statements.

Note that some of the declared variables are output parameters that do not have to be initialized. Including them in the calling statement lets you pass them back to the calling program so you can display their values. The output parameters are:

- `Strt` An integer variable that contains the address of the first physical record of `File$`. A **-1** means the file was not found. If the file was not found, all other output parameters are **0**.
- `Stp` An integer variable that contains the address of the last physical record of `File$`.
- `Lrecs` An integer variable that contains the number of defined records in `File$`.
- `L_rec_l` An integer variable that contains the length (in bytes) of each defined record of `File$`.
- `Ftype` An integer variable that contains the file type: 1 is ASCII, −5775 is BIN, −5791 is BDAT, and −5808 is PROG.

Have CAT on Disc With INFO

Be sure the CAT subprogram is on the disc with the INFO subprogram. Load the PDEV binary if you use a LOADSUB FROM statement. Do not interrupt the program while it is actively running.

Notice what is implied by the **note**. The INFO subprogram calls the CAT subprogram to get additional information. But the disc for which you want directory information may be a different disc. If the CAT subprogram is not on the disc which contains your information, use `LOADSUB ALL FROM "CAT"` to append the CAT subprogram to your main (calling) program. Then, to make sure the INFO subprogram does not attempt to reload the CAT subprogram, change line 370 in the INFO subprogram to:

```
370  ! LOADSUB ALL FROM "CAT"
```

The line number will vary if you appended it to your main program, but the `!` keeps the statement from being executed. If you wish to save the CAT subprogram as a permanent part of this utility, you need to comment out the DELSUB that follows the call to the CAT subprogram.

(CREATE) Create File Subprogram

This is another subprogram for which you must create a calling program. Apply the procedures used for calling the CAT or INFO subprograms.

The input parameters are:

- `File$` Input the name of the file to be created in the directory.
- `Num_recs` Declare as an integer variable and initialize to the number of defined records in `File$`.
- `L_rec_length` Declare as an integer variable and initialize to the length of the defined records (words/record for ASCII, PROG, and binaries; bytes/record for BDAT files). The record length of ASCII, BIN, and PROG files must be 256.
- `Ftype` Declare as an integer variable and initialize to: 1 for an ASCII file, 0 for an empty file, −1 for a logical end of a directory, −5775 for a BIN file, −5791 for a BDAT file, and −5808 for a PROG file.
- `Dstart` Declare as an integer variable and initialize to the sector in which the directory starts (normally 2).
- `Dleng` Declare as an integer variable and initialize to the length of the directory in sectors (normally 14).

The output parameters are:

- `New_address` An integer variable that returns the file's new address.
- `Err` An integer variable that returns the status of the create routine: 0 for no valid entry, 1 for no room in directory for entry, 2 for no file space available, or 3 for duplicate file name.

The CALLing syntax is:

```
CALL Create(File$,Num_recs,L_rec_length,Ftype,Dstart,Dleng,New_address,Err)
```

Do not interrupt the subprogram.

(INITIALIZE) Extended Mass Storage Media Initialize

Except for the ability to create a LIF directory of non-standard size, the capabilities provided by the INITIALIZE utility can be realized using the INITIALIZE, READ LABEL, and PRINT LABEL keywords.

Access this program by executing:

```
LOAD "INITIALIZE"
```

Type RUN after the asterisk (lower-right corner) disappears.

On the prompt, enter the **msvs** for the device which contains the disc you wish to initialize. [ENTER] or [Return] with no entry accepts the current msvs. Insert the disc to be initialized and type [CONTINUE] or the Continue softkey [SYSTEM menu of an ITF keyboard].

Select the initialization action via softkey. The displayed softkey labels are: YES, NO, or EXIT. Wait while several initialization actions take place. Do not interrupt the process. If you inserted a previously initialized disc, some additional options are provided. They are:

- OKAY (to reinitialize).
- RESTART (to begin the utility again).
- EXIT (to leave the utility).

After initialization, enter a volume name (six characters, uppercase, alpha first character). The default is six blanks. You then enter a directory length and type [CONTINUE] (or the Continue softkey [SYSTEM menu of an ITF keyboard]). The default is 14 physical records. Avoid creating a directory too large to fit onto your disc. Keep directory lengths between 0 and the maximum allowable value for your disc. If you selected the NO option, you still have an option to change the volume label within the six character limitations.

The program selects the optimum interleave factor for the given msvs.

(VERIFY_LIF) Logical Interchange Format

Skim the entire section before you do anything.

Access this program by executing:

```
   LOAD "VERIFY_LIF"
```

Type RUN when the asterisk (lower-right corner) disappears.

On the prompt, insert the disc to be verified and type [CONTINUE] or the **Continue** softkey [SYSTEM menu of an ITF keyboard].

Wait a moment. Then, the status of your disc is displayed. It either meets the LIF standard or it does not.

(CBACKUP) Complete Disc Backup

Use this utility when you have one flexible disc drive. Skim through the entire procedure first.

Access this program by executing:

```
LOAD "CBACKUP"
```

Wait for the asterisk in the lower-right corner to disappear. Execute the program with the `RUN` command. Wait for the program to go through an initialization process. **Do not** remove the BASIC utilities disc until prompted to do so.

Have the master disc (the one containing files you wish to backup) and an initialized backup disc (the one you copy the files onto for backup) available. Do **not** type [RUN] or the `RUN` softkey [SYSTEM menu of an ITF keyboard] while the program is backing up a disc. Be patient during apparent inactivity.

To protect a disc from accidental erasure, write-protect it by following the instructions in the *BASIC User's Guide.* When prompted, insert the master disc and type [CONTINUE] (the `Continue` softkey in the System menu of an ITF keyboard). When prompted, remove the master disc and insert the backup disc, and then continue.

You can do a CHECK READ, but this is seldom necessary. Copy time is increased considerably. If you decide to do a CHECK READ, the utility reads back what was just written to the disc and compares it with what was just read from the master disc. You may have to exchange discs several times. The exchanges are displayed. On an error, the master disc is read again, the disc is written again, and the two are compared a second time. On a second failure, the error is flagged, a message is displayed, and the program stops. You can PRINT the values of the variables `Start` and `Stop` to see the range of addresses within which the error occurred. The CHECK READ and PRINT options are selected by entering `Y` (yes) or `N` (no) and typing [ENTER] or [Return]. On completion, you can do another backup or exit. Choose `Y` or `N` and type [ENTER] or [Return].

Do Not Interrupt Program

Do not type `RUN` while the program is backing up a disc. If you do, type [PAUSE] or [Shift]-[Stop]. Then, start over. Do not use this program with the `REMOTE` msvs (SRM).

(FBACKUP) Selective File Backup

Skim the entire procedure first.

Access the program by executing:

```
LOAD "FBACKUP"
```

Type `RUN` after the asterisk (lower-right corner) disappears.

Wait for the program to go through an initialization process. **Do not** remove the utilities disc until prompted to do so.

To protect a disc from accidental erasure, you can write-protect it by following the instructions in the *BASIC User's Guide*. When prompted, remove the utilities disc and insert the master disc. Have an initialized backup disc available. Type [CONTINUE] (the `Continue` softkey in the System menu of an ITF keyboard).

The master disc is cataloged. If it is necessary, you can use the cursor keys to scroll the screen. Enter the name of a file you wish to backup and type [CONTINUE] (the `Continue` softkey in the System menu of an ITF keyboard). Repeat this procedure for each file you wish to backup (limit is 112 files). Then, type [CONTINUE] (the `Continue` softkey) with no typed file entry to begin the copy process.

Avoid Use With the SRM

Do not use the FBACKUP program with the SRM system, because SRM does not support PHYREC. If you inadvertently type `RUN` while the program is active, type [PAUSE] ([Shift]-[Stop]) and start over.

(TAPEBACKUP) CS/80 Tape Backup

Skim the entire section before you do anything. Backing up files stored on a disc in a CS/80 drive which has tape backup capability is more complex than other types of backup. One backup method involves use of switches and buttons found on the disc drive. BASIC provides two methods.

- The TAPEBACKUP utility program provides backup of the contents of a disc.
- The BASIC File System provides backup of selected files.

TAPEBACKUP

This section discusses only the TAPEBACKUP utility.

Selective file backup is not available strictly within the utility. Some selective file backup is available within the BASIC File System. The TAPEBACKUP utility lets you do a complete backup of an HP 7908, HP 7911, HP 7912, or HP 7914 disc to one or two DC600 tapes; or conversely.

The contents of an HP 7908 disc fit on a DC150 tape. The disc unit number is 0. The tape unit number is 1. The dual controller option is **not** supported. Copying occurs at a rate of 2 Megabytes per minute. The HP 7914 disc requires two DC600 tapes. It can take a few minutes for a tape to load.

Load the CS80 binary if you wish to CATalog the disc or tape. The HPIB or FHPIB binary is required, depending on which interface is installed in your system, but the utility program provides for transfer of data directly from unit 0 to unit 1 without traveling over HPIB.

Access the program by executing:

```
LOAD "TAPEBACKUP"
```

Wait for the * to disappear. Then type [RUN].

The utility provides three options: `COPY`, `VERIFY`, and `CERTIFY`. The `COPY` operation, [USER 1 menu], is called by typing [k0] or [f1]. The `VERIFY` operation, [USER 1 menu], is called by typing [k1] or [f2]. The `CERTIFY` operation, [USER 1 menu], is called by typing [k2] or [f3].

The COPY Operation

On the prompt, enter the HP-IB address for the CS/80 disc. Use `702` and type [ENTER] or [Return] for a drive that has select code 7 and address 2. You must know the **msvs** for each device you use.

Then, you decide to copy from: the **disc to the tape**, or the **tape to the disc**. This is crucial! Enter the name of the **destination** media. For example, to copy from a disc to a tape, type: `TAPE` and type [ENTER] or [Return]. If you select the tape, you are also asked whether you wish to verify the backup. Select `YES` now (via softkey) if you wish to verify because an auto-verify checks only that portion of the tape that was written. A stand-alone verify operation checks the entire tape. Selecting `YES` (to verify) lets you avoid reloading both tapes from an HP 7914 backup to do a stand alone verify.

On the other hand, if you want to copy from a tape to a disc, type: `DISC` and type [ENTER] or [Return].

Since the contents of the destination media are overwritten and lost during the copy operation, here are some particulars you should note:

- Mark the type of drive (e.g., HP 7912) on each tape; especially if you have more than one CS/80 drive.
- Remember that an HP 7914 disc requires two DC600 tapes. Proceed as you would for any other disc; except after the backup with the first tape is completed, remove the tape and insert a fresh one in accordance with the prompt. The backup operation automatically proceeds after the second tape has loaded. For your information, the current PRINTER IS device indicates the starting sector numbers for the source and destination units.
- When copying backup contents of two tapes to an HP 7914 disc, the tapes may be loaded in either order because the starting sector number (technically an address) is written on each tape.
- You can perform an HP 7914 backup with one tape when you know that no data is present on the second half of the disc. Do this by typing [CLR I/O] or [Break] when you are prompted to insert the second tape.
- Errors are reported on the current PRINTER IS device. You can opt to retry the operation or exit.
- When a tape does not verify, do not recertify it. Just repeat the entire copy operation again. The bad blocks detected on the tape are marked and not used again.

- If you copy to an uncertified tape, the tape is automatically certified before the copy begins.
- After recertification, a tape is unloaded. You need to load it again to access it.

Proceed slowly and you should be able to adequately backup your CS/80 discs.

The VERIFY Operation

On the prompt, enter the HP-IB address of your CS/80 disc drive (e.g., 702). Then, decide to verify either a tape or a disc. This is crucial! For example, if you wish to verify the tape, type: `TAPE` and type ENTER or Return. If you wish to verify the disc, type: `DISC`.

For safety, you are asked if you wish to proceed. Enter `Y` and type ENTER or Return to proceed. Otherwise, enter `N`.

If a disc does not verify correctly, reinitialize the disc. If a tape does not verify, repeat the copy process. Do not recertify it. The bad blocks detected on the tape are automatically marked and not used again. Tapes tend to get better with use.

The CERTIFY Operation

Use this operation **only** with tapes. Use this operation when you wish to certify several tapes at one time.

The certify operation works in a manner similar to those described above.

Quitting (EXIT)

Note throughout any operation that you can type k3 or f4 [USER 1 menu] to `EXIT` from the program.

Regarding these operations, you cannot access a CS/80 tape via a TRANSFER statement. But you can use the BASIC File System to copy files to a tape, lists a tape's directory, or store programs on a tape.

You can also use BASIC mass storage statements to access a disc image that was copied to a tape via the TAPEBACKUP program. This is useful for retrieving a single file from a disc image backup without copying the entire tape back onto the disc. A limitation is that you cannot access a file stored on the second tape used to backup an HP 7914 disc.

Selective retrieval must be done with care due to the tape's slow seek times and inability to start and stop rapidly. Each file written to a tape causes the following tape motion:

1. A seek is performed to the beginning of the tape to scan the directory.
2. The entire directory is scanned one block at a time.
3. A seek is performed to somewhere in the middle of the tape to write the file.
4. A seek is performed back to the beginning of the tape to update the directory.

This can take several minutes due to tape movement. Typical times range from 8 to 60 minutes, depending on which CS/80 drive you have.

Consequently, you should not place very large directories on a tape. Something over 80 entries in a directory is large. Avoid doing unnecessary CAT operations. Avoid copying small files. Avoid file copies when little memory is available.

Remember that two lengths of tapes are available; 17 and 67 Megabytes; DC150 and DC600 respectively. You can use either length.

The TAPEBACKUP utility uses some compiled subprograms. These subprograms were generated from compiled code; not from BASIC statements. Consequently, the subprograms cannot be syntaxed within BASIC. This means you cannot modify them via BASIC or access them via a GET statement. You can place a SUB or DEF FN statement directly after the last CSUB statement, but do not use any comments or other BASIC statements.

(PHYREC) Physical Record CSUB

Skim the entire section before you do anything.

PHYREC lets you perform bit-by-bit copies of an integer array in memory to a file on a mass storage media, and vice versa. PHYREC is a CSUB which was created by using the CSUB Utility.

The PHYREC utility contains two CSUBs:

```
10    CSUB Phyread(Sector,INTEGER Int_array(*))
```

and

```
20    CSUB Phywrite(Sector,INTEGER Int_array(*))
```

The `Phyread` CSUB copies data from the media in the current msvs into an integer array. The array can have one to six dimensions. The numeric expression, (e.g., `Sector`), is evaluated to a real and rounded to specify the sector (address) on the disc where the copy begins. The disc is read, beginning with the starting sector, and the data is copied into the integer array in a row-major order. Data is copied until each array element is occupied or until an attempt is made to read beyond the end of data on the media. In the latter case, an error is reported. The implication is that your array should be adequately dimensioned. No pathname is involved. Consequently, the error is trapped by ON ERROR, but not by ON END statements.

The `Phywrite` CSUB works the same way, except that data is copied from the array to the media and an error is reported on an attempt to write beyond sector 1055.

Be Aware of OPTION BASE

The usual OPTION BASE circumstances apply to how PHYREC uses arrays. Be sure to use 0 or 1 as required by the nature of your data.

