1926L HEXBUS Disk Drive/Controlle Part 1 0!=Bullet μ =Question Mark Alpha=Number Sign

Federal Communications Commission Requirements Concerning Radio Frequency Interference

The Texas Instruments Computer and peripherals generate and use radio frequency (RF) energy. If not installed and used properly (as outlined in the instructions provided by Texas Instruments), this equipment may cause interference to radio and television reception.

This equipment has been type-tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules. These rules are designed to provide reasonable protection against radio and television interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause interference to radio or television reception (which you can determine by turning the equipment off and on), try to correct the interference by one or more of the following measures:

!o! Reorient the receiving antenna (that is, the antenna for the radio or television that is "receiving" the interference).

o! Change the position of the computer with respect to the radio or television equipment that is receiving interference.

!o! Plug the computer into a different wall outlet so that the computer and the equipment receiving interference are on different branch circuits.

If these measures do not eliminate the interference, please consult your dealer or an experienced radio/television technician for additional suggestions. Also, the Federal Communications Commission has prepared a helpful booklet, "How to Identify and Resolve Radio-TV Interference Problems." This booklet is available from

The US Government Printing Office Wasnington, D. C. 20402

Please specify Stock Number QQ4-QQ0-QQ345-4 when ordering copies.

WARNING: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

This book was developed by: David Thomas Robert E Whitsitt, II

With contributions by:
Nancy Bain Barnett
Craig H. Benson
Wendell W. Broom
Greg Evans
Kenneth S. Gregg
Robert G. Harr
Kenneth E. Heichelheim
Jeffrey G. Jones
Ben Korte
Jerry L. Pyles
Gary L. Rouse
Jef Winsor

IMPORTANT NOTICE REGARDING PROGRAMS AND BOOK MATERIALS

TEXAS INSTRUMENTS MAKES NO WARRANTY, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE, REGARDING THESE PROGRAMS OR BOOK MATERIALS OR ANY PROGRAMS DERIVED THEREFROM AND MAKES SUCH MATERIALS AVAILABLE SOLELY ON AN "AS IS" BASIS.

IN NO EVENT SHALL TEXAS INSTRUMENTS BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING OUT OF THE PURCHASE OR USE OF THESE BOOK MATERIALS OR PROGRAMS AND THE SOLE AND EXCLUSIVE LIABILITY OF TEXAS INSTRUMENTS, REGARDLESS OF THE FORM OF ACTION, SHALL NOT EXCEED THE PURCHASE PRICE OF THIS PERIPHERAL. MOREOVER, TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR ANY CLAIM OF ANY KIND WHATSOEVER AGAINST THE USER OF THESE PROGRAMS OR BOOK MATERIALS BY ANY OTHER PARTY.

THESE BOOK MATERIALS AND THE PROGRAM CODE CONTAINED IN THE DISK MANAGER 3 CARTRIDGE AND THE DISK DRIVE/CONTROLLER ARE COPYRIGHTED BY TEXAS INSTRUMENTS.

Copyright (C) 1983 by Texas Instruments Incorporated

1926L HEXBUS Disk Drive/Controller Part 1 to:=Bullet μ =Question Mark Alpha=Number Sign

Table of Contents

1. Introduction

Features of the Disk Drive/Controller Features of the Disk Manager 3 Cartridge Compact Computer Disk Utilities

2. Set-Up Instructions

Package Contents

Setting Up the Disk Drive/Controller

Connecting the Drive/Controller to a 99/4A Home Computer Connecting the Drive/Controller to a Compact Computer 41

Inserting and Removing a Diskette

3. Diskette Information

Write-Protecting Diskettes

Labeling and Handling Precautions

4. A Quick Demonstration of the Disk System

For Home Computer Owners

Initializing a Diskette

Saving a Program to Diskette

Storing and Retrieving Data

Displaying a Diskette's Catalog

For Compact Computer Owners

Initializing a Diskette

Saving a Program to Diskette

Storing and Retrieving Data

5. Disk Manager 3 Cartridge Operation

Function Keys

Default Values

Correcting a Mistaken Entry

File Commands

Copying a File

Renaming a File

Deleting a File

Protecting or Unprotecting a File

Disk Commands

Cataloging a Diskette

Backing Up a Diskette

Renaming a Diskette

Initializing a Diskette

Disk Tests

Quick Test

Comprehensive Test

Single-Disk Processing

1926L HEXBUS Disk Drive/Controller $!o!=Bullet \mu=Question Mark Alpha=Number Sign$

Initializing a Diskette from BASIC The HEXBUS FORMAT MEDIA Subcommand The Compact Computer FORMAT Command

7. Storing Programs on Diskettes Program-Naming Conventions The SAVE Command Saving Protected Programs Verifying a Program The OLD Command Deleting a Program

RUN as a Program Statement

Saving Programs in MERGE Format

8. Home Computer File-Access Statements

The OPEN Statement The PRINT a Statement The INPUT a Statement The LINPUT a Statement The EOF Function

The RESTORE Statement

The REC Function The CLOSE Statement The DELETE Statement

Compact Computer File-Access Statements

The OPEN Statement The PRINT α Statement The INPUT α Statement The LINPUT a Statement The EOF Function The RESTORE Statement The CLOSE Statement The DELETE Statement

10. Using Data Files

Sequential Access

Creating a Sequential File Reading a Sequential File Additional Sequential Access Examples

Relative Access

Creating a Relative File Updating a Relative File

Maintaining a Sorted Relative File

11. Reading a Diskette's Catalog

Reading the Catalog with a Home Computer Reading the Catalog with a Compact Computer 1926L HEXBUS Disk Drive/Controller Part 1 to:=Bullet μ =Question Mark Alpha=Number Sign

Appendix A—Home Computer I/O Error Codes
Appendix B—Disk Manager 3 Error Codes
Appendix C—Compact Computer 41 I/O Error Codes
Appendix D—Home Computer Diskette/Cassette Operations
 Transferring Programs Between Cassettes and Diskettes
 Transferring Data Files from Cassette to Diskette
Appendix E—Table of Diskette Storage Capacities
Maintenance and Service Information
 In Case of Difficulty
 Service Centers
 If You Have Questions or Need Assistance
Warranty Information
Glossary

Introduction

The Texas Instruments Disk Drive/Controller 5102 is a versatile, efficient, and fast information storage device for Texas Instruments computers that have a built-in HEX-BUSTM communications port or use the HEX-BUS Interface peripheral. As part of the expanding line of low-cost computer peripheral devices that use the HEX-BUS interface, the Disk Drive/Controller connects directly to the Texas Instruments Compact Computer 41. The Disk Drive/Controller can be connected to the Texas Instruments 99/4A Home Computer by means of the HEX-BUS Interface peripheral, sold separately.

The TI Disk Drive/Controller package consists of the following components:

DISK DRIVE/CONTROLLER—The Disk Drive/Controller contains one slimline, horizontally—mounted disk drive and the electronic circuitry and software needed to control that disk drive and up to three additional disk drives. The built—in disk drive supports 41-track, double—sided, double—density operation. Additional disk drives for use with the Disk Drive/Controller are sold separately by Texas Instruments (Disk Drive model 5202).

DISK MANAGER 3 CARTRIDGE—The Disk Manager 3 Solid State Cartridge is designed for use with the Texas Instruments 99/4A Home Computer. The Disk Manager 3 cartridge contains software to help Home Computer owners manage and maintain the information on their diskettes.

HEX-BUSTM DATA CABLE—The HEX-BUS data cable connects your computer to the Disk Drive/Controller and provides a pathway for the exchange of data between the two devices.

AC POWER ADAPTER—The AC power adapter plugs into a standard 115-volt AC outlet and supplies power to the Disk Drive/Controller.

OWNER'S MANUAL—This manual contains instructions for setting up and using the Disk Drive/Controller with a Texas Instruments 99/4A Home Computer and Texas Instruments Compact Computer 40.

1926L HEXBUS Disk Drive/Controller Part 1
:o!=Bullet µ=Question Mark Alpha=Number Sign

The Disk Drive/Controller package does not contain diskettes. Diskettes suitable for use with the Disk Drive/Controller are sold separately by Texas Instruments and other manufacturers.

Note: Use of the Disk Drive/Controller with the 99/4A Home Computer requires the HEX-BUSTM Interface peripheral, sold separately.

FEATURES OF THE DISK DRIVE/CONTROLLER

The Texas Instruments Disk Drive/Controller represents a major advance over other disk-drive systems. Because the control hardware and software needed for the disk drive is built into the unit, the Disk Drive/Controller requires no additional equipment. The built-in density recognition feature of the Disk Drive/Controller enables it to automatically read from and write to diskettes initialized in single-density or double-density format. The Disk Drive/Controller also automatically recognizes diskettes in single-sided or double-sided format.

With the Disk Drive/Controller connected to your computer, you can:

- !o! RUN PREPROGRAMMED SOFTWARE—You can easily load and execute the wide variety of business, educational, and entertainment software packages available on diskettes from Texas Instruments.
- !o! INITIALIZE DISKETTES—A new diskette must be initialized before it can store information. The Disk Drive/Controller contains software that enables you to issue BASIC commands to initialize a diskette for 41-track, double-sided, double-density operation.
- io! SAVE AND RETRIEVE PROGRAMS—BASIC commands enable you to store programs quickly and accurately. Program retrieval is equally convenient.
- io: STORE AND RETRIEVE DATA—You can store data entered during program execution or generated by a program. A double-sided, double-density diskette can store over 320,000 characters of data.

io: UPDATE DATA—A BASIC program can easily update the information in your files so that they contain the latest, most accurate data.

FEATURES OF THE DISK MANAGER 3 CARTRIDGE

The Disk Manager 3 cartridge contains software designed to help Home Computer owners manage and maintain the information stored on their diskettes. With the Disk Manager 3 cartridge plugged into the computer console, you can:

- :o: INITIALIZE DISKETTES IN DIFFERENT STORAGE FORMATS—New diskettes can be initialized in the following storage formats:
 - !o! Single-sided, Single-density
 - !o! Single-sided, Double-density
 - :o: Double-sided, Single-density
 - :o! Double-sided, Double-density
- !o! NAME DISKETTES—Each diskette is assigned a name when it is initialized.

 This name can be checked by a BASIC program to verify that the proper diskette is in place. The Disk Manager 3 cartridge enables you to rename a diskette without reinitializing it.
- !o! DELETE, COPY, AND RENAME FILES—Although these operations can be performed via BASIC statements, the Disk Manager 3 cartridge enables you to perform these functions without writing a BASIC program.
- !o! MODIFY FILE PROTECTION STATUS—You may want to prevent a file from being erased or changed. With the Disk Manager 3 cartridge, you can assign a protected status to a program or data file that prevents the file from being deleted or edited. The cartridge also enables you to remove the protected status from a file previously protected by the Disk Manager 3 cartridge.

- :o: COPY DISKETTES—A diskette is a very reliable storage medium. However, data stored on a diskette can be totally or partially lost if the diskette is damaged. To insure against data loss, it is customary to make duplicate or "backup" diskettes. With the Disk Manager 3 cartridge, you can easily copy diskettes for backup even if you have only one disk drive.
- !o! PERFORM DIAGNOSTIC TESTS—To ensure that the disk system is operating correctly, the Disk Manager 3 cartridge offers several tests designed to check the components of the Disk Drive/Controller and the diskettes you are using.

COMPACT COMPUTER DISK UTILITIES

Compact Computer owners, using BASIC commands and statements, can initialize a diskette for 40-track, double-sided, double-density operation; store, verify, and retrieve programs; read and write data files; and catalog diskettes. An optional Disk Utilites package (sold separately) enables Compact Computer owners to perform the same functions available with the Disk Manager 3 cartridge.

1926L HEXBUS Disk Drive/Controller Part 1 to!=Bullet μ =Question Mark Alpha=Number Sign

Set-Up Instructions

This section contains instructions for setting up the Disk Drive/Controller and connecting it to the 99/4A Home Computer and Compact Computer 40. After setting up the Disk Drive/Controller as described in "Setting Up the Disk Drive/Controller," refer to the appropriate installation section for instructions on connecting the unit to your computer.

PACKAGE CONTENTS

Carefully unpack the contents of the Disk Drive/Controller box. In addition to this manual, the package should contain the following components:

(Drawing 1 showing components of package with components labeled)

<u>Note:</u> The Disk Drive/Controller is a precision device which uses moving parts with exacting tolerances, so handle it carefully. Save the packing material and carton for storing or transporting the Disk Drive/Controller.

9/16/83 FINAL DRAFT

SETTING UP THE DISK DRIVE/CONTROLLER

CAUTION

Use only diskettes that are certified for operation at or above 55 degrees centigrade (131 degrees Fahrenheit) if you intend to use the Disk Drive/Controller in environments in which the ambient temperature is above 35 degrees centigrade (95 degrees Fahrenheit). To prevent excessive heat build-up in all environments, do not cover or block the ventilation slots in the top of the Disk Drive/Controller. Failure to observe these precautions can result in damage to diskettes installed in the disk drive.

- 1. Ensure that the Disk Drive/Controller is turned off. The ON/OFF switch is located on the back of the unit.
- 2. Plug the power cord of the AC adapter into the outlet marked POWER on the back of the Disk Drive/Controller. Notice that the plug lines up only Note: Use only AC power adapter model number AC 9610.

(Drawing 2 showing AC adapter being plugged in)

- 3. Plug the power cord of the AC power adapter into a standard 115-volt AC power outlet.
- The Disk Drive/Controller has two HEX-BUSTM connectors. Plug either 4. end of the HEX-BUS data caple into one of the HEX-BUS connectors located on the back of the Disk Drive/Controller. The connector is molded so that the cable fits into it only when the raised tab on the cable is turned up. If the cable does not plug easily into the connector, check to see that you are inserting it correctly.

Note: The second <u>HEX-BUS</u> connector on the Disk Drive/Controller can be used to connect additional HEX-BUS peripherals to your computer system.

5. If you are using additional disk drives (sold separately), connect them to the Disk Drive/Controller.

CAUTION

Do not stack additional disk drives on the Disk Drive/Controller.

Refer to the owner's manual supplied with the additional disk drives for instructions on connecting them to the Disk Drive/Controller.

6. You are now ready to connect the Disk Drive/Controller to your computer.

Refer to the appropriate installation section for instructions on connecting the unit to your computer.

CAUTION

Before transporting the Disk Drive/Controller, disconnect the data cable to prevent damage to the connector pins. For shipment over long distances, pack the system securely.

CONNECTING THE DISK DRIVE/CONTROLLER TO A 99/4A HOME COMPUTER

This section provides instructions for connecting the Disk Drive/Controller to a 99/4A Home Computer. Note: Connecting the Disk Drive/Controller to a 99/4A Home Computer requires the HEX-BUSTM Interface peripheral, sold separately.

- 1. If you have other peripherals connected to the computer, wait for their activity to stop. Then turn them off.
- 2. Turn off the computer.
- 3. Be certain that the <u>HEX-BUS</u> Interface peripheral is turned off. The power switch is located on the back of the unit. (If the <u>HEX-BUS</u> Interface peripheral is not connected to the computer, install it at this time as described in the owner's manual supplied with the Interface.)
- 4. Connect the free end of the data cable from the Disk Drive/Controller to the <u>HEX-BUS</u> connector on the back of the <u>HEX-BUS</u> Interface. The raised tab on the cable must align with the top edge of the <u>HEX-BUS</u> connector. If you are using additional peripherals with the computer, connect them at this time as described in the manuals supplied with the peripherals.

(Drawing 3 showing cable being connected to HEXBUS Interface.)

Note: $\underline{\text{HEX-BUS}}$ peripherals are designed to link to the $\underline{\text{HEX-BUS}}$ Interface in any order. As an alternative to connecting the data cable of the Disk Drive/Controller directly to the $\underline{\text{HEX-BUS}}$ Interface, you can connect it to the $\underline{\text{HEX-BUS}}$ connector of another peripheral.

5. Turn the system on, beginning with the peripheral at the outermost end of the <u>HEX-BUS</u> chain. When the Disk Drive/Controller is turned on, the red power indicator in the front of the unit should light, and the motor should turn on briefly. The last unit you should turn on is the computer.

CAUTION

Do not turn the Disk Drive/Controller on or off with a diskette in the disk drive. The sudden surge of current that occurs when the system is turned on can alter or erase information stored on the diskette.

CONNECTING THE DISK DRIVE/CONTROLLER TO A COMPACT COMPUTER 4D

This section provides instructions for connecting the Disk Drive/Controller to a Compact Computer $4\mathbf{D}$.

- 1. If you have other peripherals connected to the Compact Computer, wait for their activity to stop. Then turn them off.
- 2. Turn off the computer.
- 3. Connect the free end of the data cable from the Disk Drive/Controller to the <u>HEX-BUS</u> connector on the back of the Compact Computer. The raised tab on the cable must align with the top edge of the <u>HEX-BUS</u> connector. If you are using additional peripherals with the computer, connect them at this time as described in the manuals supplied with the peripherals.

(Drawing 4 showing cable being connected to CC-4D.)

Note: <u>HEX-BUS</u> peripherals are designed to link to the computer in any order. As an alternative to connecting the data cable of the Disk Drive/Controller directly to the computer, you can connect it to the HEX-BUS connector of another peripheral.

4. Turn the system on, beginning with the peripheral at the outermost end of the <u>HEX-BUS</u> chain. When the Disk Drive/Controller is turned on, the red power indicator in the front of the unit should light, and the motor should turn on briefly. The last unit you should turn on is the computer.

CAUTION

Do not turn the Disk Drive/Controller on or off with a diskette in the disk drive. The sudden surge of current that occurs when the system is turned on can alter or erase information stored on the diskette.

INSERTING AND REMOVING A DISKETTE

To insert a diskette into the disk drive, hold the diskette gently by the edge nearest the label, and insert it with the label facing up into the slot in the front of the Disk Drive/Controller. Secure the diskette in the disk drive by carefully turning down the diskette engage/release lever. The disk drive motor will turn on briefly as soon as the diskette is secured.

CAUTION

The engage/release lever is designed to turn down only when a diskette is installed in the disk drive. Forcing the lever downward without a diskette installed in the disk drive can seriously damage the Disk Drive/Controller.

(Drawing 5 showing diskette being inserted into drive.)

CAUTION

Never insert or remove diskettes when the red "working" light is on. The "working" light indicates that the computer is reading data from or writing data to a diskette. If you remove a diskette while the "working" light is turned on, essential information regarding the location of data may not be written to the diskette, and the data may be lost.

To remove the diskette, lift the lever and pull out the diskette. Store the diskette in its protective envelope.