You must load PHYREC into memory before a CSUB is executed (called). Execute:

```
LOADSUB ALL FROM "PHYREC"
```

When PHYREC CSUBs have been loaded into memory, your main program or another subprogram can make calls via `Phyread` and `Phywrite`. One way to see this is to load and list the CAT subprogram. The last two program lines of CAT are:

```
950   CSUB Phyread(Sector,Int_array(*))
960   CSUB Phywrite(Sector,Int_array(*))
```

Other subprogram lines call these subprograms. An example is:

```
100   Phyread((Dstart),A(*))
```

Study the CAT subprogram to conceptualize how PHYREC, CSUB, and Phyread are used. Also, study the CSUB Utility and the *BASIC Language Reference* manuals.

You Need To Be An Expert

Considerable programming savvy and computer experience are required to effectively use PHYREC. Do not be put off, but do take time to acquire skill in using HP BASIC.

CAUTION

INDISCRIMINATE USE OF `Phywrite` CAN CAUSE LOSS OF VALUABLE DATA.

You need to be aware of many things when you use PHYREC. Some considerations, programming hints, and tables follow. The information is not comprehensive. You need to determine how the information is useful.

- The integer array that receives disc data should be dimensioned to minimize disc access. For example, to read the contents of a file composed of 10 physical records (sectors), use a 10 by 128 array with option base 1 where 10 is the number of records and 128 is the size of the words in a physical record.
- You need to understand disc structure to effectively use the PHYREC CSUBs. For example, a 5.25-inch flexible disc contains 66 tracks with 16 addressable sectors per track. Track numbers are not typically used. Thus, the sector addresses range from 0 to 1055. The contents of a sector are important. Its location is not because PHYREC only accesses addressed sectors.

- The contents of a LIF directory on a disc may differ from information displayed when you execute a CAT command because the CAT process interprets some directory information and displays it in a more informative manner.
- A non-5.25-inch disc, initialized by BASIC, probably contains a directory length and file space different from the specifications given in the following *Disc Contents* table. You need to know this information before you use PHYREC.

The following tables contain information about the contents of discs and the differences between directory and CAT values. Use them to conceptualize how information is stored on a disc.

Disc Contents

```
Sector 1, An empty sector

               0 ..........................................  0

   Words:               0 through 127

Sectors 2 through 15, The LIF directory sectors

Sector 2     File entry  File entry  File entry ...  File entry
                 1           2           3               8

Sector 3     File entry  File entry  File entry ...  File entry
                 1           2           3               8

 .
 .
 .

Sector 15    File entry  File entry  File entry ...  File entry
                 1           2           3               8
```

Note: Each file entry contains 8 pieces of information stored in words 0 through 15. The 8 pieces of information are:

Respective words used are:

File name	File type	File address	File length	Time of creation	Volume number	Protect code	Defined record size
0-4	5	6-7	8-9	10-12	13	14	15

1) The file name is five words; up to 10 characters.
2) The file type is one word (numeric) as follows:

1	is ASCII
0	is Empty
-1	is Logical end of directory
-5775	is BIN
-5791	is BDAT
-5808	is PROG
-5822	is SYSTM

3) The file address is two words (numeric); address at which file contents begin.
4) The file length is two words (numeric); numbers of sectors spanned by file's contents.
5) Time of creation is three words set to 0 and not currently used.
6) Volume number is one word; set to -2^15 + 1, not currently used.
7) Protect code is one word; up to 2 characters, 0 for ASCII, defaults to blank for other file types.
8) Defined record size is one word; 0 implies BDAT file with record size of 1 byte, ignored for ASCII but set to 0 for other system compatibility, ignored for PROG or binaries but set to 128 for compatibility with future products. It may appear on the display as 256.

Note: The directory start address is 2 for 5.25 inch discs. See value of Dstart in procedure for CAT subprogram. The directory length is 14 for 5.25 inch discs. See the Dleng value in the same procedure.

Note: The volume labels for 5.25 inch discs are B9826 or B9836. These can be changed, or can be different for other discs.

Note: The actual content of all files (file space) lies in sectors 16 through 1055.

Differences Between LIF Directory and CAT Values

```
Field                File Type          Directory Value     CAT
Value
---------------------------------------------------------------------

file type  .......  ASCII  ................. 1  .......... ASCII
                    Empty entry  ........... 0
                    Logical end of  ........ -1
                        directory
                    BDAT  .................. -5791  ....... BDAT
                    BIN  ................... -5775  ....... BIN
                    PROG  .................. -5808  ....... PROG
                    SYSTEM  ................ -5822  ....... SYSTM

file address  ....  BDAT  .................. n  .......... n + 1
                    All others  ............ n  .......... n

file length  .....  BDAT  ........... No. sectors  .....  No. sectors
                                                         user sets
                                                         at creation.
                                                         Read words
                                                         4 and 5 of
                                                         system sector.
                                                         Stores as
                                                         first sector
                                                         of file.

protect code .....  ASCII  ................ 0
                    SYSTEM  ......... Upper word start
                                      address
                    Others  ......... 2 character code

defined record  ..  ASCII  ................ 0  ..........  256
     size           BIN  ................ 128  ..........  256
                    PROG  ............... 128  ..........  256
                    BDAT  ............ If n=0  ..........    1
                                       If n>0  ..........   2n
                                 (length in words) ..   (length in
                                                         bytes)
                    SYSTM  ........... Lower word .......  256
                                       start address
```

Again, it is helpful to learn how to use PHYREC by studying how it is used in other subprograms and by studying the *CSUB Utility* manual.

(LISTER) List Files

If using a printer, load the HPIB binary. Do **not** use the utility with the REMOTE msvs. Use of 8.27 by 11.69 inch paper may require alteration of lines 490 and 500 of LISTER (see last paragraph in this section)

Access the program by executing:

```
LOAD "LISTER"
```

Wait for the asterisk (lower-right corner) to disappear. Then, type RUN.

The program goes through a short initialization process. On the prompt, insert a disc that contains files for which you want program listings. ASCII files are required (they are created by a SAVE or CREATE ASCII statement). Then, type [CONTINUE] (the **Continue** softkey on the System menu of an ITF keyboard) to display a catalog of the disc.

On the prompt, enter a file to be listed and type [CONTINUE] (the **Continue** softkey in the System menu of an ITF keyboard). You can enter up to 112 files. Continuing without entering a file name lets you proceed. All entered files are printed according to the format you establish.

The format for printing a program listing is established via a list of displayed parameters that you can alter. Use the knob (if available) to pick a parameter and then type the SELECT softkey. If you have an ITF keyboard (such as the HP 46020A), toggle between the User 1 and User 2 menus and then pick a parameter by typing the corresponding softkey. For example, type [f3] (in the User 1 menu) to alter **Paging**. Here are the parameters. They are either displayed on the screen and accessed via the knob, or displayed on the screen and accessed via softkeys.

Device Selector The default is 701. You can enter a different device select code and type [ENTER] or [Return].

Paging A toggle parameter (YES or NO). **YES** uses 11 inch pages. NO uses continuous printing that breaks between files. Just press the softkey.

Perforation A toggle parameter (YES or NO). **YES** assumes perforated (fan-fold) paper. NO causes dashes to be printed at points where you should cut the paper. Just press the softkey.

Lines per page The default is 63 lines. You can alter this to change the top and bottom margins. Just follow the prompts.

`Spacing` Another toggle parameter (SINGLE or DOUBLE).

`Omit page numbers` Another toggle parameter. The default is NO which retains page numbers. YES omits.

`First page number` The default is 1. You can alter this. Just follow the prompts.

`Print range` You can select low and high page numbers. ALL prints everything, which means the entire program is listed. ALL can be the high number (e.g. enter 12 as the low number and enter ALL as the high number). Then, the program is listed from the low page number to the end of the program. Execution is slowed when you print a small listing in the middle of a large program, because the program goes through the printing process for each line even though only those in the specified range are printed. To specify ALL, select the function, type ALL, and then type ENTER or (Return).

`Trailer` You can enter a string which is printed at the bottom of each page next to the page number. DO NOT PHOTOCOPY is an example of a trailer.

`Edit text` When set to YES, line numbers and exclamation marks (!) are deleted from the file. For example:

```
10 ! Great ideas
20 ! Are often simple
```

changes to:

```
Great ideas
Are often simple
```

`Width` The default is 80 characters. You can change this in the range from 1 to 132.

`No. of listings` The default is 1. You can change this as you require. Just remember that beyond 2 or 3, it is much faster to use a copy machine.

`EXIT` Exits from the utility.

After you have selected the print format, begin printing by typing the START PRINTING softkey (User 1 menu on the ITF keyboard). On the prompt, line up the top of form for fanfold paper. Then, type CONTINUE (the Continue softkey on the System menu of an ITF keyboard).

LISTER is programmed to have 70 maximum lines per page. This works with an HP 2631A/G printer. If you have an HP 9876 printer and use 8.5 by 11 inch paper, you have 74 maximum lines per page. If you use 8.27 by 11.69 inch paper, you have 74 maximum lines per page. Thus, if 70 lines per page will not work and you need 74 lines per page, load LISTER and EDIT lines 490 and 500 so they read:

```
490     Maxlines=74
500     Ff_space=10
```

Be sure to `RE-STORE "LISTER"` if you want to make the change permanent.

(82905DUMP) The 82905B Dump Graphics Subprogram

An HP 82905B printer is required. You may need to load the HPIB, GRAPH, and GRAPHX binaries. The subprogram **does not** work with any bit-mapped display (such as the Series 200 Model 237 or a Series 300 bit-mapped display).

You can access this subprogram by creating an appropriate calling program. An example calling program is provided shortly. Remember that a sequence is required. You need to:

1. Load and run a program which contains the graphics you wish to dump to get the graphics image onto the CRT.
2. Load a calling program that can access the dump utility. It is probably better to create the calling program, store it as a PROG file, and then load it when you want to do a graphics dump. Here is the example calling program.

```
10     ! Program to Call 82905DUMP subprogram
20     ! Enter device select code
30     INTEGER Device_selector
40     INPUT "Enter printer select code (e.g. 701)",Device_selector
50     ! Call the subprogram
60     CALL Graphics_dump(Device_selector)
70     ! Reset system defaults
80     PRINTER IS 1
90     GRAPHICS OFF
100    OUTPUT 2 USING "#,K";CHR$(255)&"K"   ! Clear the screen
110    PRINT "The dump is completed.  Proceed as you wish."
120    END
```

3. Append the dump utility to your calling program. Execute:

```
LOADSUB ALL FROM "82905DUMP"
```

   You can execute `EDIT` to confirm this.

Now, you should be ready to dump the graphics image. Adjust the paper in the printer so that the left margin is the printer's left margin. Then, run the program. No interaction is required. The graphics display is dumped to the printer in about three minutes. Notice that the last few lines of the calling program reset your system to its original default conditions.

(GDUMP_R) Rotated Graphics Dump

.index GDUMP_R

This CSUB utility dumps graphics raster images to a printer. It provides the same function as the DUMP GRAPHICS statement, except that it "rotates" the image 90 degrees before sending it to the printer.

Here is an example of calling the routine to dump the image from the display raster (PLOTTER IS CRT,"INTERNAL") to a printer at device selector 701.

```
400  LOADSUB ALL FROM "GDUMP_R" !  Load the CSUB into memory.
410  !
420  Gdump_rotated(CRT,701)     !  Dump raster (select code 1) to
430                             !  HP-IB printer (select code 7,
440                             !                 address 1).
```

Here is another example. This time, the raster at select code 6 is sent to a printer at select code 9 (there is no address, since this is a serial interface).

```
560  Gdump_rotated(CRT,9)       !  Dump raster (select code 1) to
570                             !  serial printer (select code 9).
```

You can delete the CSUB from memory by executing this statement:

```
DELSUB "Gdump_rotated"
```

(BPLOT) Raster Store and Load

This CSUB utility lets you store and load rectangular "blocks" of raster data (using numeric arrays). It can be used on all graphics frame buffers that are supported by BASIC 4.0 or later versions.

This utility provides the Bstore and Bload CSUBs, which are similar to GSTORE and GLOAD statements with the following differences:

- Bstore and Bload affect **only** a specified portion of the frame buffer; GSTORE and GLOAD affect the entire buffer.
- Bstore and Bload also allow you to specify a replacement rule (logical operation such as AND, OR, and EXCLUSIVE OR) to combine the source and destination pixels; GSTORE and GLOAD are always dominant (i.e., they overwrite any existing destination pixels).

Subroutine `Bstore` stores a rectangular area of the frame buffer in an INTEGER array. Here is an example of using the routine:

```
Bstore(Int_array(*),X_pixels,Y_pixels)
```

The INTEGER parameters `X_pixels` and `Y_pixels` specify the width and height of the rectangular raster area to be placed into the INTEGER array `Int_array` (*in pixels*, not in the current graphics unit of measure). The data is stored in the array as one byte per pixel for all frame buffers[1]. This array variable must be of sufficient size to store the specified pixels, or an error is reported (error 16—improper dimensions).

If part of the area is outside the current clip limits, then only the portion of the frame buffer within these limits will be stored in the array—the remainder of the array will remain unchanged.

Here is another example of calling the routine, this time with an optional INTEGER parameter that specifies how the pixels are placed into the array.

```
Bstore(Int_array(*),X_pixels,Y_pixels,Rplcmt_rule)
```

[1] Note that displays with non-square pixels (i.e. medium-resolution Series 300 displays) require twice as much array variable space as their square-pixel counterparts.

The optional parameter **Rplcmt_rule** specifies the replacement rule to use in combining:

- Source bits (frame buffer with **Bstore**; array with **Bload**)

  with

- Destination bits (current contents of the array with **Bstore**; frame buffer with **Bload**).

Note that these replacement rules correspond to the rules provided by CRT CONTROL register 14 for bit-mapped displays.

Parameter Value	Effect on Destination Bits
0	All bits set to 0
1	Source AND Destination[1]
2	Source AND NOT Destination
3	Source[1] ← (Default)
4	NOT Source AND Destination[1]
5	Destination
6	Source EXOR Destination[1]
7	Source OR Destination[1]
8	Source NOR Destination
9	Source EXNOR Destination
10	NOT Destination
11	Source OR NOT Destination
12	NOT Source
13	NOT Source OR Destination
14	Source NAND Destination
15	All bits set to 1

If no replacement rule is specified, then rule 3 (the default) is used.

[1] Only rules 1, 3, 4, 6 and 7 are allowed on non-bit-mapped hardware (Models 216, 217, 220, 226, 236A/C, and the 98627 display interface card). All rules are available on Series 200 Model 237, all Series 300, and on the 98700 Display Controller.

Here is another example that shows two more optional REAL parameters, **X_start** and **Y_start**, which specify the upper left corner of the area to be stored:

```
Bstore(Int_array(*),X_pixels,Y_pixels,Rplcmt_rule,X_start,Y_start)
```

The **X_start** and **Y_start** parameters are in *current graphics unit of measure* (not in pixels). If these parameters are not specified, then the current graphics position is used.

If you want the source bits to be placed into the destination without modification, then specify a value of 3 for the replacement rule parameter.

Subroutine **Bload** loads a rectangular area of the frame buffer from an INTEGER array. Here is an example of calling the routine:

```
Bload(Int_array(*),X_pixels,Y_pixels,Rplcmt_rule,X_start,Y_start)
```

This statement loads a rectangular area on the frame buffer **X_pixels** wide and **Y_pixels** high with the current contents of the INTEGER array **Int_array**.

The optional REAL parameters **X_start** and **Y_start** specify the upper left corner of the destination (in *current graphics unit of measure*, not in pixels). If these parameters are not specified, then the current graphics position is used.

The optional INTEGER parameter **Rplcmt_rule** specifies the replacement rule to use in combining the array elements with the current contents of the affected frame buffer area. If part of the area is outside the current clip limits, then only the portion of the array that maps into the current clip limits will be loaded into the frame buffer—the clipped area of the frame buffer will remain unchanged.

(LEX_AID) Creating a Lexical Order Table

The LEX_AID program simplifies the creation of user-defined lexical tables. You may wish to make modifications to the program to suit your particular needs. The program and default lexical order tables for French, German, Spanish, and Swedish are provided on the *BASIC Utilities 2* disc[1]; the names of these BDAT files are FRENCH, GERMAN, SPANISH, and SWEDISH, respectively.

Steps in Creating a Table

1. Use the LEXICAL ORDER TABLE WORKSHEET in the "User-defined LEXICAL ORDER" section of the "String Manipulations" chapter of *BASIC Programming Techniques.* To avoid confusion, always assign sequence numbers in ascending order. Also mark special cases ("1 for 2", "Don't Care", etc).
2. When you have completed your lexical order table, scan it for blocks of consecutive sequence number assignments. For example, the control characters (codes 0 through 31) generally retain their original sequence number. The LEX_AID program has a "FILL BLOCK" mode that will simplify assigning groups of sequence numbers.
3. Run the LEX_AID program to create your table. (Be sure to save your table when you are finished.)
4. Install the table created by this program.

The remainder of this section further expands steps 3 and 4.

Running LEX_AID

LEX_AID uses the softkeys to provide menu selections. One of the following main menus appears on the screen when the program is run.

Key Labels for ITF (HP 98203A/B/C) Keyboards

```
Seq    Mode   Show   Show     Show   Fill     Get    Quit   Save   List
Num    Index  Seq    Mode     Table  Block    Table         Table  Table
```

Key Labels for Non-ITF Keyboards (e.g., HP 98203A/B/C)

```
Seq Num    ModeIndex  Show Seq   ShowMode   ShowTable
FillBlock  GetTable   Quit       SaveTable  ListTable
```

[1] With double-sided media, *BASIC Utilities* disc contains the contents of both the *BASIC Utilities 1* and *BASIC Utilities 2* discs.

For either keyboard, the options work as follows:

- **Seq Num** Allows a sequence number to be assigned to a character. The program prompts for the character, and then for the sequence number to be assigned. Whenever the program is prompting for a character, either the character may be typed from the keyboard or its decimal value may be entered.
- **ModeIndex** Displays a sub.menu (shown later) from which the special mode entries can be selected.
- **Show Seq** Prompts for a character then displays the character, its value and its current sequence number.
- **ShowMode** Displays all of the currently defined mode table entries.
- **ShowTable** Displays the current assignments on the CRT. The mode type and mode index are displayed as a single value to save space.
- **FillBlock** Prompts for the beginning sequence number and the first and last characters of the block to be filled. Thus a group of characters can be assigned consecutive sequence numbers quickly.
- **GetTable** Loads a previously defined lexical order table from the disc.
- **Quit** Returns you to normal keyboard mode and terminates the program.
- **SaveTable** Saves the currently defined lexical order table to the disc.
- **ListTable** Prints the currently defined lexical order table to a printer on the HP.IB (device selector 701).

Requesting `ModeIndex` displays one of the following menus.

Key Labels for ITF (HP 98203A/B/C) Keyboards

```
Don't  1for2  2for1  Accent   Normal
Care
```

Key Labels for Non-ITF Keyboards (e.g., HP 98203A/B/C)

```
Don'tCare  1for2      2for1      Accent     Normal
```

Note that the keys shown without labels (on these drawings) may have previously defined "typing-aid" definitions and labels.

If you are not familiar with these mode types, you may wish to review the aforementioned "User-defined LEXICAL ORDER" section. With either keyboard, the options work as follows:

- **Don'tCare** Requests the character to be ignored for collating purposes.
- **1 for 2** Asks if a suitable entry already exists. If the current character can use a previously defined "1 for 2" entry, the same mode table entry can be used. Answering "Yes" to the question prompts for the new character and then prompts for the character which already has the proper entry. Answering "No" to the question prompts for the character and then asks for the number of secondary characters. Each character and its sequence number is then entered.
- **2 for 1** Asks if a suitable entry already exists. If several characters use the same secondary character (as in the GERMAN lexical order), the same mode table entry can be used for more than one character. Answering "Yes" to the question prompts for the new character and then prompts for the character which already has the proper entry. Answering "No" to the question prompts for the sequence number of the second character (both upper and lower case).
- **Accent** Requests the priority (0 through 63) and then the character to be assigned the accent priority.
- **Normal** Allows you to cancel a special mode entry that was mistakenly assigned to a character. No provision is implemented in the LEX_AID program to actually delete the mode table entry.

With practice, you should be able to create user-defined lexical order tables with the help of LEX_AID. Be sure to save the table when you are finished.

Installing Your Lexical Order Table

Having created a table and stored it on disc, you can install it with the following BASIC program.

```
10    INTEGER Table(0:320)
20    ASSIGN @File TO "MYTABLE"  ! Open file
30    ENTER @File;Table(*)       ! Read file
40    ASSIGN @File TO *          ! Close file
50    LEXICAL ORDER IS Table(*)
60    !
70    END
```

You may need to replace the name "MYTABLE" with the name of the file in which you have stored the table you have created with LEX_AID.

The lexical order will remain in effect until a SCRATCH A or another LEXICAL ORDER IS statement is executed, or the BASIC system is re-booted.

A LEX_AID Example

The following example shows how to use the LEX_AID program to create a case-independent lexical order. Both uppercase and lowercase characters collate together.

1. Load and run the LEX_AID program.
2. Type the **FillBlock** softkey. Enter 0 for the initial sequence number, 000 for the value of the first character, and 255 for the value of the last character. As each character is assigned a sequence number, the character will be flashed on the screen.
3. Type **FillBlock** again. Enter 65 for the starting sequence number, 097 for the value of the first character, and 122 for the value of the last character.
4. Type **SaveTable** and save the table under the name "NOCASE". The newly created table will be saved on the disc as a BDAT file.
5. Execute `SCRATCH`, then enter and run the following program:

```
10     INTEGER Table (320)
20     ASSIGN @File to "NOCASE"
30     ENTER @File;Table(*)
40     LEXICAL ORDER IS Table(*)
50     IF "Hello" = "hElLO" THEN
60       PRINT "NO DIFFERENCE"
70     ELSE
80       PRINT "LEX DOESN'T WORK"
90     END IF
100    END
```

The message— `NO DIFFERENCE`— is printed. Similar results could have been achieved by using the UPC$ and LWC$ functions.

Applications needing specialized collating sequences can now be simplified by the use of the LEXICAL ORDER IS statement.

When a lexical table needs to be created (by a program such as LEX_AID), it is often easier to separate the table into two arrays, one for the sequence number and a second for the mode entry. This simplifies the calculations of the mode entry. The two arrays are then merged into a single INTEGER array. You may wish to list the LEX_AID program to see the operation.

The Status Utilities

Four utility programs are available that let you see the contents of status registers: Data Files (FILE_STAT), HP-IB Interface (HPIB_STAT), RS.232 Interface (RS232_STAT), and GPIO Interface (GPIO_STAT). Each program works in a similar manner. Each requires that you enter fundamental information about such things as device select codes and mass storage unit specifiers.

The best way to learn how to use these programs is to load them and call assorted functions via softkeys. (Remember to toggle between the ITF-keyboard User 1 and User 2 menus.) Be sure to load appropriate binaries; for example, load the GPIO binary when you use the GPIO_STAT utility.

Pay attention to softkey labels. They are completely self-explanatory. Use them for calling functions. If an interface is not present, type [ENTER] or [Return] with no file entry to exit. Execute any of the four utilities with:

```
LOAD "FILE_STAT"
```

or

`LOAD "HPIB_STAT"` *(works* ***only*** *with an HP-IB interface)*

or

```
LOAD "RS232_STAT"
```

or

```
LOAD "GPIO_STAT"
```

Unlike utilities which can be rather tedious to use, these four are simple and friendly. Yet they can provide valuable information about the contents of Status Registers.

(MEM_UTILS) Memory Utilities

MEM_UTILS requires the KBD, MAT, MS, IO, and CRTX binaries. You also need at least an 80 column alpha display. The memory volume— `":,0,15"`— created by MEM_UTILS uses about 11 Kbytes of memory.

To use any of the five memory-resident soft-key accessed utility subprograms provided by MEM_UTILS, insert the appropriate utilities disc into your flexible disc drive. Do a CAT if you need to find out which disc the program is on. Then, substituting an appropriate volume specifier for *msvs*, load the utility into memory by typing:

`LOAD "MEM_UTILS`*msvs*`"` Return

RUN the program and wait for it to display the typing aids. Before you do anything, note the following items:

- The subprograms in MEM-UTILS are stored in a memory volume, are loaded and called by a typing aid, and delete themselves from program memory when they are done.
- A program that will prerun successfully must be present in memory for any of the five utilities to run (the utilities are subprograms). In addition, if the program has COM declarations, the specified common must exist in memory. Attempting to invoke a utility when one of these conditions is not met causes an error and leaves the utility at the end of your program (use DELSUB to remove it). If you have done a SCRATCH, a typing aid defined in the User 3 menu will create the following trivial program that can serve as **a program**:

  ```
  10 END
  ```

- It is recommended that any other program or utility, present in memory, be at least paused before invoking one of the MEM-UTILS utilities. They can be invoked from the EDIT mode, but doing so causes a paused program to go to the stopped state (this does not happen outside of EDIT mode). Also, when the utility finishes, the system will not return to the EDIT mode.
- A MEM_UTILS utility should be allowed to complete and exit normally so it will be deleted from program memory.
- The typing aids are assigned to softkeys in the ITF User menus (note plural). When using MEM_UTILS with an HP 98203A/B/C keyboard, you might want to modify the key numbers in lines 20 through 70 to put the typing aids on the softkeys of your choice. You might also want to delete line 80 and use `10 END` Return instead of using a typing aid to create the program.

- The utilities use LINPUT, which means it is not necessary to put quotes ("") around responses that contain commas as part of a mass storage volume specifier (msvs). Doing so results in an error.
- The end of each utility has an OUTPUT KBD which contains two 'Alpha' keys. These "closure" keys delay the 'DELSUB' command until after the 'SUBEND' statement has executed. On some computers, these two keys turn graphics off. If you wish to avoid this, replace these keys with some other "closure" keys. For example, the Roman and Katakana keys would be good choices for computers that do not have Katakana keyboards.

The Cat by Pages Utility

Selecting **Cat by Pages** displays an 80 column CAT for a specified mass storage device. The program uses the default directory if you do not enter one on the prompt. The program pauses for each screenful of files until you type the key for [CONTINUE]. Any error message appears in the DISP line, and the program exits. To exit early, rather than continue, type the key for [PAUSE] and then type:

```
CONT Cat_done [Return]
```

The Append Files Utility

Selecting **Append Files** lets you COPY all or selected files in one directory to another directory (but is not a volume-to-volume copy) without losing what is already in the other directory. You can copy all files or be asked about each file. The utility detects existing files that have the same name and asks if you want to PURGE the conflicting file. If you do not wish to do the purge, or if the file name is inappropriate (e.g. trying to copy a 16-character SRM file name to LIF media, which has a 10-character limit for file names), you are asked to enter a new file name. Any error messages appear in the DISP line, and the program exits. To exit early, type the key for [PAUSE] and then type:

```
CONT Append_done [Return]
```

The Purge Files Utility

Selecting **Purge Files** lets you PURGE all or selected files in a directory. You can choose to verify each purge before the file is removed or have the files removed without individual verification. The program detects and skips open or protected files. Error messages appear in the DISP line, and the program exits. To exit early, type the key for [PAUSE] and then type:

```
CONT Purge_done [Return]
```

The Print File Utility

Selecting **Print File** lets you enter a file name and a printer device selector. Then, the program copies the file to the printer. The file can be an ASCII file, or a (FORMAT ON) BDAT or HP-UX file. Both BDAT and HP-UX files can be PRINTER IS files, which are FORMAT ON. Lines cannot be longer than 160 characters. Error messages appear in the DISP line, and the program exits. To exit early, type the key for [PAUSE] and then type:

```
CONT Print_done [Return]
```

The Take-a-Memo Utility

Selecting **Take a Memo** lets you create a 60 line by 80 column page of text. The program uses inverse video to identify the "current" line; therefore, you should avoid using the program with the HP 98204A monitor, which does not have inverse video. Note that, if you want to edit larger amounts of text, you can find **SUB Kled** and edit the **61**s in the next two lines— **Tmax=61** and **Text$(61)[80]**— to be the desired number of lines plus one (keeps two buffer lines).

The following items explain how the memo editor works. Take time to read each item, and then use the editor according to your needs.

Stopping

Type [Stop] while the utility is running to exit; typing [Stop] while the utility is paused does not remove it from memory. **Note that it quits without asking if you have saved your work to a file**. You can also exit by typing the key for [PAUSE] and then typing:

```
CONT Take_done [Return]
```

Editing

You can edit two lines of text in the keyboard lines by using the same line editing keys that you use in editing BASIC programs. On typing [Return], the first 80 characters of the "keyboard" line overwrite the "current" line. If characters remain from the "keyboard" line, after the first 80 characters are written into the "current" line, the utility attempts to insert blank lines into the text after the current line and write the remaining characters into them. The new "current" line is the next line after the newly entered text. If the total text will overflow 62 lines, the utility beeps and leaves the rest of the new text in the keyboard line.

Scrolling

You can scroll text through the "current" line window with the up/down arrow keys, a knob, a mouse, and the [Prev] and [Next] keys.

Getting a line

You can move a line of text into the "keyboard" line by making it the "current" line and typing [Select] (type [EXECUTE] if you have an HP 98203A/B/C keyboard). Doing this does not clear the keyboard line; and this feature lets you concatenate two lines or insert one line into another.

Deleting

Delete the current line by typing [Delete line].

Inserting

Insert a blank line into the text above the current line and move the blank line into the current-line window by typing [Insert line]. This works unless the insertion would cause the text to overflow 62 lines.

Recalling

The utility has two recall buffers, and each one is two lines deep. The "delete buffer" contains the last two lines deleted by typing [Delete line]. Retrieve these lines by typing [RECALL]. The "enter buffer" has the last two lines that were **overwritten** by typing [Return]. Retrieve these lines by typing [SHIFT] [RECALL].