Diskette Information

The 5 1/4-inch "mini-floppy" diskette used by your Texas Instruments Disk Drive/Controller provides a high performance, reliable storage medium. Mini-floppy diskettes, or simply <u>diskettes</u> as they are usually called, are constructed of a round plastic film coated with a thin layer of iron oxide. This film is enclosed in a protective jacket to allow easy handling of the diskette and prevent fingerprints or foreign matter from contaminating the film's surface. For additional protection, diskettes are usually supplied with a storage envelope for covering the diskette when it is not in use.

The large hole in the center of the diskette is used to rotate the diskette when it is installed in the disk drive. The slots cut in the jacket are used for reading and writing data, timing revolutions, and "write-protecting" the diskette. A standard 5 1/4-inch diskette is shown in Fig. 1.

(Drawing 7)

Fig. 1 Drawing of a standard diskette showing: (a) protective jacket, (b) read/write window, (c) write-protect notch, (d) timing (or index) hole, (e) diskette label, and (f) storage envelope.

Information is stored on the diskette by magnetizing very small areas in the iron oxide layer of the diskette. Because your Disk Drive/Controller can store over two million of these small magnetized areas on a single diskette, it is essential that the information be well organized. The Disk Drive/Controller organizes a diskette by "writing" identifying information to the diskette that divides it into units known as tracks and sectors. The process of placing this organizational information on a diskette is known as initializing (or formatting) the diskette. Every new diskette must be initialized before it can be used.

Fig. 2 illustrates the manner in which tracks and sectors are arranged on a diskette. Tracks are arranged in concentric circles, with the first track being the outermost circle and the last track being the innermost circle. Sectors are subdivisions of tracks. Once a diskette has been initialized, the computer can store up to 256 bytes of data in each sector. (A byte is enough storage space to store one character.) Each sector of an initialized diskette is "empty" until the computer writes information to it.

Fig. 2 Drawing showing the arrangement of tracks and sectors on a diskette.

The number of tracks and sectors that can be placed on a diskette depends on two factors: the capability of a disk drive and the storage capacity of a diskette. The TI Disk Drive/Controller has the capability to support several storage capacities. You can, for example, initialize a diskette in either double-density or single-density format. If single-density format is chosen, each track contains 9 sectors. If double-density format is chosen, each track contains 16 sectors. You can also select whether information is recorded on one or both sides of a diskette. Recording data on one side of a diskette is known as single-sided operation. Recording data on both sides of a diskette is known as double-sided operation.

As a rule, you will probably use double-sided, double-density format so that you can store a maximum amount of information per diskette.

Diskettes are manufactured in a variety of shapes, sizes, and capacities. If you intend to use double-sided, double-density format, you should purchase diskettes that are certified for that type of operation. Diskettes that support this capability will have the words "double-sided, double-density" written somewhere on their label.

<u>Note:</u> Diskettes that are initialized in double-density or double-sided format can not be read from or written to by the Texas Instruments PHP125**D** or PHP18**DD** Disk Memory Systems.

WRITE-PROTECTING DISKETTES

Diskettes can be "write-protected" by covering the write-protect notch of the diskette with a small piece of tape. (Pieces of tape designed specifically for this purpose are usually included with the diskettes.) The write-protect notch is located along the upper-right edge of the diskette, as shown in Fig. 1. By covering this notch, you mechanically instruct the Disk Drive/Controller to reject any command to write information to the diskette. The computer can read from a write-protected diskette, but it cannot write to the diskette.

The purpose of write-protecting a diskette is to prevent any loss of data on the diskette. You should write-protect diskettes containing finished programs that you do not intend to change, and diskettes containing duplicate (backup) copies of files that you are keeping in case the originals are damaged or lost.

To remove the write-protected status of a diskette, remove the tape used to cover the write-protect notch.

LABELING AND HANDLING PRECAUTIONS

When handling diskettes, do not touch the recording surface of the diskette (visible through the read/write window of the diskette jacket). Handle the diskette by its protective jacket when inserting and removing it from a disk drive, and place it in its storage envelope when it is not in use.

Avoid writing on the protective jacket with a pencil or ballpoint pen, because the pressure of writing with these instruments can damage the diskette. Use a felt-tip pen, or write on the label before it is put on the protective jacket. Adhesive labels are usually supplied with the diskettes.

Any magnet or magnetic field can affect data on a diskette. Do not bring a magnet near a diskette, and do not put a diskette near anything that generates a magnetic field, such as your Disk Drive/Controller or a television, monitor, speaker, electric motor, transformer, etc. Also be aware of more obscure magnetic sources, such as magnetic cabinet latches, magnets in children's toys, and magnetic note holders. As a rule, do not store diskettes within three feet of a magnetic source.

Do not turn the Disk Drive/Controller on and off with a diskette installed in the disk drive. The sudden surge of current that occurs when the system is turned on and off can alter or erase information stored on the diskette.

Keep diskettes out of direct sunlight and away from heat.

Avoid exposing diskettes to cigarette smoke, ashes, dust, moisture, or other surface contaminants.

A Quick Demonstration of the Disk System

This section provides a quick demonstration of the capabilities of the Disk Drive/Controller. The section is divided into two parts, a demonstration for Home Computer owners, and a demonstration for Compact Computer owners. If you are unfamiliar with disk-drive operation, read the appropriate demonstration section before proceeding to the more detailed reference sections of this manual.

FOR HOME COMPUTER OWNERS

This section illustrates initializing a diskette, saving a BASIC program on the diskette, writing and reading data from the diskette, and displaying the catalog of a diskette using the Disk Manager 3 cartridge.

Initializing a Diskette

To begin this exercise, turn on the Disk Drive/Controller, any additional disk drives, the display monitor, and the computer console, in that order. (Refer to the section entitled "Set-Up Instructions" for directions on setting up the Disk Drive/Controller and connecting it to your computer.) Insert a new diskette into the disk drive and lower the drive lever. Insert the Disk Manager 3 cartridge into the computer.

When the Home Computer title screen appears, press any key to display the master selection list. Then select the English-language version of the Disk Manager 3 cartridge. After a brief title sequence, the following selection list appears:

DISK MANAGER

- 1 FILE COMMANDS
- 2 DISK COMMANDS
- 3 DISK TESTS
- 4 SET ALL COMMANDS FOR SINGLE-DISK PROCESSING

YOUR CHOICE 1

The number 1 and the cursor (a small rectangle) alternately flash in the space following the \underline{YOUR} CHOICE \underline{u} prompt. Number 1 is the default option and can be selected by pressing \underline{ENTER} . For this demonstration, select option 4. Note that choosing option 4 speeds up the Disk Manager functions when only one disk drive is being used. After you press $\underline{4}$, the screen changes so that option 4 reads $\underline{SINGLE-DISK}$ PROCESSING HAS BEEN INITIALIZED.

Next, select the DISK COMMANDS option by pressing $\underline{2}$. The following selection list appears:

DISK COMMANDS

- 1 CATALOG DISK
- 2 BACKUP DISK
- 3 MODIFY DISK NAME
- 4 INITIALIZE NEW DISK

YOUR CHOICE 1

(Option number 1 is the preselected default.)

A new diskette must be initialized (formatted) before the computer can use it. The initialization process places index information on the diskette that the computer uses to locate program and data files.

Start the initialization process by pressing $\underline{4}$. The following display appears.

INITIALIZE NEW DISK DISK DRIVE $(1-4)_{\mu}$ 1 DISK NOT INITIALIZED NEW DISKNAME $_{\mu}$

(*** Note to typesetter: These are strange underlines ***)

Note

If the diskette has been initialized previously, the name assigned to the diskette when it was initialized will be displayed instead of the message DISK NOT INITIALIZED. In this case, consider whether you want to initialize the diskette again, because initializing a diskette erases all information stored on the diskette.

To proceed with the initialization process, answer the prompt $\underline{\text{NEW DISKNAME}}_{\mu}$ with the name you want to assign to the disk. For this demonstration, type the name $\underline{\text{DEMO}}$ and press $\underline{\text{ENTER}}$.

The prompt \overline{TRACKS} PER $\overline{SIDE_{\mu}}$ appears next. Press \overline{ENTER} to accept the default value of 40 tracks.

The prompt $\underline{\text{DOUBLE SIDED }(Y/N)_{\mu}}$ appears next. Your response to this prompt depends upon the type of diskette you are using. If your diskette supports double-sided operation, select the double-sided option by pressing $\underline{\text{ENTER}}$. (A diskette that can be initialized for double-sided operation will usually have the words "double-sided" printed on the label of the diskette.) If you are using a single-sided diskette, press N for single-sided initialization.

The prompt <u>DOUBLE DENSITY $(Y/N)_{\mu}$ </u> appears next. Press <u>ENTER</u> if your diskette supports double-density operation. (A diskette that can be initialized for double-density operation will usually have the words "double-density" printed on the label of the diskette.) If you are using a single-density diskette, press N for single-density initialization.

After your response to the <u>DOUBLE DENSITY</u> prompt, the computer initializes and verifies the diskette. This process takes about seven minutes for a 4**D**-track, double-sided, double-density diskette. While the computer initializes the diskette, the following message appears on the screen:

INITIALIZE NEW DISK
WORKING.... PLEASE WAIT.

The computer then displays the number of each sector on the diskette as it verifies that the sector is usable. This verification process is an important check to ensure that information can be reliably stored on and retrieved from the diskette.

After the verification procedure is complete, the following message appears:

DSK1 - DISKNAME= DEMO
AVAILABLE= nnnn USED= nnnn

The number following the prompt $\underline{AVAILABLE}=$ indicates the number of sectors available for data storage, and the number following the prompt $\underline{USED}=$ indicates the number of unusable sectors. The computer automatically prevents information from being stored in sectors that are found to be unusable.

Unusable sectors are usually caused by flaws in a diskette. If the diskette you are initializing shows a large number of used sectors, then you selected the wrong format for your diskette or the diskette is badly flawed. If the problem is a badly flawed diskette, you should consider not using the diskette because it may prove unreliable.

The numbers actually shown on your screen depend upon the initialization options you selected. Refer to Appendix E for a chart of expected values for various combinations of the options.

1926L HEXBUS Disk Drive/Controller Part 1 to!=Bullet μ =Question Mark Alpha=Number Sign

After the diskette has been initialized and verified, the following message appears:

COMMAND COMPLETED
PRESS: PROC'D, REDO,
BEGIN, OR BACK

Remove the diskette and label it DEMO. You may also want to write the current date on the diskette so that you can easily identify it later. After labeling the diskette, reinsert it in the disk drive.

The first stage of the demonstration, initializing a diskette, is complete. To prepare for the next stage, press \underline{BEGIN} to return to the main Disk Manager selection list. Then press \underline{BACK} . When the master title screen appears, you are ready to proceed to the second stage of the demonstration.

Saving a Program to Diskette

Next you will save and retrieve a program from the diskette.

Begin by pressing any key to get the master selection list. Then select BASIC (or Extended BASIC). When the BASIC & prompt appears, enter the following BASIC program.

110 REM DEMO PROGRAM--HOME COMPUTER VERSION

11D CALL CLEAR

120 OPEN al: "DSK1.DATAFILE", RELATIVE, INTERNAL

13**D** FOR R=1 TO 1**D**

140 INPUT "ENTER A NAME: ":NAMES

15**D** NAME &= SEG&(NAME & , 1, 79)

160 PRINT al: NAMES

17**0** NEXT R

180 INPUT "DISPLAY WHICH RECORD" ":A

190 IF A<1 THEN 180

200 IF ANT THEN 240

210 INPUT al, REC A-1: NAMES

1926L HEXBUS Disk Drive/Controller Part 1 to:=Bullet μ =Question Mark Alpha=Number Sign

22**D** DISPLAY "REC";A;"= ";NAMES::

23 D GOTO 18 D

240 CLOSE al

After you have entered the program and verified that you have made no entry mistakes, you are ready to save the program on the diskette in drive 1. Type

SAVE DSK1.PROGRAM1

and press <u>ENTER</u>. This command instructs the computer to save the program in memory to disk drive 1. The program file is assigned the name PROGRAM1.

While the computer stores the program on the diskette, the "working" light on the disk drive is turned on. After the light goes off and the flashing cursor reappears, type <u>NEW</u> and press <u>ENTER</u> to erase the program from the computer's memory. Then type <u>LIST</u> and press <u>ENTER</u> to verify that there is no longer a program in memory.

To load the program back into the computer, type

OLD DSK1.PROGRAM1

and press $\underline{\mathsf{ENTER}}$. After the flashing cursor reappears, $\underline{\mathsf{LIST}}$ the program.

Storing and Retrieving Data

RUN the program written in the previous section. Each time the ENTER A NAME prompt appears, type in a new name and press ENTER. Observe that the "working" light in the front panel of the disk drive is turned on each time the disk controller writes data to the diskette.

After you have entered ten sample names, the prompt <u>DISPLAY WHICH RECORD</u> $_{\mu}$ will appear. Enter a number from 1 through 10. The name stored in the corresponding record in the file is retrieved and displayed on the screen. Experiment with the program by entering different record numbers. Notice that you can retrieve the records in any order, as often as you like.

When you are finished, enter a number larger than 10. The file is closed and the program execution stops. Enter <u>BYE</u> to leave BASIC and return to the master title screen.

Displaying a Diskette's Catalog

Press any key to advance to the computer's master selection list. Then select the Disk Manager 3 cartridge. When the Disk Manager selection list appears, press 4 to select single-disk processing.

Choose option 2, DISK COMMANDS, again. The DISK COMMANDS selection list appears as shown below.

DISK COMMANDS

- T CATALOG DISK
- 2 BACKUP DISK
- 3 MODIFY DISK NAME
- 4 INITIALIZE NEW DISK

YOUR CHOICE 1

Choose option 1, CATALOG DISK, by pressing <u>ENTER</u>. The following display appears:

CATALOG DISK

DISK DRIVE (1-4)μ 1 DISKNAME= DEMO

WHERE DO YOU WANT LISTINGL

- 1 SCREEN
- 2 SOLID STATE PRINTER
- 3 RS232 INTERFACE
- 4 OTHER

YOUR CHOICE 1

Press ENTER to indicate that you want the catalog displayed on the monitor screen. The following display appears.

CATALOG DISK

DSK1 - DISKNAME= DEMO

AVAILABLE=nnnn USED=nnnn

FILENAME	SIZE	TYPE	LRL	P
				_
DATAFILE	5	I/F	81	
PROGRAM1	3	PGM		

COMMAND COMPLETED
PRESS: PROC'D, REDO,
BEGIN, OR BACK

At the top of the screen, the catalog displays the number of the disk drive, the name of the diskette, and the number of available (unused) and used sectors. Because the numbers of sectors in these categories vary depending upon the storage capacity of the diskette and disk drive you are using, they are represented by nnnn in this manual.

Next the catalog displays the filename, the number of sectors occupied by the file (its size), and certain characteristics of the file. For this example, the catalog shows that the first file is a data file in INTERNAL format with FIXED-length records. The records are 81 bytes long. (These characteristics are discussed in the section of this manual entitled "Home Computer File-Access Statements.") The second file is a program file. Neither of the files is protected (if a file is protected, the letter Y will appear under the P column for the files that are protected).

This concludes the demonstration of the Disk Drive/Controller for the Home Computer. The capabilities of the disk system are described in greater detail in later sections.

To leave the Disk Manager 3 program, press BEGIN and BACK.

1926L HEXBUS Disk Drive/Controller Part 1 to!=Bullet μ =Question Mark Alpha=Number Sign

FOR COMPACT COMPUTER OWNERS

This section illustrates initializing a diskette, saving a BASIC program on the diskette, and writing and reading data from the diskette.

Initializing a Diskette

To begin the exercise, turn on the Disk Drive/Controller and then the Compact Computer, in that order. (Refer to the section entitled "Set-Up Instructions" for directions on setting up the Disk Drive/Controller and connecting it to your computer.) Insert a new diskette into the disk drive and lower the drive lever. The diskette that you use must support 40-track, double-sided, double-density operation, or you will not be able to proceed with this demonstration. If a diskette can be initialized for double-sided, double-density operation, the words "double-sided" and "double-density" will usually appear on the label of the diskette.

A new diskette must be initialized (formatted) before the computer can use it. The initialization process places index information on the diskette that the computer uses to locate program and data files.

CAUTION

Initializing a diskette erases any information stored on the diskette.

To initialize the diskette in drive 1, type

FORMAT 101

and press <u>ENTER</u>. This command instructs the computer to initialize the diskette for 40-track, double-sided, double-density operation. The "working" light in the front panel of the disk drive is turned on for approximately three minutes while the Disk Drive/Controller initializes the diskette. The diskette is assigned the name TI-DISK.

1926L HEXBUS Disk Drive/Controller Part 1 to:=Bullet μ =Question Mark Alpha=Number Sign

If the diskette does not initialize properly, the computer displays an error code. Refer to Appendix C for a list of Compact Computer 4D error codes and their probable causes.

Saving a Program to Diskette

Next you will save and retrieve a program from the diskette.

Enter the following BASIC program.

100 :Demo Program—Compact Computer Version

110 OPEN al, "101. DATAFILE", RELATIVE, INTERNAL

120 FOR R=1 TO 10

130 INPUT "Enter a name: ";NAMES

140 PRINT al, NAMES

150 NEXT R

160 INPUT "Display which record"; A

170 IF A<1 THEN 160

180 IF A>10 THEN 220

190 INPUT al, REC A-1, NAME &

200 DISPLAY "Rec"; A; "= "; NAME &: PAUSE

210 GOTO 16D

220 CLOSE al

After you have entered the program and verified that you have made no entry mistakes, you are ready to save the program on the diskette in drive 1. Type

SAVE "101.PROGRAM1"

and press $\underline{\sf ENTER}$. This command instructs the computer to save the program in memory to disk drive 1. The program file is assigned the name PROGRAM1.

While the computer stores the program on the diskette, the "working" light on the disk drive is turned on. After the light goes off and the flashing cursor reappears, type <u>NEW</u> and press <u>ENTER</u> to erase the program from the computer's 1926L HEXBUS Disk Drive/Controller Part 1 $!o!=Bullet \mu=Question Mark Alpha=Number Sign$

memory. Then type \underline{LIST} and press \underline{ENTER} to verify that there is no longer a program in memory.

To load the program back into the computer, type

OLD "101.PROGRAM1"

and press ENTER. After the flashing cursor reappears, LIST the program.

Storing and Retrieving Data

RUN the program written in the previous section. Each time the <u>Enter a name</u> prompt appears, type in a new name and press <u>ENTER</u>. The light in the front panel of the disk drive is turned on each time the disk controller writes data to the diskette.

When the prompt Display which $record_{\mu}$ appears, enter a number from 1 through 1**D**. The name stored in the corresponding record in the file is retrieved and displayed. Experiment with the program by entering different record numbers. Notice that you can retrieve the records in any order, as often as you like.

When you are finished, enter a number larger than lo. The file is closed and program execution stops.

This concludes the demonstration of the Disk Drive/Controller for the Compact Computer. The capabilities of the disk system are described in greater detail in later sections.