Write file

Write text to an ASCII file by typing [Print] or [DUMP ALPHA]. Enter a file name on the prompt. If necessary, you are asked if you want to overwrite an existing file. An existing file must be able to hold the text to be saved (62 lines by 80 columns). The overflow buffer is included in the text written to the file.

Read file

You can read a file into the utility by typing [RESULT] (or [EDIT] on an HP 98203A/B/C keyboard, but note that the `EDIT` typing aid on the ITF keyboard just types `EDIT` into the keyboard line). You are asked to enter a file name.

Update display

On reading or writing a file; on getting an error message; or on continuing after a pause; you will need to update the display by typing the key for [CONTINUE].

Line No.

The first number in the DISP line is the current line number (1 to 62) in the memo. The second number is the number of characters in the line. This second number is useful for telling a line of blanks from a null line. (A line of X blanks takes up X+2 bytes in a file when the text is saved— a null line uses only 2 bytes.)

Constraints

While the text can be 62 lines, the last two lines are intended as an overflow buffer. Either line can become the current line via entering lines 61 or 62, or by using the up-arrow key, but line 60 is treated as the default last line of text. An error occurs when you read a file containing less than 62 lines of text; type the key for CONTINUE and the text will be there as will any previous text that was not overwritten— the array is not cleared before a read. Reading a file of more that 62 lines does not cause an error, but the read stops at line 62.

The file

The utility creates an ASCII file of 20 records (62*80 or 4960 characters). However, an existing file need only be large enough to hold the current text to be successfully written.

(FONT_ED) Font Editor

This utility provides a method of creating and editing the bit patterns of character fonts usable with BASIC systems.

Displays with "Soft Font" Capabilities

Soft fonts can be used with the following displays on which the alpha and graphics share the same raster—usually called "bit-mapped alpha displays":

- Series 200 displays—**only** the Model 237 display.
- Series 300 displays—**all** displays **except** the HP 98546 Display Compatibility Interface.

This display architecture makes possible the definition of custom characters (and entire fonts); and, in addition, you can change them under program control.

The "soft-font" display architecture and corresponding BASIC language capabilities are described in the "Custom Character Fonts" section of "Communicating with the Operator" chapter of the *BASIC Programming Techniques* manual. You may need to be familiar with this information to fully understand how to use this utility.

Fonts Available from Your Local HP Sales Office

Besides the default font that is in the machine at start-up time, the following fonts are available from HP Sales Offices:

- The US/European font for *medium*-resolution displays.
- The US/European font for *high*-resolution displays.
- The Katakana font for *medium*-resolution displays.
- The Katakana font for *high*-resolution displays.

These fonts and their availability are subject to change.

Restoring the Default Soft Font

The Font Editor lets you "get" and "edit" characters in character fonts. Should you want to restore the default font (the one in place when BASIC is booted), type:

```
CONTROL CRT,21;0 [Return]
```

The statement re-initializes the alpha display to its power-up state, and this operation includes changing back to the power-up ("default") character set. (This is also possible with the `Default Font` option of the utility. The CONTROL statement was provided in case the utility program encounters an irrecoverable error and stops running.)

Loading the Font Editor

Insert the *Manual Examples* disc into your flexible disc drive. Using the appropriate volume specifier for *msvs*, load the Font Editor into memory by typing:

```
GET "FONT_ED:,msvs" [Return]
```

Run the program by typing:

```
RUN [Return]
```

On running the program, the following prompt asks you to specify the size of the character cells.

```
What size character cells will you be editing?
```

You will see a default character size of `8, 16` in the keyboard line.

You can accept the default size by typing [Return]. Otherwise, specify the other possible character-cell size by typing:

```
12,15 [Return]
```

Font Editor Options

On running the Font Editor, you get the following aids in the softkey labels:

`Get Font` Reads a specified font from a file (created by this utility) into an array in memory. Use other options to edit, save, or install the font.

`Edit Char` Lets you edit bit patterns of characters in the font.

`Save Font` Saves the edited font to a file.

`Install Font` Loads the current font into memory.

`Default Font` Restores the default font (the one that was in memory at the time that the BASIC system was booted).

`EXIT` Clears the display and exits the utility.

Using the “Get Font” Option

Selecting `Get Font` prompts you to specify a file that contains a font. The file can be a font supplied with the system, or it can be a “font-file” that you created using this utility.

```
Please specify the file that contains the font to be edited.
```

Enter the file specifier of the desired font file, using both a directory path and an msvs as appropriate (again, do a CAT on your utilities discs as required by your situation if you need to find the disc that has the font-files).

If you should specify a file that does not exist, or if another error occurs while attempting to read a font file, the program displays this message:

```
Error nn occurred while trying to read the file. Please try again.
```

You can then correct the error (such as re-type the file specifier), and the program attempts the operation again.

If the file is found and contains the proper data, the program displays the codes of the characters as they are being read.

```
Reading character: 1
```

After all characters have been read, you briefly see the following message; and then, you can choose another option from the menu (e.g. edit characters, install the font) according to your needs.

```
Finished reading characters.
```

Using the “Edit Char” Option

Selecting `Edit Char` prompts you to enter the code number (0 through 256) of the character for the font you want to edit. (The numbers correspond with ASCII code; and for example, typing **65** Return lets you edit the font for an upper-case A.)

```
Enter code of character to be edited.
```

The utility then displays a matrix of pixels, with a cross-hair on the upper-left character; and the softkey menu changes to:

```
Accept                    Toggle
Shape                     Pixel
```

Use the arrow keys to move the cross hair onto a pixel you want to change. Then, typing the key for **Toggle Pixel** turns the pixel from off-to-on **or** from on-to-off. Repeat this process until the shape is acceptable and then type the key for **Accept Shape** to stop editing the character cell. Note that, if you are working on a bit-mapped display, the program displays the character and asks you if it is "acceptable". If it is, type Return. if it is not, type N and the program returns to editing the character cell.

Using the "Save Font" Option

Selecting **Save Font** prompts you to type the range of numbers for characters whose fonts you want to save. The total range is 0 through 256, but you can specify a sub-range. Accept the default by typing Return, or type the range you want and type Return.

```
Enter the range of characters to be saved.
 0, 256
```

You Can Get Blank Characters

Characters not defined by **Get Font** or modified in the **Edit Char** options will be "blank" characters.

The program displays the code of the character currently being saved in the file:

```
Saving character:  1
```

After all specified characters have been written to the file, you briefly see the following message and then get back the menu that lets you select options.

```
Finished saving characters.
```

Using the "Install Font" Option

Selecting **Install Font** lets you begin using the characters you read with **Get Font**. Note that if you edited any of the characters earlier, the edited versions are the ones that will be loaded.

The program prompts you indicate if overwriting an existing fault is acceptable (see the descriptions below before you type something):

```
Installing a font overwrites the existing font.  Is this acceptable? (Y/N)
```

Type Y Return to begin using this font; and type N Return otherwise.

On entering N, the program returns to the main menu. On entering Y, the program asks for a range of characters to install (the default range is the entire character set— 0, 256). Edit the range (if necessary), and type Return.

```
Enter the range of characters to be installed.
 0, 256
```

The program then installs the new font, and displays the following message before returning to the menu of options:

```
New font installed.
```

(Note that if you don't like this new font, or it was installed in the wrong character-code range, you can correct the problem by selecting the **Default Font** option described below.)

Using the "Default Font" Option

Selecting **Default Font** prompts you to indicate if it is acceptable to restore the default font—the one that BASIC was using when the computer was powered-up (see the alternative below before you type something):

```
Restoring default font overwrites existing font.  Is this acceptable? (Y/N)
N
```

On entering N, you briefly see the following message and then get the menu of options:

```
Default font restoration aborted.
```

On entering Y, you briefly see the following message and then get the menu of options:

```
Power-up font restored.
```

Using the "EXIT" Option

Selecting `EXIT` displays the following message, exits the Font Editor, and restores the graphics and alpha displays to their power-up states.

The Loader Utility

This section discusses the Loader utility. The Loader Utility consists of three files: **SYSTEM_LD**, **CONFIGER**, and **CONFIG_CHK**; and in general, the utility lets you boot a preselected operating system without operator intervention. Since there are several versions of HP BASIC, you need to search through the following sections and use the ones that apply to your situation.

Booting BASIC 2.0 and Binaries

You can boot a BASIC 2.0 system and its binaries with one operation. The Loader uses disc drivers in the Boot ROM to load other drivers from an external disc. For example, if you want to use Shared Resource Management (SRM) but do not have a local mass storage device, you can use the Loader to boot BASIC 2.0 and the SRM binary from the SRM. Another example: you want to use a CS80 disc but the AP binary is on the hard disc. You can use the Loader to boot BASIC 2.0 and the AP binary from that disc.

Changing Default Operating System Selection

The Loader can also benefit users on the SRM who want to boot different operating systems at each node without having to override the "default operating system selection" by pressing a key during the boot process (e.g. one user on the SRM boots BASIC 2.0 and the SRM binary; a second user boots Pascal; a third boots HPL and a fourth uses an application. (Note that HPL does not support SRM, so the HPL user would have to have a local mass storage device to store HPL files.) The point is this: the Loader boots the operating system selected by each node.

Boot ROM Versions Required

All Series 300 computers have Boot ROMs that are compatible with the Loader utility. To use the Loader, a Series 200 computer must have a 3.0 or greater Boot ROM (but not revision 3.0L). During power up, the computer should display `''BOOTROM 3.0''`; and if no Boot ROM message is displayed, you have an earlier version of the Boot ROM. If `''BOOTROM 3.0L''` is displayed, the Boot ROM is a subset of the 3.0 Boot ROM and does not support the Loader.

How the Loader Utility Works

The Loader is an operating system which performs its function and then deletes itself, turning control over to the selected operating system. It is loaded just like any other operating system; and in genera, it needs the following things:

- the Loader system.
- an operating system file such as BASIC 2.0.
- any BIN file (needed only for BASIC 2.0).
- and an ASCII file called a configuration file.

All of these files must reside on the same mass storage device and on the same directory for a SRM.

With BASIC 2.0

You can use the Loader to boot BASIC 2.0 and load binaries in one operation. You create a configuration file which contains the name of the operating system file and the names of any binaries you want loaded. The configuration file can also contain a string of keystrokes which provides autostart capability.

When BASIC 2.0 is booted through use of the Loader, any autostart program is ignored.

With BASIC 3.0 and Later Versions

Unlike BASIC 2.0, you cannot load individual BASIC 3.0 (or later) binaries with the Loader Utility. However, this is not a problem, because the binaries may be either stored with the system (using STORE SYSTEM) or loaded programmatically (using an autostart file).

Using the Loader Utility

When you power up your computer, the boot ROM (3.0 or later) looks for the first SYSTM file with a file name beginning with SYSTEM_ or SYS_. This file is then booted unless the operator holds down a key on the keyboard. If the operator does hold down a key, the boot ROM lists all the bootable files it finds and waits for the operator to select one.

The Loader is booted if it is the first file found and the operator does not hold down any key. The Loader then looks for a configuration file. The Loader reads this file and loads the operating system (and binaries for BASIC 2.0). The Loader deletes itself, turning control over to the operating system.

The Configuration File

The configuration file contains the name of the operating system to be loaded (and any binaries for BASIC 2.0). The CONFIGER program creates this ASCII file. CONFIGER is provided on the Loader Utility disc (BASIC 2.0) and on the Utilities Library Disc 2 (BASIC 3.0 and later versions). You can also create the configuration file using the Pascal Editor.

Creating the Configuration File in BASIC

The CONFIGER program prompts for the name of the configuration file and then for each line of the file. For example:

```
CONFIG FILE NAME
CONFIG_LD
?SYSTEM_BA2
?APBIN
?
```

Pressing Return or Enter without entering any character ends the program.

BASIC 2.0

You can include the following four items in the configuration file:

1. the BASIC 2.0 operating system name:
2. a keystroke string, if any;
3. non-scratchable BIN file names, if any;
4. scratchable BIN file names, if any.

The items must appear in the order shown. For example:

```
SYSTEM_BAS
!MSI "MY_DIRECTORY:REMOTE"KXLOAD"AUTOST",1KX
APBIN
SRM2_1
```

The character string in the key string is sent to the keyboard line after the BASIC 2.0 system and specified binaries are loaded. This line begins with an "!" to distinguish it from a file name and may contain up to two CRT lines of characters. A key string is a feature that provides you with autostart capability when using BASIC 2.0 on a Shared Resource Manager (SRM).

An inverse video K followed by X appears on the display when you type CTRL EXECUTE; and the key string is executed immediately after being displayed. The "Outputs to the Keyboard" section in Chapter 9 of the *BASIC Interfacing Techniques* manual contains information about key-code strings.

After the key string, if any, you list as many binaries as desired, including none at all. There are two requirements:

1. If you want to load a scratchable BIN file (e.g. PHYREC), that file name must appear in the configuration file after any non-scratchable BIN file names.
2. If you are using the Loader to load a BIN file that contains drivers for your source device, that BIN file name must be the **last** file name in the configuration file. Note that only non-scratchable binaries contain drivers for mass storage devices. Thus, you cannot load any scratchable binaries.

ROM BASIC 2.0

To load a BASIC 2.0 ROM system instead of a soft system, use "ROM" for the system name. To load a ROM system other than BASIC 2.0, use "X" where X is the first character of the language name. For instance, use "H" to load a ROM HPL system. If you use "B" instead of "ROM" for BASIC 2.0, BASIC is loaded but binaries are not loaded. ("" is the null character and is accessed in BASIC by typing ANY CHAR and 000.)

You Can Need an Extra Line

When you load binaries from an SRM in a BASIC 2.1 ROM configuration file, you must include an extra line. This is the last line in the file and contains a single blank character. If you do not do this, your system will not boot.

You Might Need to Use SCRATCH

When loading binaries from a hard disc in a BASIC 2.1 ROM configuration file, you must include the command SCRATCH A in the key string. If you do not do this, your system will not boot.

The following is an example of a ROM BASIC configuration file.

```
ROM
!SCRATCH AKx
APBIN
b
/
```

BASIC 3.0 and Later Versions

The Loader determines the type of operating system to be loaded. If it is not BASIC 2.0, the Loader uses only the first line in the configuration file. Thus, the configuration file for BASIC 3.0 and later versions contain only one line: the name of the BASIC operating system file.

There are two ways you can include binaries in the system:

1. Boot BASIC 3.0 (or later versions), load each BIN you want, then store the system using the STORE SYSTEM command. Use this stored system as the file the Loader boots.
2. Create an AUTOST file which loads each BIN you want. Unlike BASIC 2.0, the AUTOST file is loaded and run with BASIC 3.0 (or later versions).

Other Operating Systems

The Loader determines the type of operating system to be loaded. If it is not BASIC 2.0, the Loader uses only the first line in the configuration file. Thus, the configuration file for other operating systems contains only one line: the name of the operating system file.

Refer to the operating system documentation for information on system files.

Creating the Configuration File in Pascal

You can create your configuration file in Pascal by accessing the Editor and inserting a line of text for each logical record. When you save the file on your mass storage device, save it as an ASCII file. You do this by appending ".ASC" to the file name when you save the file; e.g., "CONFIG.ASC"

After creating the configuration file, you must change the name of the file to correspond to the name of the Loader system file.

Naming the Configuration File

There is a correspondence between the name of the Loader system file and the name of the configuration file. The file name of the Loader system file on your utility disc is "SYSTEM_LD"; your configuration file name must be "CONFIG_LD". If you change the name of the Loader system file, say to "SYSTEM_XYZ", you must also change the file name of your configuration file to "CONFIG_XYZ".

SRM File Names

When the Loader is on the SRM, the Loader appends the node number to the configuration file name and looks for it. If the Loader does not find it, it looks for the file without the node number appended. For example, if the Loader system file is "SYSTEM_LD" and the computer is at node number 10, the Loader first looks for the configuration file "CONFIG_LD10". If it is not found, the Loader uses the file "CONFIG_LD". This allows each node on the SRM to have its own configuration file.

The Loader system file, the configuration file and the operating system file must all be in the SYSTEMS directory of SRM.

With BASIC 3.0 and later systems, the autostart file can be in either the root directory with a file name of AUTOST, or in the SYSTEMS directory with a file name of AUTOSTxx, where xx is the node number. On powerup, the system first looks for an autostart file in the SYSTEMS directory. If it does not find one, it looks in the root directory. Refer to Chapter 2 of the *BASIC Programming Techniques* manual for more information on autostart files.

Copying Files

Once you have created your configuration file, you must place all the necessary files on the appropriate mass storage device. Those files are:

- The Loader system file (the SYSTM file on your utility disc). This must be the **first** SYSTM file on the mass storage device.
- Your configuration file that is named to correspond to the Loader system file.
- The operating system file named in the configuration file.
- If the operating system is BASIC 2.0, any binaries named in the configuration file.
- If the operating system is BASIC 3.0 or later version, an autostart file, if desired.

Copying Files with BASIC 2.0

If your mass storage device is supported by BASIC 2.0, you can use the COPY command to copy the appropriate files onto the device. If you device is supported by the AP binary, set up your system as you normally would for accessing the desired mass storage device. You can then use the COPY command

If you intend to use the Loader on SRM, you must first load the SRM BIN file onto a node that has a local mass storage device. You can then use the SRM Copy Utility which is on the same disc as the SRM binary to copy the appropriate files into the SRM "SYSTEMS" directory. If you have Revision B of the SRM binary, you can use the COPY command to perform the file transfer.

Copying Files with BASIC 3.0 and Later Versions

Most discs require one of the driver BINs. Load the appropriate BIN and then use the COPY command to load all the files onto the disc.

If you load the files onto an SRM, you must load the SRM and DCOMM BIN files first. COPY the files into the "SYSTEMS" directory.

Copying Files with Pascal

Access the Filer and use the Fcopy command to copy the files. If you intend to use the Loader to load Pascal unattended from SRM, you must copy the operating system file "SYSTEM_P" from your Boot disc into the "SYSTEMS" directory on SRM. Consult the "Special Configurations" chapter of the *Pascal Workstation System* manual for more information on how to set up the Pascal operating system on SRM.

Booting a System

After creating the configuration file and transferring all files to one disc, you are ready to boot a system.

In BASIC 2.0 you must cycle the power to re-boot (turn the power off and then on).

In BASIC 3.0 and later versions you can execute the command SYSBOOT.

In Pascal with the Debugger type Reset and then type **sb** Return (you must use Shift to get lowercase letters). Without the Debugger, type Reset and the booting process is initiated.

The Boot ROM looks for a loadable SYSTM file.

If you do not type any key on the keyboard and the first file the Boot ROM finds is the Loader system, the Loader is booted. The Loader then finds the configuration file and boots your system and binaries (BASIC 2.0).

If you type a key on the keyboard before the Boot ROM finds a loadable SYSTM file, the Boot ROM looks for all loadable SYSTM files and lists them. You then select the system you want booted. This can be the Loader or any other operating system.

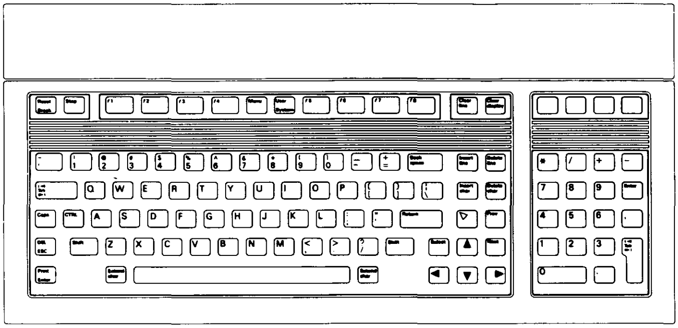
Loader Error Messages

The following error messages indicate some problems that may occur while the Loader system is executing.

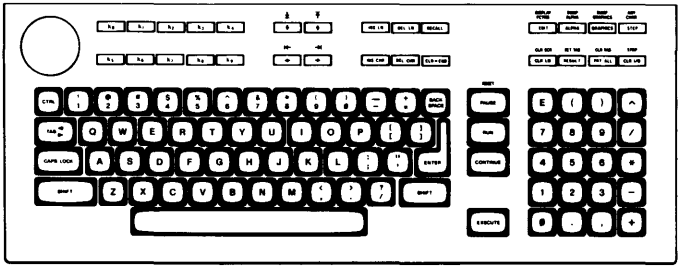
Error Message	Description
NOT FOUND	File name in configuration file not found. Check your configuration file for misspelled file names and make sure all files are on the same mass storage device.
IMPROPER NAME	File name in configuration file contains improper characters.
NAME TOO LONG	File name in configuration file contains more than 10 characters (16 characters for SRM).
DEVICE MISSING	Mass storage device is disconnected during Loader execution.
MASS STORAGE ERROR	Any error related to a mass storage device that occurs while device is being accessed (eg., door opened).
INSUFFICIENT MEMORY	Not enough read/write memory to load operating system or BIN file.
INCOMPATIBLE RELEASE	A BIN file cannot be loaded because it is not compatible with the version of the BASIC operating system which has just been loaded.
INCOMPATIBLE BOOTROM	Loader cannot be used with this version of the Boot ROM. Boot ROM is earlier that 3.0 or is 3.0L (Model 216).
DUPLICATE	Configuration file contains the same BIN file twice. BIN file has already been loaded.
FAILED POWERUP	A BIN file does not initialize properly after it is loaded. BIN file is scratched.

Keyboard Reference

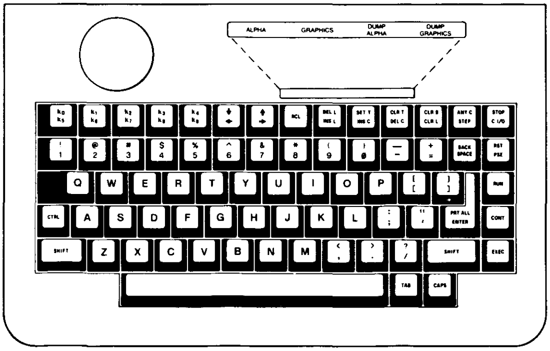
The following three chapters describe the different keyboards available with this BASIC system.



ITF Keyboard (Such as the HP 46021)



HP 98203B and C Keyboards



HP 98203A Keyboards

Every key of each keyboard is described in the corresponding chapter.

ITF Keyboards 19

Note

If you do not have the ITF keyboard, skip ahead to one of the following chapters, which describe the HP 98203B/C and HP 98203A keyboards.

The keys on the ITF keyboard are arranged into the following functional groups:

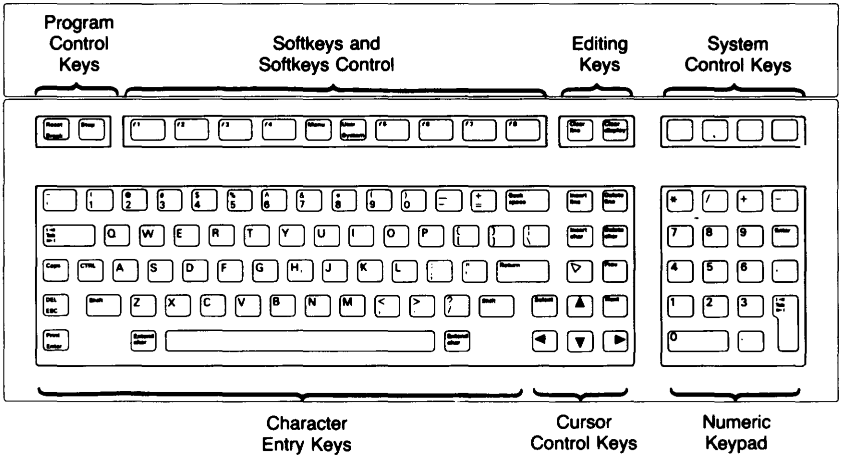


Figure 19-1. ITF Keyboard

This chapter provides a handy reference guide to BASIC's key definitions for the ITF keyboard. Keep in mind that other system programs may define the keys differently. Each key will be demonstrated where possible. One point to clarify: the **cursor** that we refer to in the following paragraphs is the blinking-underline that points to a location on the screen. (If you have a Model 237 computer or an HP 98700 Graphics Display station, the cursor does not blink.)

Note

Before you proceed, type:

SCRATCH [Return]

This clears the computer of any programs that might be left in memory from previous demonstrations.

BASIC ITF Keyboard Overlays

Two keyboard overlays designed for the ITF keyboard were included with your BASIC Language System. Place the overlays on the keyboard as shown below:

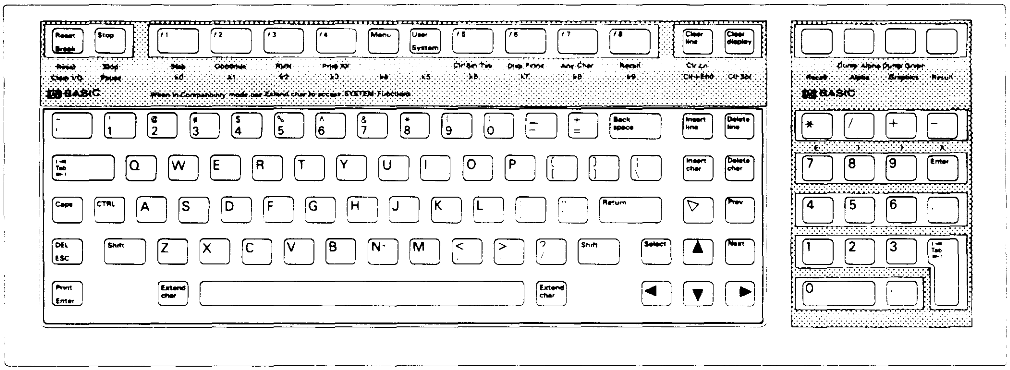
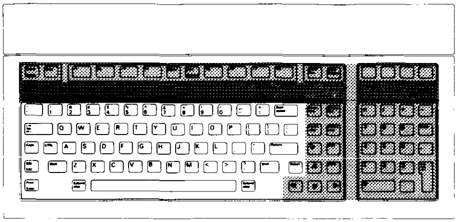


Figure 19-2. BASIC Keyboard Overlays

Character Entry Keys



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

Caps

The Caps key sets the unshifted keyboard to either uppercase (which is the default after BASIC is booted) or lowercase (normal typewriter operation). The computer displays which mode the computer is in when you press the Caps key.

Type a few words, then press Caps and continue typing. Notice the case change. Press Shift-Clear line when finished.

Shift

You can enter standard uppercase and lowercase letters, using the Shift key to access the alternate case.

Type a few words, pressing Shift to change the case of the first letter of each word. Now press Caps and continue typing. Notice that the alternate case accessed by Shift depends on the setting of Caps. Press Shift-Clear line when finished.

Return

The Return key has three functions:

- When a running program prompts you for data, respond by typing the requested data and then pressing Return. This signals the program that you have provided the data and that it can resume execution.
- When typing in lines of a program, the Return key is used to store each line of program code.
- After typing in a command, the Return key causes the command to be executed.

Type EDIT and press Return. Notice the number 10 now displayed on the screen—this is the line number of the first line of a BASIC program. The computer is waiting for you to type in the line. Type:

```
!FIRST LINE
```

and press Return. Notice that the computer accepts the statement as a program line and displays 20 in preparation for the next one. Press Stop when finished.

Enter

Pressing Enter is the same as pressing the Return key.

Print

Pressing Print (Shift-Enter) prints a complete copy of the alpha display on the default printer. The shifted version of the key directly above the / key in the numeric keypad (labeled Dump Alpha on the overlay) performs the same function.

Extend char

When pressed along with another key, this key allows you to generate the rest of the full 256-bit character set from the main typewriter section on Standard and European keyboards (see illustration). On a Katakana keyboard, the "Roman" and "Katakana" keys select the other character sets. To get Katakana characters 161 through 254 on a medium-resolution Series 300 screen, you must load the LEX language extension binary.

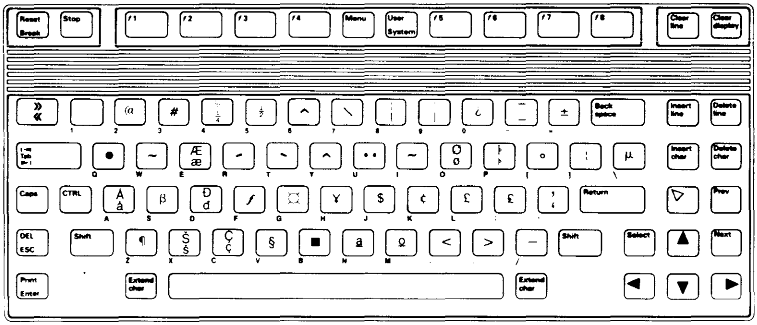


Figure 19-3. Extended Character Set

Tab

The Tab key moves the cursor forward to preset tabs. Pressing Shift-Tab moves the cursor backward to preset tabs.

Before Tab can be used, a tab must be set. Tabs are set and cleared with System menu softkeys. The Tab key is demonstrated along with the **Set Tab/Clr Tab** softkey under "System Softkeys" later in this chapter.

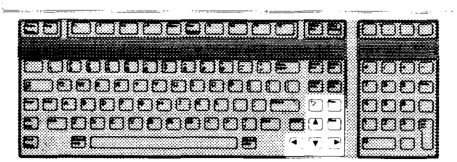
CTRL

The CTRL (control) key works like Shift to access a set of standard control characters, such as line-feed and form-feed. These characters are useful to the programmer for controlling some devices and for communicating with other computers. You probably won't need them when running programs. The available control characters are listed in the *BASIC Language Reference* in the "Useful Tables" appendix.

Select

The Select key beeps but performs no function unless it is program-defined.

Cursor-Control Keys



The cursor-control keys move the display cursor. The [▲] and [▼] keys allow you to scroll lines in the output area up and down. Shifted, the keys allow you to "jump" to the top and bottom of the output area. The [►] and [◄] keys allow you to move horizontally along a line. Shifted, they allow you to "jump" to the left and right limits of a line. The [Back space] key works just like the [◄] key.

The unshifted [➤] key positions the print position at the beginning position on the page. The shifted [➤] key places the print position at the beginning of the first empty line in the display (scrolls up if necessary). In edit mode, pressing this key (shifted or unshifted) causes the computer to beep.