Disk Manager 3 Cartridge Operation

The Disk Manager 3 cartridge is supplied with the Disk Drive/Controller for use with the system by Home Computer owners. The Disk Manager 3 cartridge enables you to copy, rename, delete, protect, and unprotect files, and to catalog, backup, rename, and initialize diskettes. With the Disk Manager, you can also test your Disk Drive/Controller and diskettes. These functions are performed by selecting options from lists of Disk Manager commands and responding to questions (prompts) on the display screen. A general overview of the functions of the Disk Manager 3 cartridge is shown in Fig. 3.

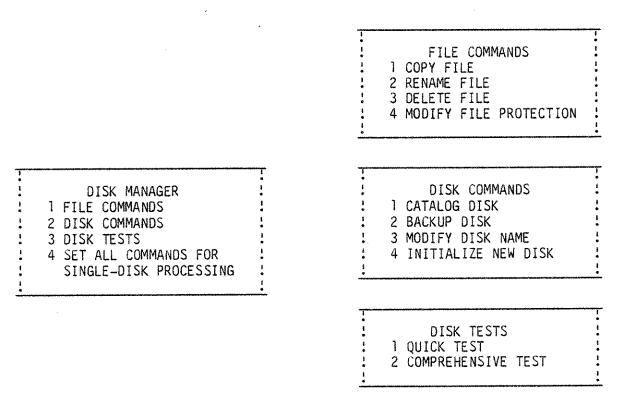


Fig. 3 Drawing illustrating the arrangement of selection lists in the Disk Manager 3 cartridge. (For use with a Home Computer only.)

FUNCTION KEYS

The following function keys are used with the Disk Manager 3 cartridge. (The function keys are accessed by pressing the \underline{FCTN} key in combination with certain other keys. The slip-in overlay provided with the Home Computer provides an easy method of identifying the function key sequences.)

<u>Function</u>	Key	
Name	Combination	Operation
CLEAR	FCTN 4	Pressing CLEAR when the message PRESS: CLEAR TO
•		CANCEL COMMAND is displayed cancels an operation such
		as backing up a diskette.
BEGIN	FCTN 5	Pressing BEGIN returns you to the initial Disk
		Manager selection list.
PROC'D	FCTN 6	Pressing PROC'D causes the computer to perform the
		action just requested or to repeat the current
		process. This function is available when the message
		PRESS: PROC'D, REDO, BEGIN, OR BACK is displayed.
AID	FCTN 7	Pressing <u>AID</u> displays a list of the most common Disk
		Manager error messages.
<u>REDO</u>	FCTN 8	Pressing REDO enables you to start again at the
		beginning of the current screen display to make
		changes in your answers to prompts.
BACK	FCTN 9	Pressing BACK returns you to the previous selection
		list. Pressing <u>ENTER</u> when the message PRESS: PROC'D,
		REDO, BEGIN, OR BACK is displayed is equivalent to
		pressing <u>BACK</u> .
TIUD	FCTN =	Pressing QUIT interrupts the Disk Manager 3 program
		and returns you to the computer's master title screen.

CAUTION

Do not use $\underline{\text{QUIT}}$ (FCTN =) to leave the Disk Manager 3 cartridge while any Disk Manager commands are in operation. Using $\underline{\text{QUIT}}$ while the Disk Manager is performing a command can result in the loss of programs or data stored on the diskette. The following procedure is recommended for leaving the Disk Manager 3 cartridge:

- l. Press $\underline{\mathsf{BEGIN}}$ to return to the initial Disk Manager selection list.
- 2. Press BACK to leave the cartridge.

1926L HEXBUS Disk Drive/Controller Part 1 to!=Bullet μ =Question Mark Alpha=Number Sign

DEFAULT VALUES

Many Disk Manager prompts are displayed with preprogrammed answers, known as default responses. Default responses represent common or usual answers to a prompt. If the default response is the one you want, just press <u>ENTER</u>. To give a different response, press the appropriate key or keys for the response you want.

As a rule, you do not have to press $\underline{\mathsf{ENTER}}$ when a one-key response such as the number of a disk drive is expected. As soon as the key is pressed, the computer proceeds with the command or operation.

CORRECTING A MISTAKEN ENTRY

Typing errors are easy to correct if you haven't pressed <u>ENTER</u>. Use the <u>LEFT ARROW (FCTN S)</u> or <u>RIGHT ARROW (FCTN D)</u> key to move the cursor to the character you want to change, and retype the character. You can also erase a line (before pressing <u>ENTER</u>) by pressing <u>CLEAR (FCTN 4)</u>. In certain applications, such as entering a diskname or filename, you can also use <u>DEL (FCTN 1)</u> and <u>INS (FCTN 2)</u>.

After you have pressed $\underline{\mathsf{ENTER}}$, you must press $\underline{\mathsf{REDO}}$ to change or correct an entry.

FILE COMMANDS

The first option in the DISK MANAGER selection list is FILE COMMANDS. This option accesses commands that operate on individual files rather than the entire diskette.

When you choose the FILE COMMANDS option, the following selection list appears:

FILE COMMANDS

- 1 COPY FILE
- 2 RENAME FILE
- 3 DELETE FILE
- 4 MODIFY FILE PROTECTION

YOUR CHOICE 1

Copying a File

The COPY FILE option enables you to copy a file from one diskette to another, or to copy the file to another name on the same diskette. When you choose this option, the following display appears:

COPY FILE SOURCE DISK (1-4)_µ 1

Your response to the prompt <u>SOURCE DISK</u> $(1-4)_{\mu}$ indicates the disk drive that contains the file you want to copy. After you answer the prompt, the name of the diskette is displayed and the prompt <u>FILENAME</u> appears. Enter the name of the file that you wish to copy.

The prompt <u>DESTINATION DISK $(1-4)_{\mu}$ </u> appears next. Your answer determines to which disk drive the file is to be copied. If you enter the same drive number for both the <u>SOURCE DISK</u> and <u>DESTINATION DISK</u> prompts, the file can be duplicated under another name on the same diskette. If you enter a different number for each of the two prompts, the file is copied from one disk drive to another.

1926L HEXBUS Disk Drive/Controller Part leads to LeBullet μ =Question Mark Alpha=Number Sign

After you answer the <u>DESTINATION DISK</u> prompt, the name of the destination diskette is displayed and the prompt <u>FILENAME</u> μ appears. Enter the name you wish to assign to the new copy of the file. If you intend to make a duplicate copy of the file on the same diskette, be sure to enter a different name for the new copy.

Renaming a File

The RENAME FILE option enables you to change the name of a file. When you choose this option, the following display appears:

RENAME FILE DISK DRIVE (1-4)µ 1

Your response to the prompt <u>DISK DRIVE</u> $(1-4)_{\mu}$ indicates the disk drive that contains the file you want to rename. After you respond, the name of the diskette is displayed and the prompt <u>FILENAME</u> appears. Enter the current name of the file. When the prompt <u>NEW FILENAME</u> appears, enter the new name that you wish to assign to the file.

Deleting a File

The DELETE FILE option enables you to delete a single file or multiple files. When you choose this option, the following display appears:

DELETE FILE SELECTIVE (Y/N)µ N

Your answer to the prompt <u>SELECTIVE $(Y/N)_{\mu}$ </u> determines whether you delete one file or multiple files. Respond <u>N</u> to delete one file. Respond <u>Y</u> to delete more than one file.

The prompt DISK DRIVE $(1-4)\mu$ appears next. Enter the number of the disk drive containing the diskette with the file(s) you are going to delete. The name of

the diskette is displayed. If you are deleting only one file, the prompt ${\sf FILENAME}_{\tt U}$ appears. Enter the name of the file that you wish to delete.

If you have chosen to delete multiple files, the name of each file on the diskette is presented one at a time. Answer the prompt DELETE $(Y/N)_{\mu}$ with Y if you want to delete the file. Respond N if you don't want to delete the file. If you attempt to delete a file that is protected, the prompt OVERRIDE PROTECTION $(Y/N)_{\mu}$ appears. To delete the file even though it is marked as protected, press Y. If you do not wish to delete it, press N.

Protecting or Unprotecting a File

The MODIFY FILE PROTECTION option enables you to assign protection to, and remove protection from, a file. When you protect a file with this option, the computer places information in the catalog indicating that the file is protected. If a file is protected, it cannot be deleted, written to, or updated by BASIC commands until the protected status is removed. Attempts to perform these operations on a protected file will cause an error condition to occur. (The program can still be copied, backed up, or loaded into BASIC and edited or listed.)

Note: The file protection status assigned by the Disk Manager 3 cartridge is designed to prevent accidental deletion or modification of files. This protection status is not designed to protect proprietary programs from being loaded into BASIC and listed or edited. To protect proprietary BASIC programs from being listed or edited, save the program to diskette using the PROTECTED option of the SAVE command (Home Computer owners must have the Extended BASIC cartridge installed to use this option). Unlike the protected status assigned by the Disk Manager 3 cartridge, the PROTECTED option of the SAVE command cannot be cancelled or removed. (For more details on the PROTECTED option of the SAVE command, refer to the section of this manual entitled "Storing Programs on Diskettes.")

1926L HEXBUS Disk Drive/Controller Part 1 :o!=Bullet µ=Question Mark Alpha=Number Sign

When you choose the MODIFY FILE PROTECTION option, the following display appears:

MODIFY FILE PROTECTION DISK DRIVE (1-4)μ 1

Your response indicates which disk drive contains the file you want to modify. After you answer the prompt, the name of the diskette is displayed and you are asked for the name of the file whose protection status you wish to alter. Your answer to the prompt $\frac{PROTECT}{(Y/N)_{\mu}}$ determines the status of the file. If you answer \underline{Y} , the file is protected. If you answer \underline{N} , the file is unprotected.

DISK COMMANDS

The second option on the DISK MANAGER selection list is DISK COMMANDS. Choosing this option gives you access to the commands that affect an entire diskette.

When you choose the DISK COMMANDS option, the following selection list appears:

DISK COMMANDS

- 1 CATALOG DISK
- 2 BACKUP DISK
- 3 MODIFY DISK NAME
- 4 INITIALIZE NEW DISK

YOUR CHOICE 1

1926L HEXBUS Disk Drive/Controller Part 1 $0!=Bullet \mu=Question Mark Alpha=Number Sign$

Cataloging a Diskette

The CATALOG DISK option enables you to catalog the files on a diskette. When you choose this option, the following display appears:

CATALOG DISK DISK DRIVE (1-4)µ 1

Respond with the number of the disk drive you want to catalog. After you enter the drive number, the name of the diskette is displayed and the following choices appear:

WHERE DO YOU WANT LISTING "

- 1 SCREEN
- 2 SOLID STATE PRINTER
- 3 RS232 INTERFACE
- 4 OTHER

YOUR CHOICE 1

If you choose l, the catalog listing appears on the display screen. If the listing is too long to be displayed on the screen all at once, the lines are scrolled. Press the SPACE BAR a second time to start scrolling again.

If you choose 2, the catalog listing is printed on the TI Thermal Printer.

If you choose 3, the catalog listing is transferred to the RS232 interface using the standard RS232 default parameters (baud rate=300, data bits=7, parity=odd, and stop bits=1). See the RS232 interface manual for more information on this device.

If you choose 4, the prompt $\underline{\text{DEVICE NAME}_{\mu}}$ appears. You can then enter any valid device name. For example, to transfer the listing to the RS232 at 9600 baud with an 8-bit word length, enter

RS282.BA=9600.DA=8

1926L HEXBUS Disk Drive/Controller Part 1 !o!=Bullet u=Question Mark Alpha=Number Sign

To store the listing on the diskette in disk drive 1 under the filename CATALOG1, enter

DSK1.CATALOG1

The catalog listing has the following general format:

DSKn - DISKNAME= diskname AVAILABLE=nnnn USED=nnnn FILENAME SIZE TYPE LRL P

AAAAAAAAA nnnn PGM
BBBBBBBBB nnnn D/V nnnnn Y
CCCCCCCCCC nnnn D/F nnnnn
DDDDDDDDDD nnnn I/V nnnnn Y
EEEEEEEEEE nnnn I/F nnnnn

The first line lists the number of the disk drive and name of the diskette. The second line lists the number of free (AVAILABLE) sectors, and the number of sectors that are being used to store data and programs plus the number of sectors that are not usable (USED). The numbers actually shown on your screen depend upon the initialization options selected for your diskette. Refer to Appendix E for a chart of the storage capacities for different options.

Under the headings FILENAME, SIZE, TYPE, LRL, and P, the listing shows the following information for each file:

- io: Name of the file
- !o! Number of sectors used by the file
- io! Type of file
- !o! Record size (in bytes)
- :o: Protection status of the file

The FILENAME and SIZE parameters give the name and number of sectors occupied by the file.

The TYPE parameter gives a description of the file. For example, if a file is a Home Computer BASIC program, the TYPE parameter shows PGM. If a file contains data in DISPLAY format with FIXED length records, the TYPE parameter shows D/F. If a file contains data in INTERNAL format with VARIABLE length records, the TYPE parameter shows I/V. (These characteristics are explained in detail in the section of this manual entitled "Home Computer File-Access Statements.")

The LRL (Logical Record Length) parameter gives the length (in bytes) of each record in the file if the file has FIXED-length records, or the maximum length (in bytes) of each record if the file has VARIABLE-length records.

The P column indicates whether the file is protected. If a Y is shown, the file is protected; if no Y is shown, the file is unprotected.

Backing Up a Diskette

The BACKUP DISK option enables you to duplicate a diskette. You should back up your diskettes regularly so that you have a spare copy of data and programs in case anything happens to the originals. As part of the backup process, the computer also compresses data and program files so that they use less storage space.

Note

You must select single-disk processing to back up a diskette on a single-drive system. If you do not select single-disk processing, the back-up command will not permit you to enter the same number for SOURCE and DESTINATION drive.

The instructions that follow describe the back-up process for systems with two or more disk drives. For instructions on backing up a diskette on a single disk drive, refer to the section entitled "Single-Disk Processing."

1926L HEXBUS Disk Drive/Controller Part 1 $:0:=Bullet \mu=Question Mark Alpha=Number Sign$

When you select the BACKUP DISK option, the following display appears:

BACKUP DISK SELECTIVE (Y/N)u N

Your response to the prompt <u>SELECTIVE $(Y/N)_{\mu}$ </u> determines whether all the files on a diskette or only selected files are to be copied. Press <u>N</u> to backup all files on the diskette. Press Y to select which files you backup.

The prompt SOURCE DISK $(1-4)\mu$ appears next. Enter the number of the disk drive containing the diskette you are backing up. The computer displays the name of the diskette so that you can be certain that you have the correct diskette.

The prompt DESTINATION DISK $(1-4)_{\mu}$ appears next. Your answer determines to which disk drive the files are copied. If the destination diskette is not initialized, the message DISK NOT INITIALIZED appears, followed by the prompt INITIALIZE NEW DISK $(Y/N)_{\mu}$. Information cannot be stored on a diskette unless it is initialized, so you should press \underline{Y} if you want to continue with the backup procedure (refer to "Initializing a Diskette" for details on how to answer the initialization prompts).

If you are copying the entire diskette, the message

WORKING.... PLEASE WAIT. PRESS:
CLEAR TO CANCEL COMMAND

appears, and information about the copying process appears on the screen as files are copied.

Press <u>CLEAR</u> if you want to stop the process. The Disk Manager 3 cartridge will stop after copying the current file.

1926L HEXBUS Disk Drive/Controller Part 1 to != Bullet μ =Question Mark Alpha=Number Sign

If you are copying only selected files, the filenames are displayed one at a time, followed by the prompt <u>COPY FILE $(Y/N)\mu$ </u> You can choose to copy the file by pressing Y, or to not copy it by pressing N.

Renaming a Diskette

The MODIFY DISK NAME option enables you to change the name of a diskette. When you choose this option, the following display appears:

MODIFY DISK NAME DISK DRIVE (1-4)µ 1

Answer the DISK DRIVE $(1-4)_{\mu}$ prompt with the number of the disk drive containing the diskette that you want to rename. The diskette's current name is displayed, followed by the prompt NEW DISKNAME $_{\mu}$. Enter the new name that you want to assign to the diskette.

Initializing a Diskette

The INITIALIZE NEW DISK option enables you to initialize (format) a diskette. Every new diskette must be initialized before it can be used by the computer. The initialization process places index information on the diskette that the computer uses to locate program and data files.

When you choose this option, the following display appears:

INITIALIZE NEW DISK
DISK DRIVE (1-4)µ 1

Answer the DISK DRIVE $(1-4)_{\mu}$ prompt with the number of the disk drive containing the diskette that you want to initialize. The message DISK NOT INITIALIZED appears, followed by the prompt NEW DISKNAME $_{\mu}$ Enter the name you want to give the diskette.

1926L HEXBUS Disk Drive/Controller Part 1 to:=Bullet µ=Question Mark Alpha=Number Sign

Note: If the diskette is already initialized, the diskette's name is shown instead of the message DISK NOT INITIALIZED. If you choose to reinitialize the diskette, all information stored on the diskette is erased.

The prompt $\overline{\text{TRACKS PER SIDE}_{\mu}}$ appears next. Press $\overline{\text{ENTER}}$ to accept the default value of 41 tracks.

The prompt $\underline{DOUBLE\ SIDED\ (Y/N)_u}$ appears next. Your response to this prompt depends upon the type of diskette you are using. If your diskette supports double-sided operation, select the double-sided option by pressing \underline{ENTER} . (A diskette that can be initialized for double-sided operation will usually have the words "double-sided" printed on the label of the diskette.) If you are using a single-sided diskette, press \underline{N} for single-sided initialization.

The prompt <u>DOUBLE DENSITY $(Y/N)_{\mu}$ </u> appears next. Press <u>ENTER</u> if your diskette supports double-density operation. (A diskette that can be initialized for double-density operation will usually have the words "double-density" printed on the label of the diskette.) If you are using a single-density diskette, press <u>N</u> for single-density initialization.

After you respond to the <u>DOUBLE DENSITY</u> prompt, the computer initializes and verifies the diskette. This process takes about seven minutes for a 41-track, double-sided, double-density diskette. While the computer initializes the diskette, the following message appears on the screen:

INITIALIZE NEW DISK WORKING.... PLEASE WAIT.

The computer then displays the number of each sector on the diskette as it verifies that the sector is usable. This verification process is an important check to ensure that information can be reliably stored on and retrieved from the diskette.

1926L HEXBUS Disk Drive/Controller Part 1 to!=Bullet μ =Question Mark Alpha=Number Sign

After the verification procedure is complete, the following message appears:

DSKn - DISKNAME= diskname AVAILABLE=nnnn USED=nnnn

The first line gives the number of the disk drive and the name of the diskette. The number following the prompt <u>AVAILABLE</u> indicates the number of sectors available for data storage, and the number following the prompt <u>USED</u> indicates the number of unusable sectors. The computer automatically prevents information from being stored in sectors that are found to be unusable.

Unusable sectors are usually caused by flaws in a diskette. If the diskette you are initializing shows a large number of used sectors, then you selected the wrong format for your diskette or the diskette is badly flawed. If the problem is a badly flawed diskette, you should consider not using the diskette because it may prove unreliable.

The numbers actually shown on your screen depend upon the initialization options you selected. Refer to Appendix E for a chart of expected values for various combinations of the options.

After the diskette has been initialized and verified, the following message appears:

COMMAND COMPLETED
PRESS: PROC'D, REDO,
BEGIN, OR BACK

Press $\underline{PROC'D}$ or \underline{REDO} to initialize another diskette, press \underline{BACK} to return to the DISK COMMANDS selection list, or press \underline{BEGIN} to return to the initial DISK MANAGER selection list.

1926L HEXBUS Disk Drive/Controller Part 1 to!=Bullet µ=Question Mark Alpha=Number Sign

DISK TESTS

The third option on the DISK MANAGER selection list is DISK TESTS. This option enables you to check the condition of a diskette, disk controller, and disk drive. Two tests are available. The first test checks only the reliability of a diskette. The second test performs an exhaustive check of the diskette, disk controller, and disk drive.

When you choose the DISK TESTS option, the following selection list appears:

DISK TESTS

1 QUICK TEST 2 COMPREHENSIVE TEST

YOUR CHOICE 1

Quick Test

The QUICK TEST option enables you to perform a quick test on a diskette that you suspect is not storing data reliably. When you choose this option, the following display appears:

QUICK TEST
DESTRUCTIVE TEST (Y/N)_µ N

The prompt <u>DESTRUCTIVE TEST $(Y/N)_{\mu}$ </u> enables you to decide whether information on the diskette is preserved or lost. If you do not care whether the information on the diskette is lost, press <u>Y</u>. To preserve the information on the diskette, press N.

If you select the destructive test, the computer initializes the diskette before beginning the test. Respond to the prompts <u>TRACKS PER SIDE</u>, <u>DOUBLE SIDED</u>, and <u>DOUBLE DENSITY</u> that are displayed next. (For details on how to answer these questions, refer to "Initializing a Diskette.")

The prompt NUMBER OF LOOPS $(1-9)_{\mu}$ appears next. Select the number of times that you want the test performed by pressing the desired number key.

The prompt LOG ERRORS $(Y/N)_{\mu}$ appears next. To display the error report on the screen only, press N. If you want errors to be recorded on a device in addition to the screen, press Y. The computer will display the prompt DEVICE NAME $_{\mu}$ and permit you to enter a device name. For example, to record any errors in a file named LOGLIST on the diskette in drive 2, enter DSK2.LOGLIST (you cannot log errors to the same disk drive that you are testing).

After you respond to the final prompt, the computer performs the test. You can cancel the test at any time by pressing <u>CLEAR</u>. After the test is run, the computer displays a report in the following general form:

BAD ADDRESS/CODE

nnnn/nn

TOTAL ERRORS:

nn

COMPLETED PASS:

nn

If any errors are found, the sector in which the error was found and the number of the error are displayed following the message <u>BAD ADDRESS/CODE</u>. If no errors are found, the <u>BAD ADDRESS/CODE</u> message is omitted and the number following the message <u>TOTAL ERRORS</u> will be **6**. The number following the message COMPLETED PASS indicates the number of times the test was performed.

If no errors are found, this test takes about 7 minutes for a 40-track, double-sided, double-density diskette.

1926L HEXBUS Disk Drive/Controller Part 1 !o!=Bullet µ=Question Mark Alpha=Number Sign

Comprehensive Test

CAUTION

The comprehensive test erases all information on a diskette. The diskette must be reinitialized before it can be reused.

If you choose to run the comprehensive test, the following display appears:

COMPREHENSIVE TEST

DESTRUCTIVE

TRACKS PER SIDE 40

The comprehesive test begins by initializing the diskette. Press ENTER to accept 40 tracks. Then respond to the DOUBLE SIDED $(Y/N)_{\mu}$ and DOUBLE DENSITY $(Y/N)_{\mu}$ prompts that appear next as appropriate for the diskette that you want to test. (For details on how to answer these questions, refer to "Initializing a Diskette.") When the prompt DISK DRIVE $(1-4)_{\mu}$ appears, enter the number of the disk drive to be tested.

From this point on, the procedure is the same as for the QUICK TEST. Respond to the prompts when they appear on the screen.

The comprehensive test consists of six tests that write to and read from each sector on the diskette. If no errors are found, this test takes about 59 minutes for a 40-track, double-sided, double-density diskette.

1926L HEXBUS Disk Drive/Controller Part 1 to:=Bullet μ =Question Mark Alpha=Number Sign

SINGLE-DISK PROCESSING

Option 4 on the DISK MANAGER selection list enables you to set the cartridge for single-disk processing. You must select this option to back up a diskette on a one-drive system. For other commands, selecting this option speeds up the Disk Manager functions. After you choose this option, you can perform multiple-disk operations with the Disk Manager 3 cartridge only by returning to the computer's master selection list and reselecting the Disk Manager 3 cartridge.

When using single-disk processing, be sure to insert a diskette into the disk drive before selecting other commands. The prompts <u>DISK DRIVE (1-4)</u> μ , <u>SOURCE DISK (1-4)</u> μ , and <u>DESTINATION DISK (1-4)</u> μ are answered automatically each time you choose an option, and the diskette is accessed immediately.

1927L HEX-BUS Disk Drive/Controller Part 2 !o!=Bullet µ=Question mark Alpha=Number sign

Initializing A Diskette From BASIC

Every new diskette must be initialized (formatted) before the computer can use it. The initialization process places index information on the diskette that the computer uses to locate program and data files. This section describes the commands available in Home Computer and Compact Computer BASIC for initializing a diskette. <u>Initializing a diskette erases any information previously stored on the diskette</u>.

THE HEXBUS FORMAT MEDIA SUBCOMMAND

The <u>HEX-BUS</u> Interface peripheral provides several special subcommands that can be executed by means of the OPEN statement. The FORMAT MEDIA subcommand enables Home Computer owners who have the HEX-BUS Interface peripheral connected to their computer to initialize and verify a diskette for 40-track, double-sided, double-density operation. For details regarding other HEX-BUS subcommands, refer to the owner's manual provided with the HEX-BUS Interface.

The FORMAT MEDIA subcommand has the following general form:

OPEN afile-number: "HEXBUS.FORMAT MEDIA.device"

The <u>file-number</u> must not be assigned to another open file. The words HEXBUS.FORMAT MEDIA must appear in quotes in the OPEN statement, with a period separating HEXBUS from FORMAT MEDIA. The <u>device</u> name must be separated from the preceding words by a period, as shown. The <u>device</u> must be a number from lol through lo4, where the last digit of the number identifies the disk drive containing the diskette to be initialized. For example,

OPEN al: "HEXBUS.FORMAT MEDIA. 101"

instructs the computer to initialize the diskette in disk drive 1.

After the diskette has been initialized, execute a CLOSE statement. If you do not CLOSE the <u>file-number</u> assigned to the FORMAT MEDIA subcommand, that number

1927L HEX-BUS Disk Drive/Controller Part 2 to:=Bullet μ =Question mark Alpha=Number sign

cannot be used in another OPEN statement.

Note: Executing the FORMAT MEDIA subcommand with any diskette files open will cause an error condition.

The FORMAT MEDIA subcommand assigns the diskette the default name TI-DISK.

The FORMAT MEDIA subcommand instructs the computer to initialize and verify a diskette for 40-track, double-sided, double-density operation. It does not permit you to select other storage formats, nor does it warn you that a diskette already contains data. As soon as the subcommand is executed, the computer begins initializing the diskette. The procedure requires about three minutes to complete.

The FORMAT MEDIA subcommand enables Home Computer owners to initialize a diskette without inserting the Disk Manager 3 cartridge into the computer console. Use the Disk Manager 3 cartridge when you want to initialize a diskette for a different storage format, or when you want a report on how many sectors are used (flawed) and available after initialization.

THE COMPACT COMPUTER FORMAT COMMAND

The FORMAT command enables Compact Computer owners to initialize and verify a diskette for 40-track, double-sided, double-density operation.

The FORMAT command has the following general form:

FORMAT device

The device must be a number from 10 through 104, where the last digit of the number identifies the disk drive (1-4) containing the diskette that you want to initialize. For example,

FORMAT 101

instructs the computer to initialize the diskette in disk drive 1.

1927L HEX-BUS Disk Drive/Controller Part 2 to!=Bullet u=Question mark Alpha=Number sign

Note: Executing the FORMAT command with any diskette files open will cause an error condition.

The FORMAT command assigns the diskette the default name TI-DISK.

The FORMAT command instructs the computer to initialize a diskette for 40-track, double-sided, double-density operation. The command does not permit you to select other storage formats, nor does it warn you when a diskette already contains data. As soon as the command is executed, the computer begins initializing the diskette. The procedure requires about three minutes to complete.

The Disk Utilities package (sold separately by Texas Instruments) enables Compact Computer owners to initialize diskettes in other storage formats.

Storing Programs on Diskettes

This section describes the commands used to save, protect, verify, load, and delete programs. It also describes the use of RUN as a program statement and the procedure to follow to merge programs. Except where noted, all sections apply to the 99/4A Home Computer and Compact Computer 40.

PROGRAM-NAMING CONVENTIONS

A program name can be any combination of letters, numbers, and punctuation signs up to ten characters in length, so long as the name does not contain any spaces or periods. It is valid to use lower-case letters in Home Computer program names. If lower-case letters are used in Compact Computer program names, the computer converts them to upper-case letters.

THE SAVE COMMAND

The SAVE command stores a copy of a program on a diskette. The SAVE command does not alter or erase the program in memory.

The SAVE command has the following general form:

SAVE "device.program-name"

The quotation marks around the <u>device.program-name</u> parameter are optional for the Home Computer, but are required for the Compact Computer. The <u>device</u> name must be separated from the program-name by a period.

For a Home Computer, the device can be specified in four ways:

!o! By using the letters DSK and the number of the disk drive (1-4) on which you want the program to be saved. For example, DSK1 specifies disk drive 1 and DSK4 specifies disk drive 4.

- !o! By using the letters DSK, a period, and the name of a diskette. For example, DSK.TI-DISK specifies the disk drive containing the diskette named TI-DISK.
- !o! By using the word HEXBUS, a period, and a number from 101 through 1024, where the last digit of the number specifies the disk drive on which you want the program to be saved. For example, HEXBUS.101 specifies disk drive 1 and HEXBUS.102 specifies disk drive 2.
- !o! By using the word HEXBUS, a period, the number 100, a period, and the name of a diskette. For example, HEXBUS.101.TEXTFILE specifies the disk drive containing the diskette named TEXTFILE.

For the Compact Computer, the device can be specified in two ways:

- io! By using a number from 101 through 104, where the last digit of the number identifies the disk drive (1-4) on which you want the program to be saved. For example, 102 specifies disk drive 2 and 103 specifies disk drive 3.
- :o! By using the number 100, a period, and the name of a diskette. For example, 100.TI-DISK specifies the disk drive containing the diskette named TI-DISK.

Examples—Home Computer

SAVE "DSK1.TEST"

Saves a program named TEST on the diskette in drive 1.

SAVE DSK.PAY7.PAYROLL
SAVE HEXBUS.100.PAY7.PAYROLL

Both examples save a program named PAYROLL on the diskette named PAY7.

Examples--Compact Computer

SAVE "101.TEST"

Saves a program named TEST on the diskette in drive 1.

SAVE "101.PAY7.PAYROLL"

Saves a program named PAYROLL on the diskette named PAY7.

1927L HEX-BUS Disk Drive/Controller Part 2 !o!=Bullet µ=Question mark Alpha=Number sign

SAVING PROTECTED PROGRAMS

You can prevent the examination or editing of programs by using the PROTECTED option with the SAVE command. (Home Computer owners must have the Extended BASIC cartridge installed in the computer to use the PROTECTED option.)

To protect a program, use the following format:

SAVE "device.program_name", PROTECTED

The quotation marks around the <u>device.program-name</u> specification are optional for a Home Computer, but are required for the Compact Computer.

CAUTION

The PROTECTED option is not reversible. Therefore, maintain an unprotected copy of any protected program.

A program saved with the PROTECTED option can be loaded into memory and executed, but it cannot be listed, edited, or saved. A program saved with the PROTECTED option can be copied or backed up with the Disk Manager 3 cartridge.

Programs saved with the PROTECTED option are not shown as protected by catalog programs or by the Disk Manager 3 catalog option.

Examples--Home Computer

SAVE DSK2.PRIVATE, PROTECTED

Saves a program named PRIVATE on the diskette in drive 2.

SAVE HEXBUS. 100. PERSONAL. PRIVATE, PROTECTED

Saves a program named PRIVATE on the diskette named PERSONAL.

1927L HEX-BUS Disk Drive/Controller Part 2 !o!=Bullet $\mu=Question$ mark Alpha=Number sign

Example—Compact Computer

SAVE "102.PRIVATE", PROTECTED

Saves a program named PRIVATE on the diskette in drive 2.

VERIFYING A PROGRAM

The Disk Drive/Controller provides an extremely reliable method of storing information. Occasionally, however, information may not be transmitted perfectly between your computer and the Disk Drive/Controller. The cause may be a worn or scratched diskette, a damaged HEX-BUSTM cable, or a malfunction in the computer or Disk Drive/Controller. The verify command enables you to check that a program has been correctly saved on a diskette.

For a Home Computer, the VErify subcommand has the following form:

SAVE HEXBUS. VE. device. filename

Note: The SAVE HEXBUS.VE.<u>device.filename</u> subcommand does not save a program. The program must be saved before the VErify subcommand is executed.

For the Compact Computer, the VERIFY command has the following form:

VERIFY "device.filename" [,PROTECTED]

For all computers, the device can be specified in two ways:

the number of the disk drive. (Note that Home Computer owners cannot use the device name DSK with the VErify command.)

10! By using the number 100, a period, and the name of a diskette.

The PROTECTED option is required when VERIFYing a Compact Computer program saved in PROTECTED format. The VErify command cannot be used to verify a PROTECTED Home Computer program.

The verify command compares the information stored on the diskette with the information in the computer's memory. If any discrepancy is found, an error message is displayed.

1927L HEX-BUS Disk Drive/Controller Part 2 to:=Bullet u=Question mark Alpha=Number sign

Example—Home Computer

SAVE DSK1.TEST
SAVE HEXBUS.VE.101.TEST

The first command saves a program named TEST to the diskette in drive 1. The second command verifies that the program named TEST on the diskette in drive 1 is identical to the program in memory.

Example—Compact Computer

SAVE "101.TEST", PROTECTED VERIFY "101.TEST", PROTECTED

The first command saves a program named TEST to the diskette in drive 1. The program is saved in protected format. The second command verifies that the (protected) program named TEST on the diskette in drive 1 is identical to the program in memory.

THE OLD COMMAND

The OLD command is used to load a program from a diskette into the computer's memory. Any program previously stored in the computer's memory is lost when the program is loaded.

The OLD command has the following general form:

OLD "device.program-name"

The quotation marks around the <u>device.program-name</u> specification are optional for a Home Computer, but are required for the Compact Computer.

As with the SAVE command, the <u>device</u> is specified differently depending upon whether you have a Home Computer, or a Compact Computer. Refer to the discussion of the SAVE command for a description of the different methods of specifying the device name.

Examples—Home Computer

OLD DSK1.TEST
OLD HEXBUS.101.TEST

Both examples load a program named TEST from the diskette in drive 1.

Example—Compact Computer

OLD "101.TEST"

Loads a program named TEST from the diskette in drive 1.

1927L HEX-BUS Disk Drive/Controller Part 2 to!=Bullet μ =Question mark Alpha=Number sign

DELETING A PROGRAM

The DELETE command is used to erase a program from a diskette. Once a program is deleted, it cannot be recovered.

The DELETE command has the following general form:

DELETE "device.program-name"

The quotation marks around the <u>device.program-name</u> specification are required for all three computers.

As with the SAVE command, the <u>device</u> is specified differently depending upon whether you have a Home Computer or a Compact Computer. Refer to the discussion of the SAVE command for a description of the different methods of specifying the device name.

Example—Home Computer

DELETE "DSK1.TEST"

Erases the program named TEST from the diskette in drive 1.

Example—Compact Computer

DELETE "101.TEST"

Erases the program named TEST from the diskette in drive 1.

RUN AS A PROGRAM STATEMENT

Home Computer Extended BASIC and Compact Computer BASIC enable you to use RUN as a program statement to load and execute program. The current program and its associated variables are erased before the new program is loaded. (Home Computer owners must have the Extended BASIC cartridge installed in the computer to use RUN as a program statement.)

When used as a program statement, RUN has the following general form:

RUN "device.program-name"

The quotation marks around the <u>device.program-name</u> specification are required for all three computers.

As with the SAVE command, the <u>device</u> is specified differently depending upon whether you have a Home Computer, or a Compact Computer. Refer to the discussion of the SAVE command for a description of the different methods of specifying the device name.

Examples--Home Computer

RUN "HEXBUS. 104. PROGRAM1"

Loads and runs the program named PROGRAM1 from the diskette in drive 4.

A8="DSK1.CALC"

RUN AS

Loads and runs the program named CALC from the diskette in drive 1.

Examples—Compact Computer

RUN "104.PROGRAM1"

Loads and runs the program named PROGRAM1 from the diskette in drive 4.

1927L HEX-BUS Disk Drive/Controller Part 2 to:=Bullet μ =Question mark Alpha=Number sign

A8="101.CALC"

RUN AS

Loads and runs the program named CALC from the diskette in drive 1.

1927L HEX-BUS Disk Drive/Controller Part 2 $0!=Bullet \mu=Question mark Alpha=Number sign$

SAVING PROGRAMS IN MERGE FORMAT

The Extended BASIC cartridge available for the Home Computer enables you save programs in a special "character for character" format by adding the MERGE option to the SAVE command. (In MERGE format, BASIC keywords such as DISPLAY require 7 bytes of storage rather than the usual 1 byte.) Although programs saved in MERGE format require more diskette storage space than programs saved without the MERGE option, they have the following advantages:

- :o: A program saved in MERGE format can be combined with another program already stored in the computer's memory. (The Extended BASIC command MERGE is used to combine the programs.)
- io: Programs stored in MERGE format can be opened as data files and read into the computer's memory for manipulation and editing. (Open the file as a SEQUENTIAL file in DISPLAY format with VARIABLE length records, and use the LINPUT a statement to read the file.)
- !o! Programs stored in MERGE format can be loaded into the TI-WRITERTM word processor for easy editing.
- !o! Programs stored in MERGE format can be transmitted over a telephone line to another computer. (The proper telecommunications equipment and software is required to send a file over a telephone line.)