To verify operation of the [➤] key, press [Clear display]. Then type `PRINT "SOMETHING"` and press [Return]; repeat twice. You should now have the following display:

```
SOMETHING
SOMETHING
SOMETHING
```

Press the [➤] key (unshifted).

Type `PRINT "ANY " and press` [Return]. Your display should look like this:

```
ANY THING
SOMETHING
SOMETHING
```

Press [Clear display].

In normal mode, pressing the [Prev] key causes the display to scroll down one page and pressing the [Next] key causes the display to scroll up one page. In edit mode, these keys move the display one-half page.

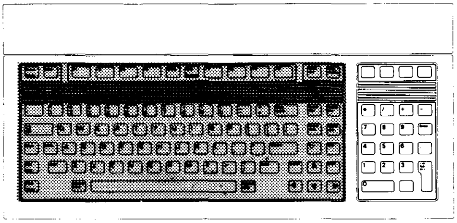
To test the horizontal movement of the cursor, type a few words and press the shifted and unshifted [◄] and [►] keys. Notice that the cursor cannot be moved beyond the characters you have typed. Press [Shift]-[Clear line] when finished.

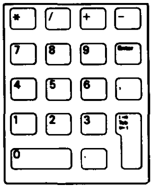
To test the vertical movement of the cursor, type EDIT and press [Return]. Now type the following lines, pressing [Return] after each line (the first line may be there already, so just press [Return] to accept it):

```
10 !FIRST LINE
20 !SECOND LINE
30 !THIRD LINE
40 !FOURTH LINE
```

Try out the shifted and unshifted [▲], [▼], and [▼] keys. Then try the [Prev] and [Next] keys. When you're done, press [Stop] to exit. Then, type SCRATCH [Return] to clear memory.

Numeric Keypad





The numeric keypad provides a convenient way to enter numbers and perform arithmetic operations. Simply type in the arithmetic expression you want to evaluate, then press [Enter]. The result is displayed in the lower-left corner of the screen.

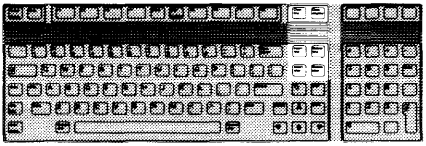
The [Enter] key performs the same function as the [Return] key. The [Tab] key on the numeric keypad functions like the [Tab] key in the character entry area. The shifted versions of the [*], [/], [+], and [-] keys are E, (,), and ^, respectively (see labels on the overlay). The shifted versions are also available in the character entry area.

Type in the following problem using the numeric keypad:

```
(26+14)/4
```

Now press [Enter] to perform the calculation. The answer, 10, is displayed in the lower-left corner of the screen.

Editing Keys



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

Insert line

Pressing Insert line inserts a new line above the cursor's current position (edit mode only).

Type EDIT, then press Return. Type in this line (if it isn't already there):

```
10 !FIRST LINE
```

Now, with the cursor somewhere on line 10, press Insert line. Notice that a new line number (1) is inserted before line 10. Press Stop when finished.

Delete line

Pressing Delete line deletes the line containing the cursor (edit mode only).

Type EDIT, then press Return. Position the cursor to the line:

```
10 !FIRST LINE
```

and press Delete line. The line is removed. To restore it, press the key directly above * (labeled Recall on the overlay) to recall it, then press Return to enter it into the program. Press Stop to exit edit mode.

Insert char

Pressing Insert char sets insert mode, allowing you to insert characters to the left of the cursor. Press the key a second time to cancel insert mode.

Carefully type the following line exactly as shown:

```
THIS IS A TEST .
```

Position the cursor under the period and press Insert char. Now type:

```
OF INSERT MODE
```

and press Insert char again. The line should now look like this:

```
THIS IS A TEST OF INSERT MODE.
```

The new characters were inserted to the left of the period. Press Shift-Clear line when finished.

Delete char

Pressing Delete char deletes the character at the cursor's position.

Type a few words and experiment with Delete char, positioning the cursor at various places on the line. Notice that if you hold the key down, characters are deleted until you release it. Delete all of the characters you typed.

Clear line

Pressing unshifted-Clear line (labeled Clr → End on the overlay) clears from the current cursor position to the end of the line.

Pressing Shift-Clear line (labeled Clr Ln on the overlay) clears the keyboard line and message/results line.

Type in a few words and use the ◄ key to position the cursor in the middle of the line. Press unshifted-Clear line to clear to the end of the line. Press Shift-Clear line to clear the rest of the line.

Clear display

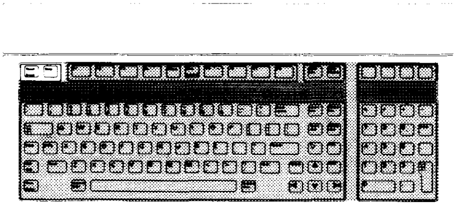
Pressing either the shifted or unshifted version of Clear display clears the entire alpha screen.

Type the following BASIC command:

```
PRINT "PUT THIS MESSAGE IN THE OUTPUT AREA."
```

Now press Return to execute it. Press the key directly above * (labeled Recall on the overlay) to recall the command, and press Return again. Repeat this step several times to fill the screen with messages. Now press Clear display to erase all lines at once.

Program Control Keys



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

Stop

Pressing unshifted-Stop (labeled Pause on the overlay) **pauses** program execution after the current line. Pressing **Continue** (unshifted f2) in the System menu resumes program execution from the point where it was paused.

Pressing Shift-Stop (labeled Stop on the overlay) **stops** program execution after the current line. To restart the program, press **RUN** (unshifted f3) in the System menu.

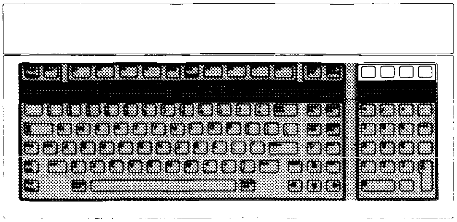
Break

Pressing Break (labeled Clr I/O on the overlay) pauses program execution when the computer is performing or trying to perform an I/O operation. Press Break instead of unshifted-Stop when the computer is hung up on an I/O operation, since unshifted-Stop works only after the computer finishes the current program line. Pressing Break cancels the I/O operation and pauses the program at the current line.

Reset

Pressing Reset (Shift-Break) pauses program execution immediately without erasing the program from memory. The `BASIC Reset` message indicates the computer is ready for your command.

System Control Keys



Four unlabeled keys directly above the numeric keypad 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.

To easily identify the keys in the following description, we'll use this convention:

- Key 1—Above the [*] key (labeled Recall on the overlay).
- Key 2—Above the [/] key (labeled Alpha/Dump Alpha on the overlay).
- Key 3—Above the [+] key (labeled Graphics/Dump Graph on the overlay).
- Key 4—Above the [-] key (labeled RES on the overlay).

Key 1—Recall

Pressing unshifted-Key 1 (Recall) recalls the last line that you entered, executed, or deleted. Several previous lines can be recalled this way. Recall is particularly handy to use when you mistype a line. Instead of retyping the entire line, you can recall it, edit it using the editing keys, and enter or execute it again.

Type:

```
PRINT "1"  [Return]
```

to print the number 1 on the screen. Now press Key 1 to recall the print statement. Edit the statement to print the number 2 by positioning the cursor under the 1 and typing [2] over it. Press [Return] again. Now press Key 1 several times to see all of the statements it remembers. Then press [Clear display] when finished.

[Shift]-Key 1 moves forward through the recall stack.

Pressing [f8] in the System menu performs the same recall function as Key 1.

Key 2—Alpha/ Dump Alpha

Pressing unshifted-Key 2 (Alpha) once turns on the alphanumeric display. Pressing it the second time turns off the graphics display. This key function requires that the GRAPH BIN file be loaded. If you have a Model 237, an HP 98700 Graphics Display station, or Series 300 computer, this key may perform no function.

Pressing Shift-Key 2 (Dump Alpha) prints a complete copy of the alpha display on the default printer. The Dump Alpha function is also executed by Print.

Key 3—Graphics/ Dump Graph

Pressing unshifted-Key 3 (Graphics) once turns on the graphics display. Pressing it the second time turns off the alphanumeric display. If you have a Model 237, an HP 98700 Graphics Display Station, or Series 300 computer, this key may perform no function.

Pressing Shift-Key 3 (Dump Graph) prints a complete copy of the graphics display on the default printer. If you have a Model 237, an HP 98700 Graphics Display Station, or Series 300 computer, the combined alpha and graphics display is printed.

Both key functions require that the GRAPH language extension file be loaded.

Key 4—RES

Pressing Key 4 (RES) either shifted or unshifted returns the result of the last arithmetic expression that was executed.

Press Shift-Clear line, then type:

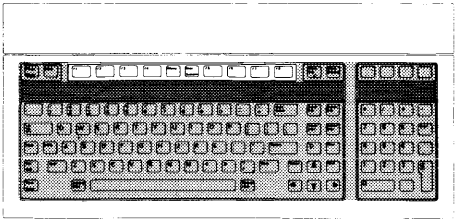
```
23+45  Return
```

The result, 68, is displayed in the lower-left corner of the screen. To add 123 to this value, press Key 4 and type:

```
+123  Return
```

The new result, 191, is now displayed. Press Shift-Clear line when finished.

Softkeys and Softkey Control



There are eight softkeys (labeled [f1] through [f8] and two keys that control the definitions of the softkeys ([Menu] and [System]).

When the BASIC system is booted, the softkeys default to System mode. The System mode menu that appears at the bottom of your display is shown. System softkeys are defined following control key definitions. In addition to the System mode, there are also three User modes: User 1, User 2, and User 3. *BASIC Programming Techniques* describes how to set up User modes.

Softkey Control Keys

[System] Pressing unshifted-[System] causes softkeys to assume System mode. The System menu is displayed, **if** the [Menu] key is toggled to the "on" position.

[User] Pressing [User] ([Shift]-[System]) puts the softkeys in User 1 mode. The User 1 menu is displayed **if** the [Menu] key is toggled to the "on" position.

[Menu] Pressing unshifted-[Menu] toggles the softkey labels—turns them on if they're off and turns them off if they're on.

Pressing [Shift]-[Menu] increments User mode and menu **if** User mode is "on".

User menus are blank unless the KBD language extension binary is loaded.

Let's get familiar with the two control keys.

First we want to get the System mode selected and menu displayed. If the System menu is displayed, continue with the next paragraph. If it is not displayed, press [System]. If it is still not displayed, press [Menu].

With the System menu displayed, press unshifted-[Menu] several times. The system menu display should go on and off. Leave the System menu displayed, and continue.

Now press [Shift]-[User]. The User 1 menu should appear on your display.

Press [Shift]-[Menu] several times. The displayed menus should rotate successively through the three User menus (User 1 → User 2 → User 3 → User 1 → User 2, etc.).

Press unshifted-[Menu] several times and the last User menu goes on and off. Leave the User menu on.

Finish this exercise by pressing unshifted-[System] to get your computer back in System mode.

System Softkeys

The following paragraphs define the eight System softkeys.

Step — **Step** (unshifted-[f1]) allows you to execute one program line at a time. This is particularly useful for debugging (fixing) programs.

Continue — **Continue** (unshifted-[f2]) resumes program execution from the point where it was paused (by an unshifted-[Stop]).

RUN — **RUN** (unshifted-[f3]) starts a program running from the beginning.

Print All

The **Print All** key (unshifted-f4) turns the printall mode on and off, allowing keyboard operations and displayed error messages to be copied to a printall device. Press **Print All** once to set printall "on" and again to set printall "off". An asterisk (*) appears next to `All` to indicate that printall is "on".

The display's output area is the default printall device at powerup. *BASIC Programming Techniques* explains how to select other printall devices.

Press **Print All** to turn on printall mode. Now type in the following command:

```
PRINT "THIS IS A KEYBOARD OPERATION"  Return
```

Both the PRINT command and the message itself are displayed on the screen, which is the default printall device. Now type:

```
THIS WILL CAUSE AN ERROR  Return
```

Because this is not an executable BASIC statement, an error message is displayed, both at the bottom of the screen and in the printall area at the top. This way, a log is produced of all commands typed and executed at the keyboard, along with any error messages. Press Clear display to clear the display, and press **Print All** to turn off printall mode.

Set Tab/Clr Tab

Set Tab (unshifted-f5) sets a tab at the cursor's current position. Tabs remain in effect until cleared by either **Clr Tab** or the SCRATCH A statement (explained in *BASIC Programming Technique*).

Clr Tab (Shift-f5) clears a tab previously set at the cursor's position.

Press the space bar to move the cursor forward a few spaces and press **Set Tab**. Move the cursor back several spaces using ◄, then press Tab. Move the cursor forward several more spaces with the space bar, then press Shift-Tab. To clear the tab, move the cursor to the unwanted tab position and press **Clr Tab**. Press Shift-Clear line when finished.

Display Fctns

Display Fctns (unshifted-f6) sets the display-functions mode, allowing you to see special control characters (e.g., form-feed, carriage return) on the screen. Pressing this key a second time cancels the display-functions mode. An asterisk (*) appears next to `Fctns` to indicate that display-functions mode is "on".

Type the following line:

```
PRINT "DISPLAY-FUNCTIONS MODE OFF"  Return
```

Notice the display at the top of the screen. Now press **Recall** (unshifted-f8) to recall the line, and edit it to read:

```
PRINT "DISPLAY-FUNCTIONS MODE ON"
```

Press **Display Fctns**, and then press Return. Notice that the carriage return (CR) and line-feed (LF) control characters are now displayed. Press **Display Fctns** again to exit display-functions mode. Press Clear display when finished.

Any char

Any char (unshifted-f7) is used to find any ASCII character. First press **Any char**. The following message appears above the menu:

```
Enter 3 digits, 000 to 255
```

Enter a three-digit number from 000 through 255 representing the decimal equivalent of an ASCII character. The computer automatically displays the character on the screen. For a list of characters and their equivalent decimal values, see the US ASCII Character Codes table in the "Useful Tables" appendix of the *BASIC Language Reference.*

Press **Any char**, then type 65 which is the decimal equivalent of "A". The display line now displays "A". Press Shift-Clear line to erase it.

Recall

The **Recall** softkey (unshifted-[f8]) acts just like System Control Key 1 (described earlier). **Recall** recalls the last line that you entered, executed, or deleted. Several previous lines can be recalled this way. **Recall** is particularly handy to use when you mistype a line. Instead of retyping the entire line, you can recall it, edit it using the editing keys, and enter or execute it again.

Type:

```
PRINT "1"  [Return]
```

to print the number 1 on the screen. Now press **Recall** to recall the PRINT statement. Edit the statement to print the number 2 by positioning the cursor under the 1 and typing [2] over it. Press [Return] again. Now press **Recall** several times to see all of the statements it remembers. Note that **Recall** goes backward through the queue.

Pressing [Shift]-[f8] allows you to cycle forward through the queue until the last line entered, executed, or deleted is displayed. In the previous exercise you pressed unshifted-[f8] several times, cycling backward through the queue. Now press [Shift]-[f8] several times to cycle forward through the queue until the last line is displayed.

HP 98203B/C Keyboards 20

Note

If you have the ITF keyboard, refer to the preceding chapter. If you have the HP 98203A keyboard, skip to the following chapter.

HP 98203B/C keys are arranged into the following functional groups:

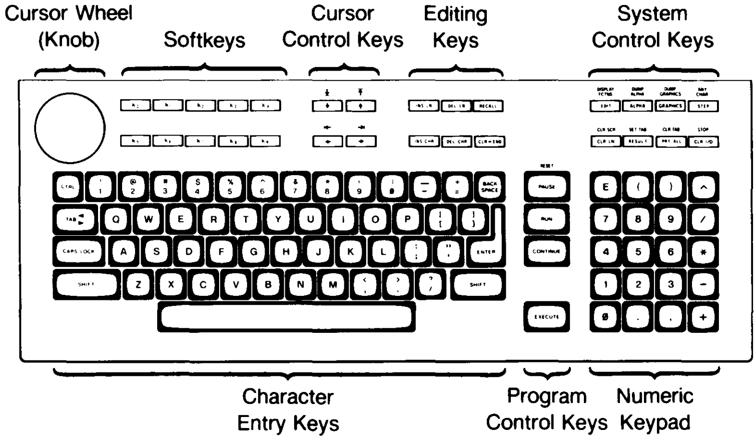


Figure 20-1. HP 98203B/C Keyboard

This chapter provides a handy reference guide to BASIC's key definitions for the HP 98203B/C keyboard. Keep in mind that other system programs may define the keys differently. Each key will be demonstrated where possible. One point to clarify: the **cursor** that we refer to in the following paragraphs is the underline that points to a location on the screen.

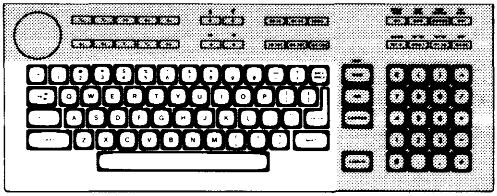
Note

Before you proceed, type:

`SCRATCH` EXECUTE

This clears the computer of any programs that might be left in memory from previous demonstrations.

Character Entry Keys



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

CAPS LOCK The CAPS LOCK key sets the unshifted keyboard to either uppercase (which is the default after BASIC is booted) or lowercase (normal typewriter operation). The computer displays which mode the computer is in when you press the CAPS LOCK key.

Type a few words, then press CAPS LOCK and continue typing. Notice the case change. Press CLR LN when finished.

SHIFT You can enter standard uppercase and lowercase letters, using the SHIFT key to access the alternate case.

Type a few words, pressing SHIFT to change the case of the first letter of each word. Now press CAPS LOCK and continue typing. Notice that the alternate case accessed by SHIFT depends on the setting of CAPS LOCK. Press CLR LN when finished.

ENTER

The ENTER key has several functions:

- When a running program prompts you for data, respond by typing the requested data and then pressing ENTER. This signals the program that you have provided the data and that it can resume execution. The EXECUTE key can also be used for this function.
- When typing in lines of a program, the ENTER key is used to store each line of program code. The EXECUTE key can also be used for this function.
- Like the EXECUTE key, the ENTER key can be used to execute commands and calculations.

Type `EDIT` and press ENTER. Notice the number `10` now displayed on the screen—this is the line number of the first line of a BASIC program. The computer is waiting for you to type in the line. Type:

```
!FIRST LINE
```

and press ENTER. Notice that the computer accepts the statement as a program line and displays `20` in preparation for the next one. Press PAUSE when finished.

TAB

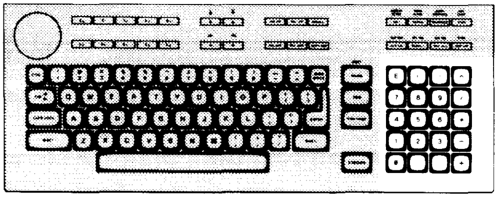
The TAB key moves the cursor forward to preset tabs. Pressing SHIFT-TAB moves the cursor backward to preset tabs.

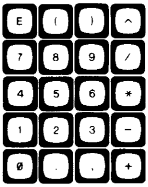
Before TAB can be used, a tab must be set. Press the space bar to move the cursor forward a few spaces and press SET TAB (SHIFT-RESULT). Move the cursor back several spaces using ←, then press TAB. Move the cursor forward several more spaces with the space bar, then press SHIFT-TAB. To clear the tab, move the cursor to the unwanted tab position and press CLR TAB (SHIFT-PRT ALL). Press CLR LN when finished.

CTRL

The CTRL (control) key works like SHIFT to access a set of standard control characters, such as line-feed and form-feed. These characters are useful to the programmer for controlling some devices and for communicating with other computers. You probably won't need them when running programs. The available control characters are listed in the "Useful Tables" appendix of *BASIC Language Reference.*

Numeric Keypad





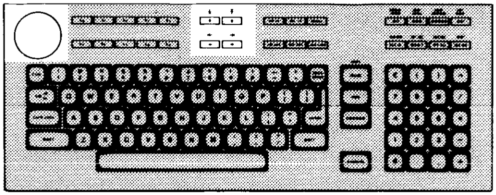
The numeric keypad provides a convenient way to enter numbers and perform arithmetic operations. Simply type in the arithmetic expression you want to evaluate, then press EXECUTE. The result is displayed in the lower-left corner of the screen.

Type in the following problem using the numeric keypad:

```
(26+14)/4
```

Now press EXECUTE to perform the calculation. The answer, `10`, is displayed in the lower-left corner of the screen.

Cursor-Control Keys



The cursor-control keys move the display cursor. The [↑] and [↓] keys allow you to scroll lines in the output area up and down. Shifted, the keys allow you to "jump" to the top and bottom of the output area. The [→] and [←] keys allow you to move horizontally along a line. Shifted, they allow you to "jump" to the left and right limits of a line. The [BACK SPACE] key works just like the [←] key.

The cursor control wheel (also called the knob) allows you to rapidly scroll the print area (with [SHIFT] pressed) or move the cursor left and right (unshifted).

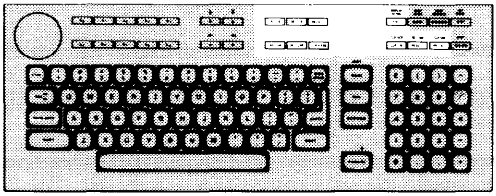
To test the horizontal movement of the cursor, type a few words and press the [←] and [→] keys. Notice that the cursor cannot be moved beyond the characters you have typed. Now rotate the wheel to move the cursor. Press [CLR LN] when finished.

To test vertical scrolling, type **EDIT** and press [EXECUTE]. Now type the following lines, pressing [ENTER] after each line (the first line may be there already, so just press [ENTER] to accept it):

```
10 !FIRST LINE
20 !SECOND LINE
30 !THIRD LINE
40 !FOURTH LINE
```

Press the [SHIFT] key and rotate the wheel to scroll the text up and down. Also try out the [↑] and [↓] keys. When you're done, press [PAUSE] to exit.

Editing Keys



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

EDIT

The EDIT key is a typing convenience; pressing EDIT followed by EXECUTE puts the computer in program edit mode. Edit mode allows you to enter and edit program lines.

Press EDIT, then EXECUTE to enter edit mode. The number 10 appears on the screen. This is a line number for a BASIC program; the computer is waiting for you to type in a line of code. If there is a program already in memory, the computer displays it on the screen. Press PAUSE to exit edit mode.

RECALL

The RECALL key recalls the last line that you entered, executed, or deleted. Several previous lines can be recalled this way. RECALL is particularly handy when you mistype a line. Instead of retyping the entire line, you can recall it, edit it using the editing keys, and enter or execute it again.

Type:

```
PRINT "1"  EXECUTE
```

to print the number 1 on the screen. Now press RECALL to recall the PRINT statement. Edit the statement to print the number 2 by positioning the cursor under the 1 and typing 2 over it. Press EXECUTE again. Now press RECALL several times to see all of the statements it remembers. Then press CLR SCR when finished.

SHIFT-RECALL moves forward through the recall stack.

INS LN

INS LN inserts a new line above the cursor's current position (edit mode only).

Press EDIT, then EXECUTE. Type in this line (if it isn't already there):

```
10 !FIRST LINE
```

Now, with the cursor somewhere on line 10, press INS LN. Notice that a new line number (1) is inserted before line 10. Press PAUSE when finished.

DEL LN

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

Press EDIT, then EXECUTE. Position the cursor to the line:

```
10 !FIRST LINE
```

and press DEL LN. The line is removed. To restore it, press RECALL to retrieve it, then ENTER to enter it into the program. Press PAUSE to exit edit mode.

INS CHR

INS CHR sets insert mode, allowing you to insert characters to the left of the cursor. Press the key a second time to cancel insert mode.

Carefully type the following line exactly as shown:

```
THIS IS A TEST .
```

Position the cursor under the period and press INS CHR. Now type:

```
OF INSERT MODE
```

and press INS CHR again. The line should now look like this:

```
THIS IS A TEST OF INSERT MODE.
```

The new characters were inserted to the left of the period. Press CLR LN when finished.

DEL CHR

DEL CHR deletes the character at the cursor's position.

Type a few words and experiment with DEL CHR, positioning the cursor at various places on the line. Notice that if you hold the key down, characters are deleted until you release it. Delete all of the characters you typed.

SET TAB

SET TAB (SHIFT-RESULT) sets a tab at the cursor's current position. Tabs remain in effect until cleared by either CLR TAB or the SCRATCH A statement. The SCRATCH commands are explained in *BASIC Programming Techniques.* To demonstrate SET TAB, see TAB.

CLR TAB

CLR TAB (SHIFT-PRT ALL) clears a tab previously set at the cursor's position. To demonstrate CLR TAB, see TAB.

CLR LN

CLR LN clears the keyboard line and message/results line.

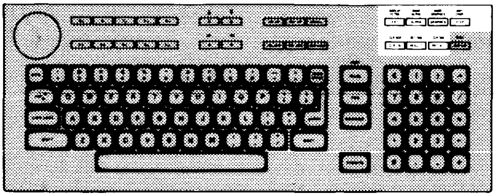
Type a few words and press CLR LN to clear them.

CLR→END

CLR→END clears from the current cursor position to the end of the line.

Type in a few words and use the cursor control wheel or ← to position the cursor in the middle of the line. Press CLR→END to clear to the end of the line. Press CLR LN to clear the rest of the line.

System Control Keys



These keys 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

EDIT types the EDIT command on the keyboard line. See the Editing Keys section for more information.

DISPLAY FCTNS

DISPLAY FCTNS (SHIFT-EDIT) sets the display-functions mode, allowing you to see special control characters (e.g., form-feed, carriage return) on the screen. Pressing this key a second time cancels the display-functions mode.

Type the following line:

```
PRINT "DISPLAY-FUNCTIONS MODE OFF"  EXECUTE
```

Notice the display at the top of the screen. Now press RECALL to recall the line, and edit it to read:

```
PRINT "DISPLAY-FUNCTIONS MODE ON"
```

Press the DISPLAY FCTNS key, and then press EXECUTE. Notice that the carriage return (CR) and line-feed (LF) control characters are now displayed. Press DISPLAY FCTNS again to exit display-functions mode. Press CLR SCR when finished.

ALPHA
GRAPHICS

ALPHA and GRAPHICS allow you to turn the alpha and graphics display modes on and off. The GRAPH binary must be loaded for these keys to function.

DUMP ALPHA

The DUMP ALPHA (SHIFT-ALPHA) key prints a complete copy of the alpha display on the default printer.

DUMP GRAPHICS

The DUMP GRAPHICS (SHIFT-GRAPHICS) key prints a complete copy of the graphics display on the default printer. The GRAPH binary must be loaded for this key to function.

STEP

STEP allows the programmer to step through a program, one line at a time. Using the STEP key to debug programs is covered in *BASIC Programming Techniques.*

ANY CHAR

ANY CHAR (SHIFT-STEP) is used to find any ASCII character. First press ANY CHAR. Then enter a three-digit number from 000 through 255 representing the decimal equivalent of an ASCII character. The computer automatically displays the character on the screen. For a list of characters and their equivalent decimal values, see the US ASCII Character Codes table in the "Useful Tables" appendix of the *BASIC Language Reference.*

Press ANY CHAR, then type 65 which is the decimal equivalent of "A". The "A" is now displayed in the keyboard line. Press CLR LN to erase it.

CLR SCR

CLR SCR (SHIFT-CLR LN) clears the entire alpha screen.

Type the following BASIC command:

```
PRINT "PUT THIS MESSAGE IN THE OUTPUT AREA."
```

Now press EXECUTE to execute it. Press RECALL to recall the command and press EXECUTE again. Repeat this step several times to fill the screen with messages. Now press CLR SCR to erase all lines at once.

RESULT

RESULT returns the result of the last arithmetic expression that was executed.

Press CLR LN, then type:

```
23+45  EXECUTE
```

The result, 68, is displayed in the lower-left corner of the screen. To add 123 to this value, type:

```
RESULT +123  EXECUTE
```

The new result, 191, is now displayed. Press CLR LN when finished.

PRT ALL

The PRT ALL key turns the printall mode on and off, allowing keyboard operations and displayed error messages to be copied to a printall device. Press PRT ALL once to set printall "on" and again to set printall "off". The printall mode is displayed in the lower-left corner of the screen.

The screen's output area is the default printall device. Selecting an external printall device is explained in *BASIC Programming Techniques.*

Press PRT ALL to turn on printall mode. Now type in the following command:

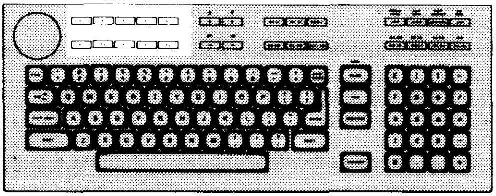
```
PRINT "THIS IS A KEYBOARD OPERATION"  EXECUTE
```

Both the PRINT command and the message itself are displayed on the screen, which is the default printall device. Now type:

```
THIS WILL CAUSE AN ERROR  EXECUTE
```

Because this is not an executable BASIC statement, an error message is displayed, both at the bottom of the screen and in the printall area at the top. This way, a log is produced of all commands typed and executed at the keyboard, along with any error messages. Press CLR SCR to clear the screen, and press PRT ALL to turn off printall mode.

Softkeys

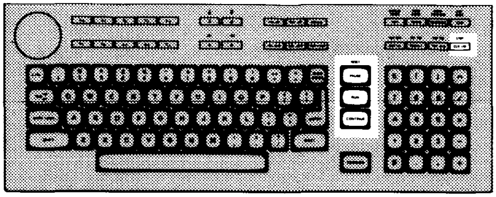


The ten keys labeled [k0] through [k9] are defined under program control. The program may also display a label for each defined key. Pressing a defined key tells the computer to interrupt whatever it's doing and start running another part of the program.

We call these keys "softkeys" because the program or "software" defines and labels them. Another ten softkeys (without the displayed labels) can be defined at the same time and accessed with the [SHIFT] key. These shifted softkeys are often referred to as **k10** through **k19**.

With KBD language extension binary loaded, softkeys are defined as typing aids.