The command to save a program in MERGE format has the following general form:

SAVE "device.program-name", MERGE

The quotation marks around the device.program-name specification are optional.

Refer to the discussion of the SAVE command for a description of the different methods of specifying the device name.

1927L HEX-BUS Disk Drive/Controller Part 2 !o!=Bullet u=Question mark Alpha=Number sign

The following example demonstrates the process of "merging" two programs. Enter the following short program.

100 REM PROGRAM 1-LINE 100

110 REM PROGRAM 1-LINE 11D

140 REM PROGRAM 1-LINE 140

150 REM PROGRAM 1-LINE 150

Now save the program to diskette using the MERGE option, as shown below.

SAVE DSK1.PROGRAM1, MERGE

Next type $\underline{\text{NEW}}$ and press $\underline{\text{ENTER}}$ to clear the computer's memory. Then enter the following program. Notice that the first and second programs overlap at lines 110 and 140.

110 REM PROGRAM 2--LINE 110

120 REM PROGRAM 2-LINE 120

130 REM PROGRAM 2--LINE 130

.140 REM PROGRAM 2-LINE 140

To merge the stored program with the program in memory, enter

MERGE DSK1.PROGRAMI

When the flashing cursor reappears, <u>LIST</u> the program. The program should appear as shown below.

100 REM PROGRAM 1-LINE 100

110 REM PROGRAM 1-LINE 110

120 REM PROGRAM 2-LINE 120

13D REM PROGRAM 2-LINE 13D

140 REM PROGRAM 1-LINE 140

150 REM PROGRAM 1-LINE 150

1927L HEX-BUS Disk Drive/Controller Part 2 to:=Bullet μ =Question mark Alpha=Number sign

The program on the diskette has been merged with the program in memory. Where lines of the two programs overlapped, the lines from the program on the diskette have replaced the lines in memory. The lines that did not overlap have been inserted in the proper numeric order.

Home Computer File-Access Statements

This section defines the statements used to access diskette data files by the 99/4A Home Computer. If you have the Disk Drive/Controller connected to a Compact Computer, refer to the section entitled "Compact Computer File-Access Statements."

The BASIC statements used to access data files by the Home Computer are listed pelow.

OPEN Prepares the computer to create or to use a data file by defining the file's characteristics.

PRINT a Writes information to a data file.

INPUT α Reads information from a data file.

LINPUT α Reads an entire record from a data file (available in Extended BASIC only).

EOF Provides a program with a means of testing whether the end of a sequential file has been reached.

RESTORE Provides a program with a means of specifying the record to be read from or written to next.

REC Provides a program with a means of checking the number of the next record to be read from or written to (available in Extended BASIC only).

CLOSE Terminates program access to a file.

DELETE Deletes a file from a diskette.

THE OPEN STATEMENT

The OPEN statement prepares the computer to access a data file by defining the characteristics of the file. The OPEN statement is used both to create a new file and to prepare the computer to read from or write to a file that already exists. A new file is automatically created when the computer does not find an existing file matching the parameters of the OPEN statement.

The Disk Drive/Controller permits a maximum of four files to be open at one time, regardless of the number of disk drives being used.

The OPEN statement has the following general form:

OPEN afile-number: "device.filename" [,file-organization] [,file-type] [,open-mode] [,record-type]

The <u>file-number</u> and <u>device.filename</u> parameters must be provided in every OPEN statement, but the other parameters can be included in any order, or omitted. A default characteristic is assumed for any parameter that is omitted.

!o! <u>file-number</u>—Every data file must be assigned an identification number from 1 through 255 when it is opened. This number provides the BASIC statements that read from and write to the file with an abbreviated method of referring to the file. Any numeric constant, expression, or variable having a value within the range 1 through 255 is a valid <u>file-number</u>, so long as that <u>file-number</u> is not assigned to another open file.

The <u>file-number</u> parameter must be preceded by a number symbol (α) and followed by a colon.

!o! device.filename—The device.filename parameter identifies the peripheral containing the file and the name of the file. If the device.filename specification is a string constant, it must be enclosed in quotation marks.

The device name can be specified in four ways:

- !o! By using the letters DSK and the number of the disk drive (1-4) on which you want the program to be saved. For example, DSK1 specifies disk drive 1 and DSK4 specifies disk drive 4.
- !o! By using the letters DSK, a period, and the name of a diskette. For example, DSK.TI-DISK specifies the disk drive containing the diskette named TI-DISK.
- !o! By using the letters HEXBUS, a period, and a number from lol through lo4, where the last digit of the number specifies the disk drive (1-4) on which you want the program to be saved. For example, HEXBUS.101 specifies disk drive l and HEXBUS.102 specifies disk drive 2.
- the name of a diskette. For example, HEXBUS. 100. TEXTFILE specifies the disk drive containing the diskette named TEXTFILE.

The <u>filename</u> can be any combination of letters, numbers, and punctuation signs up to locharacters in length, so long as the name does not contain any spaces or periods. It is valid to use lower-case letters in a filename.

The <u>device.filename</u> parameter must have a period separating the <u>device</u> from the <u>filename</u>. If the <u>device.filename</u> specification is a string constant, it must be enclosed in quotation marks. If the parameter is specified by a variable, it must be a string variable.

!o! file-organization—The file-organization parameter indicates whether the file is opened for SEQUENTIAL access or RELATIVE access. A sequential data file must be read or written in sequence, beginning with the first piece of information in the file. Relative data files can be read or written in any order, including sequentially. To specify sequential file access, include the word SEQUENTIAL in the OPEN statement. To specify relative file access, include the word RELATIVE in the OPEN statement.

If the <u>file-organization</u> parameter is omitted, the computer assumes the file is being opened for SEQUENTIAL access.

!o! <u>file-type-</u>The <u>file-type</u> parameter specifies whether the information in the file is stored in DISPLAY format or INTERNAL format. If DISPLAY format is specified, data values are stored as ASCII characters. If INTERNAL format is specified, data values are stored in the binary format used to represent information in the computer's memory.

As a rule, it is preferable to use INTERNAL format for storing data in diskette files. Because no format conversions are required, data values in INTERNAL format are generally processed faster than data values in DISPLAY format. The DISPLAY option is intended primarily for files opened on devices other than disk drives.

If the <u>file-type</u> parameter is omitted, the computer assumes DISPLAY format.

!o! open-mode—The open-mode parameter specifies whether the file can be both read from and written to (UPDATE), only read from (INPUT), only written to (OUTPUT), or only added to (APPEND).

<u>Note:</u> If a file already exists on a diskette, opening the file for OUTPUT completely erases the previous contents of the file.

If the open-mode parameter is omitted, the computer assumes UPDATE mode.

to: record-type—The record-type parameter specifies whether the individual records in the file are of the same length (FIXED) or vary in length (VARIABLE). If FIXED-length records are selected, the FIXED specification can be followed by a numeric-expression giving the record length for the file. If VARIABLE-length records are selected, the VARIABLE specification can be followed by a numeric-expression giving the maximum record length for the file.

SEQUENTIAL files can use either FIXED— or VARIABLE-length records. RELATIVE files must use FIXED—length records.

The maximum record length for a SEQUENTIAL file is 254 bytes. The maximum record length for a RELATIVE file is 255 bytes.

If the <u>record-type</u> parameter is omitted for an existing file, the computer uses the record length established when the file was created.

If the <u>record-type</u> parameter is omitted when a new SEQUENTIAL file is opened, VARIABLE records with a maximum length of 81 bytes are assumed. If the <u>record-type</u> parameter is omitted when a new RELATIVE file is opened, FIXED records with a length of 255 bytes are assumed.

Examples:

100 OPEN a200: "DSK2.PAYROLL", INPUT, INTERNAL

Opens a file named PAYROLL on the diskette in drive 2. The <u>file-number</u> is 200, the <u>open-mode</u> is INPUT, and the <u>file-type</u> is INTERNAL. Because no <u>file-organization</u> or <u>record-type</u> is specified, the file is assumed to be a SEQUENTIAL file with VARIABLE-length records. The record length is set to the value established when the file was created.

OPEN al: "HEXBUS.111.TEMPFILE", RELATIVE, INTERNAL, OUTPUT, FIXED 64

Opens a file named TEMPFILE on the diskette in drive 1. The <u>file-number</u> is 1, the <u>file-organization</u> is RELATIVE, the <u>file-type</u> is INTERNAL, the <u>open-mode</u> is OUTPUT, and the record-type is FIXED with a record length of 64 bytes.

100 A=3

110 A8="DSK.DATADISK.GL DATA"

120 OPEN aA:AS, RELATIVE

Opens a file named GL_DATA on the diskette named DATADISK. The <u>file-number</u> is 3 and the <u>file-organization</u> is RELATIVE. Because the <u>file-type</u> and <u>open-mode</u> parameters are omitted, the file is in DISPLAY format and is opened for UPDATE. The records are 255 bytes long if the file is new. If the file already exists, the <u>record-type</u> is set to the value established when the file was created.

The OPEN statement produces an error condition if:

- :o: The characteristics of an existing file and those of the OPEN statement do not match.
- io: The device specifed by the OPEN statement is not connected to the computer.
- io! An attempt is made to OPEN more than four diskette files at one time.
- !o! An attempt is made to OPEN a file that is already OPEN.
- io! An attempt is made to OPEN a new file on a diskette that is full.
- !o! An attempt is made to OPEN a file on a diskette that is not initialized.
- !o! An attempt is made to OPEN a file with invalid or contradictory parameters.

1927L HEX-BUS Disk Drive/Controller Part 2 to:=Bullet μ =Question mark Alpha=Number sign

THE PRINT & STATEMENT

The PRINT α statement writes information to a data file that has been opened in UPDATE, OUTPUT, or APPEND mode. Writing to a file opened in INPUT mode causes an error condition.

The PRINT α statement has the following general form:

PRINT afile-number [, REC record-number][:print-list]

The <u>file-number</u> parameter is required and must match the number assigned to the file by the OPEN statement. If the file was opened for RELATIVE access, the PRINT α statement can optionally specify the number of the record to write with the REC <u>record-number</u> parameter. Using a REC <u>record-number</u> parameter with a file opened for SEQUENTIAL access causes an error condition.

When the computer performs the PRINT α statement, the <u>print-list</u> data is stored in a temporary memory area called an input/output (I/O) buffer. A separate buffer is established for each file that is opened. If the <u>print-list</u> does not end with a comma or semicolon, the information in the I/O buffer is immediately written to the file. If the <u>print-list</u> does end with a comma or semicolon, a "pending" print condition occurs.

The pending print condition instructs the computer to wait until the next PRINT α statement is performed before writing the contents of the I/O buffer to the file. When the next PRINT α statement is performed, one of the following actions occurs:

io! If the next PRINT α statement has no REC parameter, the computer adds the <u>print-list</u> information of the second PRINT α statement to the information already stored in the I/O buffer. Then, unless the second PRINT α statement also ends in a comma or a semicolon, the computer writes the contents of the I/O buffer to the data file.

io: If the next PRINT α statement has a REC parameter, the computer writes the contents of the I/O buffer to the file before performing the action requested by the second PRINT α statement.

Examples:

100 PRINT al:X

Writes the value of X into the next record of the file opened as α l.

100 PRINT a23: N; NAME 8; PAYRATE

Writes the values of N, NAME\$, and PAYRATE into the next record of the file opened as $\alpha 23$.

100 PRINT a7, REC 44: "05/11/83"

Writes the string constant 05/11/83 into record 44 of the file opened as $\alpha 7$. The file must be a RELATIVE file.

100 PRINT α3:A,B,C, 110 PRINT α3:D,E

The comma after the C in the first PRINT α statement creates a pending print condition. Therefore, the computer places the values of A, B, and C into the I/O buffer of the file opened as $\alpha 3$ and waits until the next PRINT α statement to determine when to write the information to the file. When the second PRINT α statement is executed, the values of D and E are added to the values already in the I/O buffer, and the contents of the buffer are written to the file.

THE INPUT a STATEMENT

The INPUT α statement reads information from a data file that has been opened in UPDATE or INPUT mode. Reading from a file opened in OUTPUT or APPEND mode causes an error condition.

The INPUT a statement has the following general form:

INPUT afile_number [,REC record_number][:variable_list]

The <u>file-number</u> parameter is required and must match the number assigned to the file by the OPEN statement. If the file was opened for RELATIVE access, the INPUT α statement can optionally specify the number of the record to read with the <u>REC record-number</u> parameter. Using a <u>REC record-number</u> parameter with a file opened for SEQUENTIAL access causes an error condition.

When the computer performs an INPUT α statement, the next entire record is read into the I/O buffer assigned to the file. The data values from the I/O buffer are then assigned to the variables in the <u>variable-list</u> in left-to-right order. After the variables have been assigned values, any data values remaining in the I/O buffer are discarded unless the <u>variable-list</u> ends with a comma. If the <u>variable-list</u> ends with a comma, a "pending" input condition is established.

The pending input condition instructs the computer to wait for the next INPUT α statement before deciding what to do with the unassigned data values. When the next INPUT α statement is executed, one of the following actions occurs:

- in the I/O buffer are assigned to the variables in the variable-list of the second INPUT α statement.
- :o: If the next INPUT α statement has a REC parameter, the unused contents of the I/O buffer are discarded and the action specified by the second INPUT α statement is performed.

1927L HEX-BUS Disk Drive/Controller Part 2 to!=Bullet u=Question mark Alpha=Number sign

Examples:

100 INPUT al:X

Assigns the next data value from the file opened as αl to the variable X. If the next value is not a number, an error condition occurs.

100 INPUT a23:N, NAME &, PAYRATE

Assigns the next three data values from the file opened as $\alpha 23$ to the variables N, NAMES, and PAYRATE. If the values for N and PAYRATE are not numbers, an error condition occurs.

100 INPUT a7, REC 44: DATES

Assigns the first data value in record 44 of the file opened as $\alpha 7$ to the variable DATES. The file must be a RELATIVE file.

100 INPUT α 3:A,B,C,

The comma after the C creates a pending input condition. After assigning the next three data values from the file opened as $\alpha 3$ to the variables A, B, and C, the computer waits until the next INPUT α statement to decide whether to discard any data values remaining in the I/O buffer.

THE LINPUT & STATEMENT

The LINPUT α statement reads an entire record from a file containing data stored in DISPLAY format. If part of the record has been read already, the LINPUT statement reads only the remainder of the record. (The LINPUT α statement is available to Home Computer owners only if the Extended BASIC cartridge is installed in the computer.)

The LINPUT α statement has the following general form:

LINPUT afile-number [,REC record-number]:string-variable

The <u>file-number</u> is required and must match the number assigned to the file by the OPEN statement. If the file was opened for RELATIVE access, the LINPUT α statement can optionally specify the number of the record to read with the REC record-number parameter. Using the REC record-number parameter with a file opened for SEQUENTIAL access causes an error condition.

The information read from the file by the LINPUT α statement is assigned to the <u>string-variable</u>. The LINPUT α statement is not affected by the markers used to separate data values stored in the same record. Therefore, if a record contains two numeric values, a string value, and a final numeric value, all five values are assigned to the string-variable by the LINPUT α statement.

Using the LINPUT α statement to read a file stored in INTERNAL format causes an error condition. Using the LINPUT α statement to read a record longer than 255 bytes causes an error condition occurs.

Examples:

100 LINPUT a1:L8

Assigns the data in the next record of the file opened as αl to the variable L§.

1927L HEX-BUS Disk Drive/Controller Part 2 to:=Bullet μ =Question mark Alpha=Number sign

100 LINPUT a7, REC 44: TEST8

Assigns the data in record 44 of the file opened as $\alpha 7$ to the variable TEST8. The file must be a RELATIVE file.

1927L HEX-BUS Disk Drive/Controller Part 2 !o!=Bullet u=Question mark Alpha=Number sign

THE EOF FUNCTION

The EOF (End-of-File) function tests whether another record can be read from a SEQUENTIAL file. Using the EOF function with a RELATIVE file causes an error condition.

The EOF function has the following general form:

EOF(file-number)

The <u>file-number</u> is required and must match the number assigned to the file by the OPEN statement.

The EOF function returns the following values:

<u>Value</u>	<u>Meaning</u>
1	Logical End of File
D	Not End of File
_1	Physical End of File

A logical-end-of-file result indicates that all the records in the file have been read. A not-end-of-file result indicates that at least one record remains to be read from the file. A physical-end-of-file result indicates that all the records in the file have been read and there is no space on the diskette to increase the size of the file.

Example:

100 IF EOF(1)=0 THEN 300

Tests the end-of-file condition for the file opened as α l. If the last record of the file has not been read, the program branches to line 3

1927L HEX-BUS Disk Drive/Controller Part 2 to t=8u llet $\mu=Question$ mark Alpha=Number sign

THE RESTORE STATEMENT

The RESTORE statement enables a program to reposition itself to read or write from the beginning of a file, or for RELATIVE files, from a specific record in a file.

The RESTORE statement has the following general form:

RESTORE afile-number [, REC record-number]

The <u>file-number</u> parameter is required and must match the number assigned to the file by the OPEN statement. If the file was opened for RELATIVE access, the RESTORE statement can optionally specify the number of the record to read or write next with the REC <u>record-number</u> parameter. Using a REC <u>record-number</u> parameter with a file opened for SEQUENTIAL access causes an error condition.

When used without the $\underline{\sf REC}$ record-number parameter, the RESTORE statement repositions the program to the beginning of the file.

Examples:

100 RESTORE al

Repositions the computer's internal record counter to the beginning of the file opened as α l. The next INPUT α or PRINT α statement will read from or write to the first record in the file.

100 RESTORE a7, REC 44

Positions the computer's internal record counter to record number 44 of the file opened as $\alpha 7$. The next INPUT α or PRINT α statement will read from or write to record 44.

THE REC FUNCTION

The REC function provides a program with a means of checking the number of the record to be read from or written to next. The REC function can be used only with RELATIVE files; using the REC function with a SEQUENTIAL file causes an error condition. (The REC function is available to Home Computer owners only if the Extended BASIC cartridge is installed in the computer.)

The REC function has the following general form:

REC(file-number)

The <u>file-number</u> is required and must match the number assigned to the file by the OPEN statement.

The records in a data file are numbered starting with \boldsymbol{o} . Thus, for example, the fourth record of a file is record number 3.

Example:

100 R=REC(4)

Assigns the number of the next record of the file opened as $\alpha 4$ to the variable R.

THE CLOSE STATEMENT

The CLOSE statement instructs the computer to terminate access to a file. A closed file cannot be accessed again until another OPEN statement is performed.