Program Control Keys



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

RUN

RUN starts a program running from the beginning.

PAUSE

PAUSE pauses program execution after the current line. It is also used to exit the Editor.

CONTINUE

CONTINUE resumes program execution from the point where it was paused. It is also used like ENTER or EXECUTE to respond to a program prompt.

STOP

STOP (SHIFT-CLR I/O) stops program execution after the current line. Unlike PAUSE, you cannot resume execution of a program stopped with STOP by pressing CONTINUE. To restart the program, use the RUN key.

RESET

RESET (SHIFT-PAUSE) stops program execution immediately without erasing the program from memory. The `BASIC Reset` message indicates the computer is ready for your command.

CLR I/O

CLR I/O pauses program execution when the computer is performing or trying to perform an I/O operation. Press CLR I/O instead of PAUSE when the computer is hung up on an I/O operation, since PAUSE works only after the computer finishes the current program line. Pressing CLR I/O cancels the I/O operation and pauses the program at the current line.

HP 98203A Keyboards 21

Note

If you have an ITF or an HP 98203B/C keyboard, ignore this chapter and refer to one of the two preceding chapters.

The keys on the HP 98203A keyboard are arranged into the following functional groups:

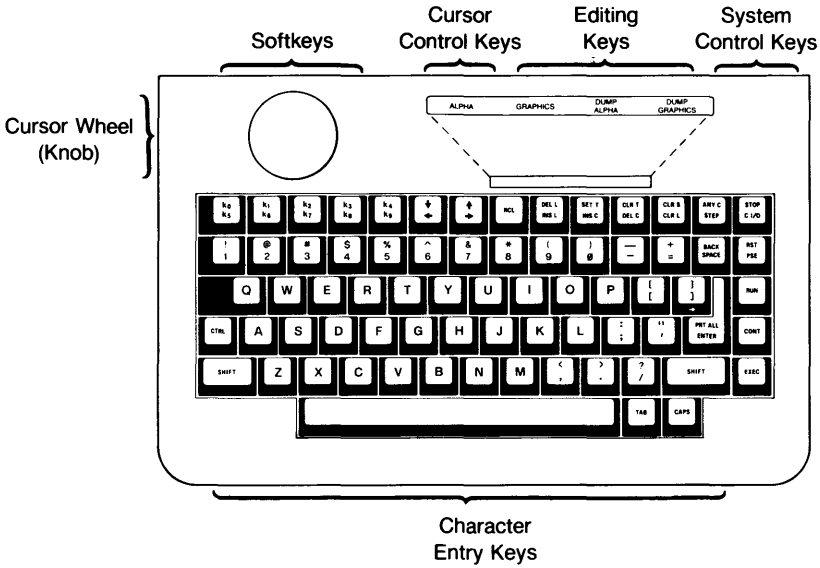


Figure 21-1. HP 98203A Keyboard

This chapter provides a handy reference guide to BASIC's key definitions for the HP 98203A keyboard. Keep in mind that other system programs may define the keys differently. Each key will be demonstrated where possible. One point to clarify: the **cursor** that we refer to in the following paragraphs is the blinking-underline that points to a location on the screen.

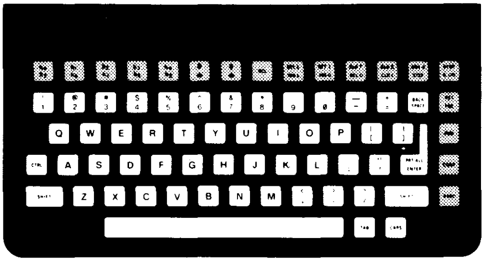
Note

Before you proceed, type:

```
SCRATCH [EXEC]
```

This clears the computer of any programs that might be left in memory from previous demonstrations.

Character Entry Keys



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

[CAPS]

The [CAPS] key sets the unshifted keyboard to either uppercase (which is the default after BASIC is booted) or lowercase (normal typewriter operation). The computer displays which mode the computer is in when you press the [CAPS] key.

Type a few words, then press [CAPS] and continue typing. Notice the case change. Press [CLR L] when finished.

[SHIFT]

You can enter standard uppercase and lowercase letters, using the [SHIFT] key to access the alternate case.

Type a few words, pressing [SHIFT] to change the case of the first letter of each word. Now press [CAPS] and continue typing. Notice that the alternate case accessed by [SHIFT] depends on the setting of [CAPS]. Press [CLR L] when finished.

[ENTER]

The [ENTER] key has several functions:

- When a running program prompts you for data, you respond by typing the requested data and then pressing [ENTER]. This signals the program that you have provided the data and that it can resume execution. The [EXEC] key can also be used for this function.
- When typing in lines of a program, the [ENTER] key is used to store each line of program code. The [EXEC] key can also be used for this function.
- Like the [EXEC] key, the [ENTER] key can be used to execute commands and calculations.

Type `EDIT` and press [ENTER]. Notice the number `10` now displayed on the screen—this is the line number of the first line of a BASIC program. The computer is waiting for you to type in the line. Type:

```
!FIRST LINE
```

and press [ENTER]. Notice that the computer accepts the statement as a program line and displays `20` in preparation for the next one. Press [PSE] to exit.

[TAB]

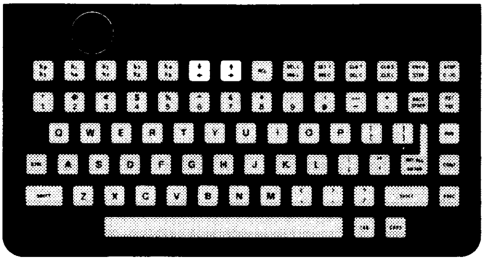
The [TAB] key moves the cursor forward to preset tabs. Pressing [SHIFT]-[TAB] moves the cursor back to preset tabs.

Before [TAB] can be used, a tab must be set. Press the space bar to move the cursor forward a few spaces and press [SET T]. Move the cursor back several spaces using [←], then press [TAB]. Move the cursor forward several more spaces with the space bar, then press [SHIFT]-[TAB]. To clear the tab, move the cursor to the unwanted tab position and press [CLR T].

[CTRL]

The [CTRL] (control) key works like [SHIFT] to access a set of standard control characters, such as line-feed and form-feed. These characters are useful to the programmer for controlling some devices and for communicating with other computers. You probably won't need them when running programs. The available control characters are listed in the "Useful Tables" appendix of *BASIC Language Reference.*

Cursor-Control Keys



The cursor-control keys move the display cursor. The ↑ and ↓ keys allow you to scroll lines in the output area up and down. The → and ← keys allow you to move horizontally along a line. The BACK SPACE key works just like the ← key.

The cursor control wheel (also called the knob) allows you to rapidly scroll the output area up and down or move the cursor left and right, depending on the SHIFT key. With the SHIFT key pressed, the knob scrolls the output area up and down. Without the SHIFT key pressed, the knob moves the cursor left and right.

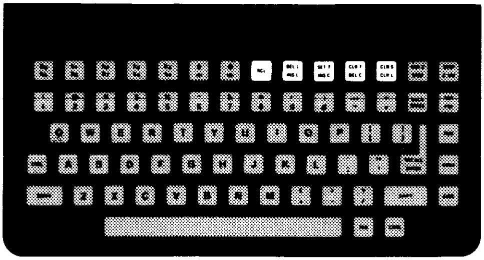
To test the horizontal movement of the cursor, type a few words and press the ← and → keys. Notice that the cursor cannot be moved beyond the characters you have typed. Now rotate the wheel to move the cursor. Press CLR L when finished.

To test vertical scrolling, type `EDIT` and press EXEC. Now type the following lines, pressing ENTER after each line (the first line may be there already, so just press ENTER to accept it):

```
10 !FIRST LINE
20 !SECOND LINE
30 !THIRD LINE
40 !FOURTH LINE
```

Now, press SHIFT and rotate the knob to scroll the text up and down. Also try out the ↑ and ↓ keys. When you're done, press PSE to exit.

Editing Keys



The editing keys put easy character editing and line editing at your fingertips. Some of these keys only work when you are in edit mode, which is entered by typing:

```
EDIT  [EXEC]
```

Edit mode is described in detail in *BASIC Programming Techniques.* To exit edit mode, press [PSE].

[RCL]

The [RCL] key recalls the last line that you entered, executed, or deleted. Several previous lines can be recalled this way. [RCL] is particularly handy to use when you mistype a line. Instead of retyping the entire line, you can recall it, edit it using the editing keys, and enter or execute it again.

Type:

```
PRINT "1"  [EXEC]
```

to print the number 1 on the screen. Now press [RCL] to recall the PRINT statement. Edit the statement to print the number 2 by positioning the cursor under the 1 and typing [2] over it. Press [EXEC] again. Now press [RCL] several times to see all of the statements it remembers from the last entered to the earliest entered. Then press [SHIFT]-[RCL] several times to review the statements from the earliest to the last. Press [CLR S] to exit.

INS L

INS L inserts a new line above the cursor's current position (edit mode only).

Type **EDIT** EXEC. Then, type in this line (if it isn't already there):

```
10 !FIRST LINE
```

Now, with the cursor somewhere on line 10, press INS L. Notice that a new line number (1) is inserted before line 10. Press PSE to exit.

DEL L

DEL L (SHIFT-INS L) deletes the line containing the cursor (edit mode only).

Type **EDIT** EXEC. Position the cursor to the line:

```
10 !FIRST LINE
```

and press DEL L. The line is removed. To restore it, press RCL to retrieve it, then ENTER to enter it into the program. Press PSE to exit edit mode.

INS C

INS C sets insert mode, allowing you to insert characters to the left of the cursor. Press the key a second time to cancel insert mode.

Carefully type the following line exactly as shown:

```
THIS IS A TEST .
```

Position the cursor under the period and press INS C. Now type:

```
OF INSERT MODE
```

and press INS C again. The line should now look like this:

```
THIS IS A TEST OF INSERT MODE.
```

The new characters were inserted to the left of the period. Press CLR L when finished.

SET T

SET T (SHIFT-INS C) sets a tab at the cursor's current position. Tabs are in effect for the keyboard line until cleared by either CLR T or the SCRATCH A statement. The SCRATCH commands are explained in *BASIC Programming Techniques.* To demonstrate SET T, see TAB.

CLR T

CLR T (SHIFT-DEL C) clears a tab previously set at the cursor's position. To demonstrate CLR T, see TAB.

DEL C

DEL C deletes the character at the cursor's position. Type a few words and experiment with DEL C, positioning the cursor at various places on the line. Notice that if you hold the key down, characters are deleted until you release it. Delete all of the characters you typed.

CLR L

CLR L clears the keyboard line and message/results line. Type a few words and press CLR L to clear them.

CLR S

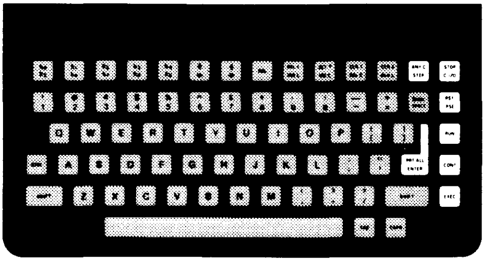
CLR S (SHIFT-CLR L) clears the entire alpha screen.

Type the following BASIC command:

```
PRINT "PUT THIS MESSAGE IN THE OUTPUT AREA."
```

Now press EXEC to execute it. Press RCL to recall the command and press EXEC again. Repeat this step several times to fill the screen with messages. Now press CLR S to erase all lines at once.

System Control Keys



The keys on the right-hand side of the keyboard 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.

STEP — STEP allows you to step through a program, one line at a time. Using the STEP key to debug programs is covered in *BASIC Programming Techniques.*

ANY C — ANY C (SHIFT-STEP) is used to find any ASCII character. First press ANY C. Then enter a three-digit number from 000 through 255 representing the decimal equivalent of an ASCII character. The computer automatically displays the character on the screen. For a list of characters and their equivalent decimal values, see the US ASCII Character Codes table in the "Useful Tables" appendix of the *BASIC Language Reference.*

Press ANY C, then type 65 which is the decimal equivalent of "A". The "A" is now displayed in the keyboard line. Press CLR L to erase it.

RST — RST (SHIFT-PSE) stops or resets program execution immediately without erasing the program from memory. The **BASIC Reset** message indicates the computer is ready for your command.

PRT ALL

PRT ALL (SHIFT-ENTER) key turns the printall mode on and off, allowing keyboard operations and displayed error messages to be copied to a printall device. Press PRT ALL once to set printall "on" and again to set printall "off". The printall mode is displayed in the lower-left corner of the screen.

The screen's output area is the default printall device. Selecting an external printall device is explained in *BASIC Programming Techniques.*

Press PRT ALL to turn on printall mode. Now type in the following command:

```
PRINT "THIS IS A KEYBOARD OPERATION"  EXEC
```

Both the PRINT command and the message itself are displayed on the screen, which is the default printall device. Now type:

```
THIS WILL CAUSE AN ERROR  EXEC
```

Because this is not an executable BASIC statement, an error message is displayed, both at the bottom of the screen and in the printall area at the top. This way, a log is produced of all commands typed and executed at the keyboard, along with any error messages. Press CLR S to clear the screen, and press PRT ALL to turn off printall mode.

RUN

RUN starts a program running from the beginning.

PSE

PSE pauses program execution after the current line. When in edit mode, PSE causes the computer to exit edit mode. Some BASIC keyboard commands cannot be executed while a program is running. In this situation, you can press PSE to suspend program execution, type and execute your keyboard command, then resume the program with the CONT key (described next). (There are some keyboard commands which will not allow a program to be resumed.)

CONT

CONT resumes program execution from the point where it was paused. It is also used like ENTER or EXEC to respond to a program prompt.

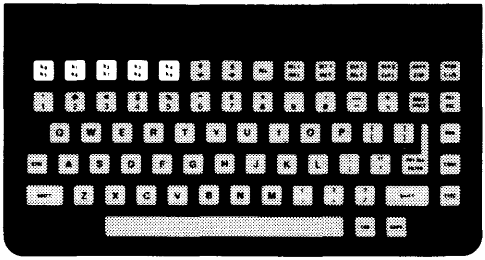
C I/O

C I/O pauses program execution when the computer is doing an I/O operation. Press C I/O instead of PSE when the computer is hung up on an I/O operation, since PSE works only after the computer finishes the current program line. Pressing C I/O cancels the I/O operation and pauses the program at the current line.

STOP

STOP (SHIFT-C I/O) stops program execution after the current line. Unlike PSE, you cannot resume execution of a program stopped with STOP by pressing CONT. To restart the program from the beginning, use the RUN key.

Softkeys



The ten keys labeled k0 through k4 (using the SHIFT key) and k5 through k9 are defined under program control. The program may also display a label for each defined key. Pressing a defined key tells the computer to interrupt whatever it's doing and start running the designated part of the program.

We call these keys "softkeys" because the program or "software" defines and labels them.

With the KDB language extension binary loaded, softkeys can be used as typing aids.

Index

HEWLETT
PACKARD

HP Part Number
98613-90042

Microfiche No. 98613-99042
Printed in U.S.A. 8/87

98613-90639
For Internal Use Only