The CLOSE statement has the following general form:

CLOSE afile-number [:DELETE]

The <u>file-number</u> is required and must match the number assigned to the file by the OPEN statement. The DELETE parameter is optional and enables a program to erase a file from a diskette when the file is closed. (<u>The DELETE parameter is available to Home Computer owners only if the Extended BASIC cartridge is installed in the computer.)</u>

It is vital to close an open data file. As part of the closing process, the computer writes any information remaining in the I/O buffer to the file. If a file is not closed, that information is lost. The computer closes all open files if any of the following actions occur:

- !o! A program is stopped with a STOP or END statement.
- io: A NEW or BYE command is executed.
- io: A program line is edited.

Note: If you leave BASIC by turning the computer off or by pressing $\overline{\text{QUIT}}$, the computer does not close any files that may be open.

Example:

100 OPEN $\alpha 7$: "DSK2.DISKFILE", INPUT, INTERNAL 110 CLOSE $\alpha 7$

Closes the file assigned file-number 7.

1927L HEX-BUS Disk Drive/Controller Part 2 $:o!=Bullet \mu=Question mark Alpha=Number sign$

THE DELETE STATEMENT

The DELETE statement erases a data file or program from a diskette. The statement has the following general form:

DELETE "device.filename"

The <u>device.filename</u> parameter identifies the disk drive containing the file, and the name of the file. Refer to the discussion of the OPEN statement for information about the device.filename parameter.

<u>Note:</u> A file protected by the Disk Manager cartridge cannot be deleted. You must remove the protected status using the Disk Manager before you can delete the file.

Examples:

DELETE "DSK2.PAYROLL"

Deletes the file named PAYROLL from the diskette in drive 2.

DELETE "HEXBUS. 101. TEMPFILE"

Deletes the file named TEMPFILE from the diskette in drive 1.

110 A8="DSK.DATADISK.GL DATA"

120 DELETE AS

Deletes the file named GL_DATA from the diskette named DATADISK.

Compact Computer File-Access Statements

This chapter defines the statements used to access diskette data files by the Compact Computer. If you have the Disk Drive/Controller connected to a 99/4A Home Computer, refer to section entitled "Home Computer File-Access Statements."

The BASIC statements used to access data files by the Compact Computer are listed below.

OPEN Prepares the computer to create or to use a data file by defining the file's characteristics.

PRINT a Writes information to a data file.

INPUT a Reads information from a data file.

LINPUT a Reads an entire record from a data file.

EOF Provides a program with a means of testing whether the end of a sequential file has been reached.

RESTORE Provides a program with a means of specifying the record to be read from or written to next.

CLOSE Terminates program access to a file.

DELETE Deletes a file from a diskette.

THE OPEN STATEMENT

The OPEN statement prepares the computer to access a data file by defining the characteristics of the file. The OPEN statement is used both to create a new file and to prepare the computer to read from or write to a file that already exists. A new file is automatically created when the computer does not find an existing file matching the parameters of the OPEN statement.

The Disk Drive/Controller permits a maximum of four files to be open at one time, regardless of the number of disk drives being used.

The OPEN statement has the following general form.

OPEN afile-number, "device.filename" [,file-organization] [,file-type] [,open-mode] [,record-length]

The <u>file-number</u> and <u>device.filename</u> parameters must be provided in every OPEN statement, but the other parameters can be included in any order, or omitted. A default characteristic is assumed for any parameter that is omitted.

!o! <u>file-number</u>—Every data file must be assigned an identification number from 1 through 255 when it is opened. This number provides the BASIC statements that read from and write to the file with an abbreviated method of referring to the file. Any numeric constant, expression, or variable having a value within the range 1 through 255 is a valid <u>file-number</u>, so long as that <u>file-number</u> is not assigned to another open file.

The <u>file-number</u> parameter must be preceded by a number symbol (α) and followed by a comma.

!o! device.filename—The device.filename 'parameter identifies the peripheral containing the file and the name of the file. If the device.filename specification is a string constant, it must be enclosed in quotation marks.

The device name can be specified in two ways:

- !o! By using a number from lal through la4, where the last digit of the number identifies the disk drive (1-4) on which you want the program to be saved. For example, la1 specifies disk drive 1 and la3 specifies disk drive 3.
- !o! By using the number 100, a period, and the name of a diskette. For example, 100.TI-DISK specifies the disk drive containing the diskette named TI-DISK.

The <u>filename</u> can be any combination of letters, numbers, and punctuation signs up to 1**D** characters in length, so long as the name does not contain any spaces or periods. If you use lower-case letters in your <u>filename</u>, the computer will convert them to upper-case letters before the file is opened.

The <u>device.filename</u> parameter must have a period separating the <u>device</u> from the <u>filename</u>. If the <u>device.filename</u> specification is a string constant, it must be enclosed in quotation marks. If the parameter is specified by a variable, it must be a string variable.

!o! file-organization—The file-organization parameter indicates whether the file is opened for sequential access or relative access. A sequential data file must be read or written in sequence, beginning with the first piece of information in the file. Relative data files can be read or written in any order, including sequentially. To specify sequential file access, omit the file-organization parameter. To specify relative file access, include the word RELATIVE in the OPEN statement.

!o! file-type-The file-type parameter specifies whether the information in the file is stored in DISPLAY format or INTERNAL format. If DISPLAY format is specified, data values are stored as ASCII characters. If INTERNAL format is specified, data values are stored in the binary format used to represent information in the computer's memory.

As a rule, it is preferable to use INTERNAL format for storing data in diskette files. Because no format conversions are required, data values in INTERNAL format are generally processed faster than data values in DISPLAY format. The DISPLAY option is intended primarily for files opened on devices other than disk drives.

If the file-type parameter is omitted, the computer assumes DISPLAY format.

!o! open-mode—The open-mode parameter specifies whether the file can be both read from and written to (UPDATE), only read from (INPUT), only written to (OUTPUT), or only added to (APPEND).

Note: If a file already exists on a diskette, opening the file for OUTPUT completely erases the previous contents of the file.

If the $\underline{\text{open-mode}}$ parameter is omitted, the computer assumes UPDATE mode.

:o: record-length—The record-length parameter consists of the word VARIABLE followed by a numeric value. If the file is opened for sequential access, the numeric value following the word VARIABLE specifies the maximum record length. If the file is opened for relative access, the value following the word VARIABLE specifies the length of each record. Note that the word VARIABLE is used even when a file is being opened for relative access.

The maximum record length for a sequential file is 65535 bytes. The maximum record length for a RELATIVE file is 255 bytes.

If the <u>record-length</u> parameter is omitted for an existing file, the computer uses the <u>record-length</u> established when the file was created.

If the <u>record-length</u> parameter is omitted when a new sequential file is opened, VARIABLE records with a maximum length of 8**D** bytes are assumed. If the <u>record-length</u> parameter is omitted when a new relative file is opened, a record-length of 255 bytes is assumed.

Examples:

100 OPEN a200, "102. PAYROLL", INPUT, INTERNAL

Opens a file named PAYROLL on the diskette in drive 2. The <u>file-number</u> is 2**DD**, the <u>open-mode</u> is INPUT, and the <u>file-type</u> is INTERNAL. Because no <u>file-organization</u> is specified, the file is a sequential file. The <u>record-length</u> is set to the value established when the file was created.

100 OPEN al, "101. TEMPFILE", RELATIVE, INTERNAL, OUTPUT, VARIABLE 64

Opens a file named TEMPFILE on the diskette in drive l. The <u>file-number</u> is l, the <u>file-organization</u> is RELATIVE, the <u>file-type</u> is INTERNAL, the <u>open-mode</u> is OUTPUT, and the <u>record-length</u> is 64 bytes.

100 A=3

110 AS="100.DATADISK.GL DATA"

12D OPEN aA,AS, RELATIVE

Opens a file named GL_DATA on the diskette named DATADISK. The <u>file-number</u> is 3 and the <u>file-organization</u> is RELATIVE. Because the <u>file-type</u> and <u>open-mode</u> parameters are omitted, the file is in DISPLAY format and is opened for UPDATE. The records are 255 bytes long if the file is new. If the file already exists, the <u>record-length</u> is set to the value established when the file was created.

The OPEN statement produces an error condition if:

- :o: The characteristics of an existing file and those of the OPEN statement do not match.
- to: The device specifed by the OPEN statement is not connected to the computer.
- !o! An attempt is made to OPEN more than four diskette files at one time.
- io! An attempt is made to OPEN a file that is already OPEN.
- io! An attempt is made to OPEN a new file on a diskette that is full.
- io! An attempt is made to OPEN a file on a diskette that is not initialized.
- io: An attempt is made to OPEN a file with invalid or contradictory parameters.

1927L HEX-BUS Disk Drive/Controller Part 2 to!=Bullet μ =Question mark Alpha=Number sign

THE PRINT & STATEMENT

The PRINT a statement writes information to a data file that has been opened in UPDATE, OUTPUT, or APPEND mode. Writing to a file opened in INPUT mode causes an error condition.

The PRINT a statement has the following general form:

PRINT afile-number [,REC record-number][,print-list]

The <u>file-number</u> parameter is required and must match the number assigned to the file by the OPEN statement. If the file was opened for RELATIVE access, the PRINT α statement can optionally specify the number of the record to write with the REC <u>record-number</u> parameter. Using a REC <u>record-number</u> parameter with a file opened for sequential access causes an error condition.

When the computer performs the PRINT α statement, the <u>print-list</u> data is stored in a temporary memory area called an input/output (I/O) buffer. A separate buffer is established for each file that is opened. If the <u>print-list</u> does not end with a comma or semicolon, the information in the I/O buffer is immediately written to the file. If the <u>print-list</u> does end with a comma or semicolon, a "pending" print condition occurs.

The pending print condition instructs the computer to wait until the next PRINT α statement is performed before writing the contents of the I/O buffer to the file. When the next PRINT α statement is performed, one of the following actions occurs:

io: If the next PRINT α statement has no REC parameter, the computer adds the <u>print-list</u> information of the second PRINT α statement to the information already stored in the I/O buffer. Then, unless the second PRINT α statement also ends in a comma or a semicolon, the computer writes the contents of the I/O buffer to the data file.

1927L HEX-BUS Disk Drive/Controller Part 2 $\pm o!$ =Bullet μ =Question mark Alpha=Number sign

:o! If the next PRINT α statement has a REC parameter, the computer writes the contents of the I/O buffer to the file before performing the action requested by the second PRINT α statement.

Examples:

100 PRINT al, X

Writes the value of X into the next record of the file opened as al.

100 PRINT a23, N; NAME 8; PAYRATE

Writes the values of N, NAMES, and PAYRATE into the next record of the file opened as $\alpha 23$.

100 PRINT a7, REC 44, "05/11/83"

Writes the string constant $\mathbf{0}5/11/83$ into record 44 of the file opened as $\alpha 7$. The file must be a relative file.

1**00** PRINT α3,A,B,C, 11**0** PRINT α3,D,E

The comma after the C in the first PRINT α statement creates a pending print condition. Therefore, the computer places the values of A, B, and C into the I/O buffer of the file opened as $\alpha 3$ and waits until the next PRINT α statement to determine when to write the information to the file. When the second PRINT α statement is executed, the values of D and E are added to the values already in the I/O buffer, and the contents of the buffer are written to the file.

THE INPUT a STATEMENT

The INPUT α statement reads information from a data file that has been opened in UPDATE or INPUT mode. Reading from a file opened in OUTPUT or APPEND mode causes an error condition.

The INPUT α statement has the following general form:

INPUT afile-number [,REC record-number][,variable-list]

The <u>file-number</u> parameter is required and must match the number assigned to the file by the OPEN statement. If the file was opened for RELATIVE access, the INPUT α statement can optionally specify the number of the record to read with the REC <u>record-number</u> parameter. Using a REC <u>record-number</u> parameter with a file opened for sequential access causes an error condition.

When the computer performs an INPUT a statement, the next entire record is read into the I/O buffer assigned to the file. The data values from the I/O buffer are then assigned to the variables in the variable-list in left-to-right order. After the variables have been assigned values, any data values remaining in the I/O buffer are discarded unless the variable-list ends with a comma. If the variable-list ends with a comma, a "pending" input condition is established.

The pending input condition instructs the computer to wait for the next INPUT α statement before deciding what to do with the unassigned data values. When the next INPUT α statement is executed, one of the following actions occurs:

- io! If the next INPUT α statement has no REC parameter, the unassigned values in the I/O buffer are assigned to the variables in the variable-list of the second INPUT α statement.
- io: If the next INPUT α statement has a REC parameter, the unused contents of the I/O buffer are discarded and the action specified by the second INPUT α statement is performed.

Examples:

100 INPUT al, X

Assigns the next data value from the file opened as αl to the variable X. If the next value is not a number, an error condition occurs.

100 INPUT a23,N,NAMES,PAYRATE

Assigns the next three data values from the file opened as $\alpha 23$ to the variables N, NAME 8, and PAYRATE. If the values for N and PAYRATE are not numbers, an error condition occurs.

100 INPUT a7, REC 44, DATES

Assigns the first data value in record 44 of the file opened as a7 to the variable DATES. The file must be a relative file.

100 INPUT a3,A,B,C,

The comma after the C creates a pending input condition. After assigning the next three data values from the file opened as $\alpha 3$ to the variables A, B, and C, the computer waits until the next INPUT α statement to decide whether to discard any data values remaining in the I/O buffer.

THE LINPUT & STATEMENT

The LINPUT α statement reads an entire record from a file containing data stored in DISPLAY format. If part of the record has been read already, the LINPUT statement reads only the remainder of the record.

The LINPUT α statement has the following general form:

LINPUT afile-number [,REC record-number],string-variable

The <u>file-number</u> is required and must match the number assigned to the file by the OPEN statement. If the file was opened for RELATIVE access, the LINPUT α statement can optionally specify the number of the record to read with the REC record-number parameter. Using the REC record-number parameter with a file opened for sequential access causes an error condition.

The information read from the file by the LINPUT α statement is assigned to the <u>string-variable</u>. The LINPUT α statement is not affected by the markers used to separate data values stored in the same record. Therefore, if a record contains two numeric values, a string value, and a final numeric value, all five values are assigned to the string-variable by the LINPUT α statement.

Using the LINPUT α statement to read a file stored in INTERNAL format causes an error condition. Using the LINPUT α statement to read a record longer than 255 pytes causes an error condition.

Examples:

100 LINPUT a1,L\$

Assigns the data in the next record of the file opened as $\alpha 1$ to the variable L\$.

100 LINPUT a7, REC 44, TEST\$

Assigns the data in record 44 of the file opened as $\alpha 7$ to the variable TEST\$. The file must be a RELATIVE file.

THE EOF FUNCTION

The EOF (End-of-File) function tests whether another record can be read from a sequential file. Using the EOF function with a RELATIVE file causes an error condition.

The EOF function has the following general form:

The <u>file-number</u> is required and must match the number assigned to the file by the OPEN statement.

The EOF function returns the following values:

<u>Value</u>	<u>Meaning</u>
1	End of File
Φ	Not End of File

An end-of-file result indicates that all the records in the file have been read. A not-end-of-file result indicates that at least one record remains to be read from the file.

Example:

100 IF EOF(1)=0 THEN 300

Tests the end-of-file condition for the file opened as α l. If the last record of the file has not been read, the program branches to line 300.

THE RESTORE STATEMENT

The RESTORE statement enables a program to reposition itself to read or write from the beginning of a file or, for RELATIVE files, from a specific record in a file.

The RESTORE statement has the following general form:

RESTORE afile-number [,REC record-number]

The <u>file-number</u> parameter is required and must match the number assigned to the file by the OPEN statement. If the file was opened for RELATIVE access, the RESTORE statement can optionally specify the number of the record to read or write next with the REC <u>record-number</u> parameter. Using a REC <u>record-number</u> parameter with a file opened for sequential access causes an error condition.

When used without the REC <u>record-number</u> parameter, the RESTORE statement repositions the program to the beginning of the file.

Examples:

100 RESTORE al

Repositions the computer's internal record counter to the beginning of the file opened as α l. The next INPUT α or PRINT α statement will read from or write to the first record in the file.

10D RESTORE a7, REC 44

Positions the computer's internal record counter to record number 44 of the file opened as $\alpha 7$. The next INPUT α or PRINT α statement will read from or write to record 44.

1927L HEX-BUS Disk Drive/Controller Part 2 10!=Bullet u=Question mark Alpha=Number sign

THE CLOSE STATEMENT

The CLOSE statement instructs the computer to terminate access to a file. A closed file cannot be accessed again until another OPEN statement is performed.

The CLOSE statement has the following general form:

CLOSE afile-number [,DELETE]

The <u>file-number</u> is required and must match the number assigned to the file by the OPEN statement. The DELETE parameter is optional and enables a program to erase a file from a diskette when the file is closed.

It is vital to close an open data file. As part of the closing process, the computer writes any information remaining in the I/O buffer to the file. If a file is not closed, that information is lost. The computer closes all open files if any of the following actions occur:

- !o! A program is stopped with a STOP or END statement.
- io! A NEW or NEW ALL command is executed.
- to! A program line is edited.
- io! The Compact Computer is turned off.

Example:

1ΦD OPEN α7: "DSK2.DISKFILE", INPUT, INTERNAL

110 CLOSE a7

Closes the file assigned file-number 7.

1927L HEX-BUS Disk Drive/Controller Part 2 to!=Bullet u=Question mark Alpha=Number sign

THE DELETE STATEMENT

The DELETE statement erases a data file or program from a diskette. The statement has the following general form:

DELETE "device.filename"

The <u>device.filename</u> parameter identifies the disk drive containing the file, and the name of the file. Refer to the discussion of the OPEN statement for information about the device.filename parameter.

Examples:

DELETE "102.PAYROLL"

Deletes the file named PAYROLL from the diskette in drive 2.

11D AS="100.DATADISK.GL_DATA"

12D DELETE AS

Deletes the file named GL DATA from the diskette named DATADISK.

Using Data Files

A diskette data file is a collection of information stored as a unit on a diskette. The information in the file is available for use by a program, but is stored separately from any program files that may also be located on the diskette. By storing data in a file, you can avoid having to retype information each time a program is run. Instead, data can be entered into a program once, stored in a file, and then read from the file into the computer's memory when needed again. Diskettes retain information until erased or written over, enabling you to store data independent of the computer's memory.

Your <u>HEX-BUS</u>TM disk system permits two types of data files to be stored on diskettes:

!o! Sequential data files

:o: Relative data files

Sequential data files, or serial files as they are sometimes called, consist of information that must be read or written in sequence, beginning with the first piece of information in the file. Relative data files, or random files as they are also called, can be read or written in any order. Thus, what distinguishes the two file types is not the kind of information stored in the files, which can be identical, but the manner in which the information is accessed.

The file type that you should use for a particular program depends upon how the information in a file is used by the program. Programming applications that require a base of information that does not change often, or that process information in a linear fashion, are well suited to sequential file-access methods. An example of an application in which file data is often processed sequentially is a payroll program, where the earnings of all employees are updated at the end of the pay period.

On the other hand, if a programming application requires that the information stored in a file be available in any order, you should use a relative file. An example of an application that is well suited for relative file access is an inventory program, where it is desirable to change inventory information each time a sale or order is made.

SEQUENTIAL ACCESS

A sequential data file can be visualized as a series of individual data values stored in a long row. This arrangement makes data access very simple—just start with the first data value and read or write until the last data value is reached. Because most sequential files use variable—length records, the file can be tightly packed with no space wasted between individual values. Thus, sequential files are simple to use and space efficient.

When a sequential file is opened in OUTPUT mode, the computer is set to start writing at the beginning of the file, even if the file already contains data. If a file matching the characteristics of the OPEN statement already exists, any information previously stored in the file is lost.

Note: Although a sequential file can be written to in UPDATE mode without destroying the file's previous contents, it is extremely difficult to manipulate successfully a sequential file in UPDATE mode. If you must write to a file without destroying the file's previous contents, use relative file access and not sequential file access.

If you want to add information to the end of a sequential file, you can open the file in APPEND mode. APPEND mode instructs the computer to permit a program to write to the end of the file. If a new file is opened in APPEND mode, the computer is set to write to the beginning of the file, just as if the file had been opened in OUTPUT mode. (Note: Only sequential files can be opened in APPEND mode—opening a relative file in APPEND mode causes an error condition.)

When a sequential file is opened in INPUT mode, the computer is set to read from the beginning of the file. There are no exceptions to this rule. Thus,

to access a data value in the middle of a sequential file, it is necessary to read from the beginning of the file until you reach the item of information you are seeking.

Data in a sequential file can be stored in either fixed— or variable—length records. Variable—length records store information without wasting space between data values, and are used for the vast majority of sequential files. Fixed—length records can be useful in cases where you may later want to access the same file as a relative file. For details on calculating the length of fixed—length records, refer to the section entitled "Relative Access."

When using variable-length records, be careful not to write a data value longer than the maximum record length. The maximum record length is specified by following the word VARIABLE in the OPEN statement with a numeric value. If no maximum length is specified, the Disk Drive/Controller assigns a default maximum length of 8D bytes. If a program attempts to write a data item with a length greater than the maximum record length, a file error occurs.

Sequential files can be written in either DISPLAY or INTERNAL format. As a rule, INTERNAL format is preferable, because data stored in INTERNAL format can be written and read faster (the computer does not have to convert the data to memory format). Also, on the average, numbers require less storage space on a diskette when stored in INTERNAL format. The default specification for file-type is DISPLAY, so remember to add INTERNAL to the OPEN statement when you do not want DISPLAY format.

For more details on the differences between DISPLAY and INTERNAL format, refer to the section entitled "Relative Access."

Creating a Sequential File

A sequential file is the simplest type of data file to create:

- OPEN the file in OUTPUT mode. To speed up the file-access procedures, specify INTERNAL format. As a rule, specify VARIABLE length records with a maximum record length of 254 bytes. If you know that a maximum length of 8D bytes is acceptable for your application, you can permit the OPEN statement to default to VARIABLE length records.
- 2. Write the data to the file using the PRINT α statement, in the order that you want the information stored.
- 3. CLOSE the file when the last item of data has been written to the file.

The following program demonstrates the process of creating a sequential file. Two versions of the program are given, one for a Home Computer and one for a Compact Computer. Enter the version that is appropriate for your computer.

Home Computer Version

100 REM SEQUENTIAL OUTPUT EXAMPLE—HOME COMPUTER VERSION

110 DISPLAY "NOW OPENING THE FILE..."

120 OPEN al: "DSK1.OUT1", OUTPUT, INTERNAL, VARIABLE 254

130 DISPLAY "NOW WRITING TO THE FILE..."

140 PRINT al: "THIS IS AN EXAMPLE OF A RELATIVELY LONG DATA ITEM"

15₽ PRINT al: "NOW"

16**₽** PRINT α1:"A"

176 PRINT al: "SERIES"

18**₽** PRINT al:"OF"

190 PRINT al: "SHORTER"

200 PRINT al:"DATA"

210 PRINT al:"ITEMS"

220 DISPLAY "NOW CLOSING THE FILE..."

230 CLOSE al

240 DISPLAY "DONE!"

Compact Computer Version

100 :Sequential Output Example—Compact Computer Version

110 DISPLAY "Now opening the file..."

120 OPEN al, "101.OUT1", OUTPUT, INTERNAL, VARIABLE 254

13D DISPLAY "Now writing to the file..."

140 PRINT al, "This is an example of a relatively long data item"

150 PRINT al, "Now"

16Φ PRINT α1, "a"

170 PRINT al, "series"

180 PRINT al, "of"

190 PRINT al, "shorter"

200 PRINT α1, "data"

21 PRINT al, "items"

220 DISPLAY "Now closing the file..."

230 CLOSE al

240 DISPLAY "Done:":PAUSE

After verifying that you have typed the program correctly, $\underline{\text{RUN}}$ it. The program instructs the computer to open a sequential file on the diskette in drive 1 and write the data specified in the PRINT α statements to the file. The file is assigned the name OUT1. After the last PRINT α statement is executed, the file is closed and program execution stops.

Reading a Sequential File

To read a sequential data file:

- OPEN the file in INPUT mode. Remember that the OPEN statement must match the characteristics assigned to the file when it was created. When you open the file in INPUT mode, the computer is set to begin reading at the beginning of the file.
- 2. Read the data in the file using the INPUT α statement. Use the EOF function before each INPUT α statement to ensure that the program does not attempt to read past the end of the file.
- 3. CLOSE the file when the last item of data has been read (or all the information needed has been read).

The following program demonstrates the general procedure to read a sequential file. This program reads and displays the information written to the file named OUTl by the sequential output example in the "Creating a Sequential File" section. Two versions of the program are given, one for a Home Computer and one for a Compact Computer. Enter the version that is appropriate for your computer.

Home Computer Version

- 100 REM SEQUENTIAL INPUT EXAMPLE—HOME COMPUTER VERSION
- 110 DISPLAY "NOW OPENING THE FILE..."
- 12₽ OPEN al: "DSK1.OUT1", INPUT, INTERNAL, VARIABLE 254
- 130 DISPLAY "NOW READING THE FILE..."
- 140 REM TEST IF END OF FILE HAS BEEN REACHED
- 150 IF EOF(1) < 1 THEN 190
- 160 INPUT al: AS
- 170 DISPLAY AS
- 180 GOTO 15D
- 190 DISPLAY "NOW CLOSING THE FILE..."
- 200 CLOSE al
- 210 DISPLAY "DONE!"

Compact Computer Version

100 :Sequential Input Example—Compact Computer Version

110 DISPLAY "Now opening the file..."

120 OPEN α1, "101.OUT1", INPUT, INTERNAL, VARIABLE 254

13D DISPLAY "Now reading the file..."

140 !Test if end of file has been reached

150 IF EOF(1) &D THEN 190

160 INPUT al, A8

170 DISPLAY AS: PAUSE 1

180 GOTO 150

190 DISPLAY "Now closing the file..."

200 CLOSE al

210 DISPLAY "Done!": PAUSE 5

The major difference between the writing and reading procedures is in the use of the EOF function. The EOF function provides a program with a means of testing whether any records remain unread in a sequential file. If the EOF function is not used, the program will eventually attempt to INPUT a data value when none remain to be read, causing an error condition to occur.

The EOF function returns a value of **o** when the file contains data to be read. If the EOF function returns any value other than **o**, the end of the file was reached by the previous INPUT statement and no data values remain to be read. (The value returned by the EOF function when the end of the file has been reached varies depending upon whether you are using a Home Computer or a Compact Computer. Refer to the file-access sections of this manual for details.)

Additional Sequential Access Examples

In the preceding examples, the data consists of string information only. When creating files containing both numeric and string information, you should know the order or pattern of the data in the file. Otherwise, you may find it difficult to read the data back into the computer successfully. The following program provides an example of storing numeric and string data in the same file; it also demostrates adding data to an existing file using the APPEND mode.

Two versions of the program are given, one for a Home Computer and one for a Compact Computer. Enter the version that is appropriate for your computer.

Home Computer Version

The first part of the program enables you to select the type of file operation you want to perform from a "menu" of choices.

100 REM CHECK TRACKING PROGRAM—HOME COMPUTER VERSION

110 DISPLAY "ENTER 1 TO CREATE A NEW FILE"

120 DISPLAY "ENTER 2 TO ADD TO THE FILE"

130 DISPLAY "ENTER 3 TO TOTAL THE CHECKS"

140 DISPLAY "ENTER 4 TO DELETE THE FILE"

150 DISPLAY "ENTER 5 TO END THE PROGRAM"

16₽ INPUT "YOUR SELECTIONµ (1-5)":Q

170 IF Q35 THEN 110

18D ON Q GOTO 190,280,370,520,540

The next part of the program creates a new file and permits data to be stored in the file. If you press ENTER in response to the CHECK PAYABLE TO prompt, the program assumes that you have no more checks to enter. (The file is closed before execution returns to the menu.)

190 REM CREATE A NEW FILE

200 OPEN α1: "DSK1. CHECKS", OUTPUT, INTERNAL, VARIABLE 254

210 INPUT "CHECK PAYABLE TO" ": PAYEES

22D IF PAYEES="" THEN 26D

230 INPUT "AMOUNT" ": AMOUNT

240 PRINT al:PAYEES; AMOUNT

250 GOTÓ 210

260 CLOSE al

270 GOTO 110

Lines 28D through 36D add data to an existing file. Again, pressing ENTER in response to the CHECK PAYABLE TO prompt ends the entry of data.

28D REM ADD TO AN EXISTING FILE

290 OPEN al: "DSK1.CHECKS", APPEND, INTERNAL, VARIABLE 254

3DD INPUT "CHECK PAYABLE TO" ": PAYEES

310 IF PAYEE 8="" THEN 350

320 INPUT "AMOUNT" ": AMOUNT

33D PRINT al:PAYEES; AMOUNT

34D GOTO 30D

35D CLOSE al

360 GOTO 110

Lines 37Φ through 51Φ read and display the file data, count the number of checks, and total the amounts. When the end of the file is reached, the number of checks and the total amount are displayed.

370 REM TOTAL THE FILE

38**0** TOTAL=**5**

39D COUNT=D

400 OPEN al: "DSK1. CHECKS", INPUT, INTERNAL, VARIABLE 254

410 IF EOF(1) <> D THEN 470

420 INPUT al:PAYEES, AMOUNT

430 DISPLAY PAYEES; AMOUNT

44 COUNT=COUNT+1

45 TOTAL=TOTAL+AMOUNT

460 GOTO 410

470 CLOSE al

480 DISPLAY "NUMBER OF CHECKS ="; COUNT

490 DISPLAY "TOTAL ="; TOTAL

500 GOTO 110

The next section deletes the file from the diskette. Once a file is deleted, the information in the file is lost.

51 D REM DELETE THE FILE 52 DELETE "DSK1.CHECKS" 53 D GOTO 11 D

The last lines end the program.

54**D** REM END PROGRAM 55**D** END

Compact Computer Version

The first part of the program enables you to select the type of file operation you want to perform from a "menu" of choices.

lad: Check Tracking Program—Compact Computer Version
11ad DISPLAY "N)ew A)dd T)otal D)elete E)nd"

120 ANS=KEYS

13D IF ANS="N" OR ANS="n" THEN 19D

140 IF ANS="A" OR ANS="a" THEN 280

150 IF ANS="T" OR ANS="t" THEN 370

160 IF ANS="D" OR ANS="d" THEN 510

170 IF ANS="E" OR ANS="e" THEN 550

180 DISPLAY BEEP: GOTO 110

The next portion of the program creates a new file and permits data to be stored in the file. If you press ENTER in response to the Check payable tou prompt, the program assumes that you have no more checks to enter. (The file

is closed before execution returns to the menu.)

19b :Create a new file

200 DISPLAY "Creating a new file..."

21 OPEN α1, "101. CHECKS", OUTPUT, INTERNAL, VARIABLE 254

220 INPUT "Check payable tou ":PAYEE8

23D IF PAYEE8="" THEN 27D

24D INPUT "Amountu "; AMOUNT

250 PRINT al, PAYEES, AMOUNT

260 GOTO 220

27 CLOSE al:GOTO 11 D

Lines 28D through 36D add data to an existing file. Again, pressing ENTER in response to the Check payable to prompt ends the entry of data.

28D:Add to an existing file

29D DISPLAY "Adding to file..."

300 OPEN α1, "101. CHECKS", APPEND, INTERNAL, VARIABLE 254

310 INPUT "Check payable tou ";PAYEE8

32D IF PAYEES="" THEN 36D

33D INPUT "Amount_µ"; AMOUNT

34 PRINT al, PAYEES, AMOUNT

350 GOTO 310

360 CLOSE al:GOTO 110

Lines 370 through 500 read and display the file data, count the number of checks, and total the amounts. When the end of the file is reached, the number of checks is displayed. Press ENTER to display the total amount.

370 !Total the file

380 DISPLAY "Totaling the file..."

39₽ TOTAL=1: COUNT=₽

4ΦΟ OPEN α], "1Φ]. CHECKS", INPUT, INTERNAL, VARIABLE 254

410 IF EOF(1) <> D THEN 47

420 INPUT al, PAYEES, AMOUNT

430 DISPLAY PAYEES; AMOUNT: PAUSE .5

440 COUNT=COUNT+1

450 TOTAL=TOTAL+AMOUNT

460 GOTO 410

470 CLOSE α1

480 DISPLAY BEEP, "Number of checks ="; COUNT: PAUSE

490 DISPLAY BEEP, "Total ="; TOTAL: PAUSE

The next section deletes the file from the diskette. Once a file is deleted, the information in the file is lost.

510 !Delete the file 520 DISPLAY "Deleting the file..." 530 DELETE "101.CHECKS" 540 GOTO 110

The last lines end the program.

55**D** :End program 56**D** END

500 GOTO 110

RUN the program and experiment with the various options.

Note: If you attempt to total the file before storing data in the file, an error condition occurs.

Another Home Computer Example

The next program provides a final example of sequential file access for Home Computer owners. No corresponding example is provided for Compact Computer owners because the Compact Computer does not have the same graphics commands.

Lines 11D through 18D of the following program create a random screen display of alphabetic characters. Lines 19D through 28D write the screen's contents one character at a time to a sequential file. The GCHAR subprogram gets each character from the screen. A program such as this can be used as a subroutine in other programs when you want to save a screen that you have created.

- 100 REM SAVE SCREEN EXAMPLE
- 110 RANDOMIZE
- 120 CALL CLEAR
- 130 FOR COUNT=1 TO 100
- 140 CHAR=INT(26*RND)+65
- 15D ROW=INT(24*RND)+1
- 16 COL=INT(32*RND)+1
- 170 CALL HCHAR (ROW, COL, CHAR)
- 180 NEXT COUNT
- 19**p** CALL COLOR(12,9,9)
- 200 OPEN a7: "DSK1.SCREEN", OUTPUT, INTERNAL
- 21**D** FOR ROW=1 TO 24
- 22**P** FOR COL=1 TO 32
- 23D CALL GCHAR (ROW, COL, CHAR)
- 240 CALL HCHAR (ROW, COL, 127)
- 250 PRINT a7:CHAR
- 260 NEXT COL
- 270 NEXT ROW
- 28**D** CLOSE a7

Notice when you RUN the program that the computer does not turn on the disk drive every time it gets a character from the screen. Instead, the computer places the characters to be written to the data file in an I/O buffer and turns on the disk drive and writes the data only when the buffer is full.

The following program reads and displays the SCREEN file.

- 100 REM RESTORE SCREEN EXAMPLE
- 11**D**CALL COLOR(12,9,9)
- 12**0** CALL HCHAR(1,1,127,768)
- 13 D OPEN α11: "DSK1.SCREEN", INPUT, INTERNAL
- 14**D** FOR ROW=1 TO 24
- 15**D** FOR COL=1 TO 32
- 16D INPUT all:CHAR
- 17 CALL VCHAR (ROW, COL, CHAR)
- 18D NEXT COL

190 NEXT ROW
200 CLOSE all
210 REM DELAY TO GIVE TIME TO VIEW SCREEN
220 FOR X=1 TO 2500
230 NEXT X

Notice that the EOF function is not needed in this program because the number of records in the file is known.

RELATIVE ACCESS

One of the most powerful features of your disk system is its capability to manipulate relative files. Unlike sequential files, which permit records to vary in length, relative files store information in equal-sized records. Because each record in a relative file has a fixed length, the computer can calculate where any record is located, without having to search from the beginning of the file. As a result, a relative file can be written or read in any order.

For example, relative access enables a program to create a new file and write data only to record 21. In this case, the computer sets aside space in the file for records 1 through 2**D**, but will not write data to these records. Records 1 through 2**D** are left empty (an empty record contains only binary zeros).

On the Home Computer, the length of each record in a relative file is specified by following the word FIXED in the OPEN statement with a numeric value of 255 or less. On the Compact Computer, the length of each record in a relative file is specified by following the word VARIABLE in the OPEN statement with a numeric value of 255 or less (the Compact Computer does not have the keyword FIXED, and so must use the word VARIABLE). If no record length is specified, the OPEN statement defaults to a record length of 255 bytes.

In addition to the versatility of accessing records in any order, relative access makes it easy to change data in a file. Because the computer can find a specific record in a relative file, it is possible to open the file in UPDATE mode and write data only to the records that need to be corrected. UPDATE mode enables the program to write data to the file without erasing the previous contents of the file. (Remember, when a file is opened in OUTPUT mode, the previous contents of the file, if any, are lost.)

A disadvantage of relative files is that the fixed-record length wastes diskette space when only a small amount of information is stored in a record. If a relative file contains data items of many different lengths, some records will be full and others will be only partially filled, or even empty. As a rule, however, the greater speed and flexibility of relative access more than compensates for its less efficient use of space.

Relative files are well suited for applications requiring direct access to any piece of data stored in a file, or to applications where it is often necessary to change some, but not all, of the information in a file.

Creating a Relative File

The general procedure for creating a relative file is:

- 1. OPEN the file in OUTPUT mode. Identify the file as a relative file by including the RELATIVE parameter in the OPEN statement. If you want a record length of less than 255 bytes, specify the desired record length following the word FIXED for the Home Computer, or the word VARIABLE for the Compact Computer. Unless you want the file to be in DISPLAY format, specify INTERNAL format.
- 2. Write the data to the file using the PRINT α statement. Unless you want the computer to write the data to the <u>next</u> record in the file, use the REC <u>record_number</u> parameter to specify where you want the information stored. The first record in a file is record number $\boldsymbol{\sigma}$.
- 3. CLOSE the file when the last data value has been written to the file.

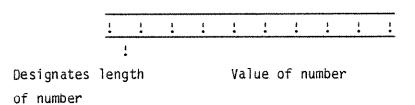
The main difference between writing to relative files and writing to sequential files is that the PRINT α statement normally gives the number of the record being written.

Creating a relative file requires more planning than creating a sequential file, because you must determine in advance the record size to use. To calculate the correct record size, you must anticipate the maximum length of the information that will be stored in each record. If you select too small a record size, an error will occur when the program attempts to write a larger data item to the file. If you select too large a record size, your file will contain much wasted space.

The amount of space required to store a numeric or string value depends upon whether the file is to be written in INTERNAL or DISPLAY format. As a rule, it is preferable to use INTERNAL format for storing data in a relative file. Not only does the computer access data faster in INTERNAL format, but the fixed length of numbers in INTERNAL format makes it easier to calculate their storage requirements. The following rules will help you determine the record size to use.

Numbers

!o! In INTERNAL format, numeric values require 9 bytes of storage space. Thus, the number 5 and the number 5555.5555 require the same amount of storage space. The 9 bytes required to store a number in INTERNAL format are allocated as shown below:



Note: Although the Home Computer and the Compact Computer use 9 bytes to store numbers in INTERNAL format, the two computers do not use identical formats. Thus, neither computer can directly read numeric data from files written in INTERNAL format by the other computer. A subprogram that converts numeric data from Home Computer INTERNAL format to values the Compact Computer can understand is given in "Reading the Catalog with a Compact Computer."

io! In DISPLAY format, the amount of space required to store a number can be calculated as follows: One byte is required for each digit in the number, one byte for the decimal point if any, one byte for the minus sign if the number is a negative number or for the leading space if the number is a positive number, and one byte for the trailing space that always follows a number in DISPLAY format.

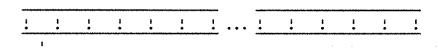
The following table gives the storage requirements for several numbers in DISPLAY format.

<u>Value</u>	Le	ength				
.0007	7	bytes				
1.0	3	bytes	(the	.0	is	dropped)
- 5	3	bytes				
5	3	bytes				
5555.5555	11	bytes				
.12345678	11	bytes				
-7.23E-44	D	bytes				
9.99999E-99	13	bytes				

The space requirements for numbers in DISPLAY format vary widely, making it difficult to pick an efficient record length when DISPLAY format is used. The maximum amount of storage space a number in DISPLAY format can take is 13 bytes (as in the last example in the table).

Strings

!o! In INTERNAL format, string items require one byte for each character in the string, and one byte for a length designator, as illustrated below.



Designates length of string

Value of string

:o: In DISPLAY format, string items require one byte for each character in the string. There is no length designator for strings in DISPLAY format.

The following example illustrates the process of creating a relative file.

This program stores a part number, a part description, and a quantity on hand for a business inventory program. The information is stored in a file named PARTS on the diskette in drive 1.

Two versions of the program are given, one for a Home Computer and one for a Compact Computer. Enter the version appropriate for your computer.

Home Computer Version

- 100 REM CREATE INVENTORY FILE-HOME COMPUTER VERSION
- 11 D INPUT "NUMBER OF ITEMS" ": ITEMS
- 12 OPEN α44: "DSK1.PARTS", OUTPUT, RELATIVE, INTERNAL, FIXED 59
- 130 PRINT a44, REC 0: ITEMS
- 140 FOR RECORD=1 TO ITEMS
- 15D INPUT "PART NUMBER" ": NUMB
- 16D INPUT "DESCRIPTION" ":DESC8
- 17D DESC8=SEG8 (DESC8, 1, 4D)
- 18D INPUT "QUANTITY" ":QUANT
- 19D PRINT a44, REC RECORD: NUMB, DESCS, QUANT
- 200 NEXT RECORD
- 210 CLOSE a44

Compact Computer Version

- 100 :Create Inventory File--Compact Computer Version
- 110 INPUT "Number of itemsu "; ITEMS
- 120 OPEN a44, "101. PARTS", OUTPUT, RELATIVE, INTERNAL, VARIABLE 59
- 130 PRINT a44, REC O, ITEMS
- 140 FOR RECORD=1 TO ITEMS
- 150 INPUT "Part number" "; NUMB
- 16D INPUT "Descriptionu ";DESCS
- 170 DESC8=SEG8(DESC8, 1, 40)
- 18D INPUT "Quantityu ";QUANT
- 190 PRINT a44, REC RECORD, NUMB, DESCS, QUANT
- 200 NEXT RECORD
- 210 CLOSE a44

Before running the program, note the following points:

- !o! The program stores the number of records to be written to the file (ITEMS) in the first record of the file, a common practice in creating a relative file. This technique ensures that a program that reads the file has a method of determining the number of records in the file. The EOF function, which enables a program to test whether the end of a sequential file has been reached, cannot be used with relative files.
- !o! The program writes a part number, description, and quantity on hand to each record by including the three data values in the same PRINT α statement (separated by commas). The commas instruct the computer to write the data to the same record. If three separate print statements were used, the three data values would be written to separate records. Using three records to store this information would use storage space less efficiently since the length of the numbers and the length of the description vary so widely.

In the record length is set at 59 bytes. Because the part number and quantity each require only 9 bytes of storage, the record contains enough additional space to store a description that is 40 characters (and 1 length byte) long: 9 + 9 + 41 = 59. To prevent a file error from occurring when if a description longer than 40 characters is entered, the SEGS function is used to shorten any descriptions that exceed 40 characters in length (line 170).

RUN the program and enter sample data such as that shown below. A program to update this file is given in the next section.

Part Number	t Number <u>Description</u>	
111	Box of 1D Diskettes	2 00
131	Disk Drive	23
137	Drive Cables	24

Updating a Relative File

The general procedure for updating a relative file is given below. Because updating a relative file is more common than simply reading it, no separate directions are given for reading a file. As a matter of general practice, many programmers use UPDATE mode even when they intend only to read a file.

- 1. OPEN the file in UPDATE mode. Because UPDATE is the default mode for relative files, omitting the <u>open-mode</u> parameter automatically opens the file for updating. Identify the file as a relative file by including the RELATIVE parameter. If the file has a record length of less than 255 bytes, specify the appropriate record length following the word FIXED for the Home Computer or VARIABLE for the Compact Computer. Unless the file is in DISPLAY format, specify INTERNAL format.
- 2. UPDATE mode permits both reading from and writing to the file. Write to the file using the PRINT α statement. Read from the file using the INPUT α statement. Unless you want the computer to access the next record in the file, use the REC record-number parameter to specify which record to access. Remember that the first record in a file is record $\boldsymbol{\Phi}$.

3. CLOSE the file when done.

When a file is opened in UPDATE mode with characteristics that do not match any existing file, the computer creates a new file. If a matching file is found, the computer permits the file to be written to, read from, or both. Existing data values in the file being UPDATED are not lost, as they are when a file is opened in OUTPUT mode.

The example that follows illustrates how to update an existing file. This program reads the inventory file created in the "Creating a Relative File" section and permits the quantity-on-hand value to be changed for an existing inventory item. The program also enables you to add new inventory items to the file.

Two versions of the program are given, one for the Home Computer, and one for the Compact Computer. Enter the version appropriate for your computer.

Home Computer Version

Lines 100 through 170 display the menu of choices and enable you to choose the action to perform.

REM UPDATE INVENTORY FILE-HOME COMPUTER VERSION

110 CALL CLEAR

12D DISPLAY : "ENTER 1 TO FIND A PART"

130 DISPLAY "ENTER 2 TO ADD A PART"

14D DISPLAY "ENTER 3 TO END PROGRAM"::

150 INPUT "YOUR SELECTION (1-3) ":0

16 P IF Q>3 THEN 12 D

170 ON Q GOTO 180, 350, 470

Lines 18D through 34D search the file for a specific part number. If the part is found in the file, the description and quantity are displayed and the program enables you to enter a new quantity. If no matching part is found, a message is displayed and execution is returned to the menu. Notice that the program reads the first record in the file (record Φ) to determine the number

of items in the file.

18D REM FIND A PART NUMBER

19 D OPEN α44: "DSK1.PARTS", UPDATE, RELATIVE, INTERNAL, FIXED 59

200 INPUT a44, REC O: ITEMS

210 INPUT "PART NUMBER TO FIND" ":PART

220 REM SEARCH FILE

23D FOR RECORD=1 TO ITEMS

240 INPUT a44, REC RECORD: NUMB, DESCS, QUANT

25D IF NUMB=PART THEN 30D

26D NEXT RECORD

27 DISPLAY : "PART"; PART; "NOT FOUND"

28D CLOSE a44

29 D GOTO 12 D

300 DISPLAY : DESCS; QUANT

310 INPUT "NEW QUANTITY" ":QUANT

32 PRINT a44, REC RECORD: NUMB, DESC8, QUANT

33**D** CLOSE @44

340 GOTO 120

The next section, lines 35 \bullet through 46 \bullet , add a new inventory item to the file. (To keep this example short, no check is made to verify that you are not entering a duplicate part number.) The information about the new part is added at the end of the file and the count of items in record \bullet is updated.

35 REM ADD A NEW PART

36₽ INPUT "NEW PART NUMBER" ":NEWPART

37D INPUT "DESCRIPTION" ": NEWDESC8

38D NEWDESCS=SEGS (NEWDESCS, 1, 4D)

390 INPUT "QUANTITY" ":NEWQUANT

400 OPEN α44: "DSK1.PARTS", UPDATE, RELATIVE, INTERNAL, FIXED 59

410 INPUT a44, REC Q: ITEMS

420 PRINT a44, REC ITEMS+1: NEWPART, NEWDESCS, NEWQUANT

43D ITEMS=ITEMS+1

440 PRINT a44, REC Q: ITEMS

45**0** CLOSE α44 46**0** GOTO 12**D**

The final section ends the program.

47♠ REM END PROGRAM 48♠ CLOSE ∝44

Compact Computer Version

Lines 100 through 160 display a menu of choices and enable you to choose the action to perform.

100 !Update Inventory File—Compact Computer Version

11D DISPLAY "F) ind part A)dd new part E)nd"

120 ANS=KEYS

130 IF ANS="F" OR ANS="f" THEN 170

14D IF ANS="A" OR ANS="a" THEN 33D

15D IF ANS="E" OR ANS="e" THEN 46D

16D DISPLAY BEEP: GOTO 110

Lines 17D through 32D search the file for a specific part number. If the part is found in the file, the description and quantity are displayed and the program enables you to enter a new quantity. If no matching part is found, a message is displayed and execution is returned to the menu. Notice that the program reads the first record in the file (record \mathbf{O}) to determine the number of items in the file. (As noted earlier, the EOF function cannot be used with relative files.)

170! Find a part number

180 DISPLAY "Opening the file..."

190 OPEN a44, "101. PARTS", UPDATE, RELATIVE, INTERNAL, VARIABLE 59

200 INPUT a44, REC 0, ITEMS

216 INPUT "Part number to findu ";PART

220 DISPLAY "Searching the file..."

23D FOR RECORD=1 TO ITEMS

240 INPUT a44, REC RECORD, NUMB, DESCS, QUANT

250 IF NUMB=PART THEN 290

26D NEXT RECORD

270 DISPLAY BEEP, "Part"; PART; "not found": PAUSE

280 CLOSE @44:GOTO 110

290 DISPLAY DESCS; QUANT: PAUSE

300 INPUT "Enter new quantity: ";QUANT

310 PRINT a44, REC RECORD, NUMB, DESCS, QUANT

32D CLOSE @44:GOTO 110

The next section, lines 33 Φ through 45 Φ , add a new inventory item to the file. (To keep the program short, no check is made to verify that you are not entering a duplicate part number.) The information about the new part is added at the end of the file and the count of items in the file that is stored in record Φ is updated.

33D : Add a new part

34**b** INPUT "New part number_µ ";NEWPART

350 INPUT "Descriptionu "; NEWDESCS

36D NEWDESCS=SEGS(NEWDESCS, 1, 4D)

37D INPUT "Quantity" "; NEWQUANT

38D DISPLAY "Opening the file..."

39D OPEN a44."101. PARTS", UPDATE, RELATIVE, INTERNAL, VARIABLE 59

400 DISPLAY "Updating the file..."

41D INPUT a44, REC Q, ITEMS

42 PRINT α44, REC ITEMS+1, NEWPART, NEWDESCS, NEWQUANT

43D ITEMS=ITEMS+1

440 PRINT a44, REC Q, ITEMS

450 CLOSE @44:GOTO 110

. The final section ends the program.

46D !End program

47**D** END

Maintaining a Sorted Relative File

In many programming applications, it is desirable to maintain data in an alphabetically sorted order. One way to keep data sorted is to write the data values to the file in the order in which they are entered, and then periodically to read these values into the computer's memory, sort them, and rewrite them to the file. It is often more convenient, however, to write the data to the file in sorted order, as illustrated in the following program. A separate operation to sort the data does not have to be performed.

Home Computer Version

The first part of the program displays a menu of options and enables you to choose the action to perform.

100 REM SORTED FILE EXAMPLE -- HOME COMPUTER VERSION

110 CALL CLEAR

120 DISPLAY "ENTER 1 TO CREATE NEW FILE"

130 DISPLAY "ENTER 2 TO ADD TO THE FILE"

140 DISPLAY "ENTER 3 TO FIND A FILE NAME"

150 DISPLAY "ENTER 4 TO SHOW FILE DATA"

160 DISPLAY "ENTER 5 TO DELETE THE FILE"

170 DISPLAY "ENTER 6 TO END THE PROGRAM"

18♥ INPUT "YOUR SELECTION" (1-6)":Q

190 IF 041 THEN 110

200 IF Q36 THEN 110

210 ON Q GOTO 220,360,550,680,850,880

The next part of the program creates the mailing-list file. The section begins by prompting for the number of names to be recorded in the file initially, and storing that value in record \bullet . Then the mailing information for the first name is written to record 1 (the subroutine beginning at line 91D is used to prompt for the mailing data.) The data for the first name is written separately from any other names to be recorded in the file because it is not necessary to sort the file for only one name. Lines 29D through 33D get the mailing information for the remaining names and write it to the file

in sorted order. The subroutine beginning at line 127D is used to insert each name in sorted order.

220 REM CREATE A NEW FILE

230 INPUT "NUMBER OF NAMES" ": NAMECOUNT

240 IF NAMECOUNTY THEN 230

250 OPEN a2: "DSK1.MAILFILE", UPDATE, RELATIVE, INTERNAL, FIXED 98

260 PRINT @2, REC : NAMECOUNT

270 GOSUB 911

280 PRINT @2, REC 1: LNAMES, FNAMES, ADDRESSS, CITYS, STATES, ZIP

290 FOR COUNT=2 TO NAMECOUNT

300 GOSUB 910

31 LASTREC=COUNT-1

32**D** GOSUB 127**D**

33D NEXT COUNT

340 CLOSE @2

350 GOTO 110

The next section adds a mailing address to the established file. Lines 370 and 380 open the file and retrieve the number of records currently stored in the file. Line 390 calls the subroutine at line 910 to prompt for the information to add to the file. Line 400 calls the subroutine at 1050 to determine if the name is already stored in the file. If the name does not match any names already in the file, the name is added to the file, using the subroutine at line 1270 to add the record in sorted order.

36D REM ADD TO THE FILE

37Φ OPEN α2: "DSK1.MAILFILE", UPDATE, RELATIVE, INTERNAL, FIXED 98

38D INPUT a2, REC 4: NAMECOUNT

390 GOSUB 910

400 GOSUB 1050

410 IF MATCH8="YES" THEN 52D

420 LASTREC=NAMECOUNT

430 GOSUB 1270

44D NAMECOUNT=NAMECOUNT+1

45Φ PRINT α2, REC Q: NAMECOUNT

460 DISPLAY

47Φ INPUT "ADD ANOTHER NAME (YµN) ":ANS

480 IF ANS="N" THEN 500

49**9** GOTO 39**0**

500 CLOSE a2

51 C GOTO 110

520 DISPLAY :"NAME ALREADY IN FILE!"::

530 INPUT "PRESS ENTER TO CONTINUE: ": ANS

540 GOTO 500

Lines 550 through 670 search the file for a specific name. If the name is found, the mailing information is displayed. If not, the message $\underline{\text{NO MATCH}}$ FOUND: is displayed. The subroutine at line $\underline{\text{NO50}}$ is used to determine if a matching name exists in the file.

55₽ REM FIND A FILE NAME

56D OPEN a2: "DSK1.MAILFILE", INPUT, RELATIVE, INTERNAL, FIXED 98

57D INPUT a2, REC a: NAMECOUNT

58D INPUT "LAST NAME" ":LNAME &

59D INPUT "FIRST NAME" ": FNAME 8

600 GOSUB 1050

610 IF MATCH8="NO" THEN 661

620 DISPLAY :L8; ", ":F8:A8:C8: " ";S8; " ";Z

63D INPUT "PRESS ENTER TO CONTINUE: ": ANS

640 CLOSE a2

650 GOTO 110

66D DISPLAY : "NO MATCH FOUND!"::

676 GOTO 630

Lines 68D through 84D read and display the information in the file. The names are displayed five at a time.

680 REM SHOW FILE DATA

69₺ OPEN a2: "DSK1.MAILFILE", INPUT, RELATIVE, INTERNAL, FIXED 98

700 INPUT a2, REC O: NAMECOUNT

710 S=0

72 FOR COUNT=1 TO NAMECOUNT

73D S=S+1

740 INPUT a2:LNAMES, FNAMES, ADDRESSS, CITYS, STATES, ZIP

750 DISPLAY :LNAMES; ", "; FNAMES: ADDRESSS: CITYS; " "; STATES; " "; ZIP

76**D** IF S/5<8INT(S/5) THEN 79**D**

77**0** DISPLAY :::

78 INPUT "PRESS ENTER TO CONTINUE:":ANS

79D NEXT COUNT

8DD IF (COUNT-1)/5=INT((COUNT-1)/5) THEN 83D

810 DISPLAY

82 INPUT "PRESS ENTER TO CONTINUE:":ANS

83D CLOSE a2

84 D GOTO 11 D

The next section deletes the file from the diskette.

850 REM DELETE THE FILE

86D DELETE "DSK1.MAILFILE"

870 GOTO 110

The final section ends the program.

88D REM END PROGRAM

89**6** END

The subroutines begin with line 900. The first subroutine prompts for the information to be stored in the file and ensures that it does not exceed the length allocated for each data value. The record length of 98 bytes is allocated as follows: 31 bytes for the last name, 16 bytes for the first name, 23 bytes for the street address, 16 bytes for the city, 3 bytes for the state, and 9 bytes for the zip code. (Remember, each string value in INTERNAL format requires one extra byte for a length byte.)

900 REM ***** SUBROUTINES ******

910 REM GET FILE DATA

920 DISPLAY

```
930 INPUT "LAST NAME" ": LNAME &
```

940 LNAME &= SEG8 (LNAME & , 1, 30)

950 INPUT "FIRST NAME" ": FNAME &

96P FNAMES=SEGS(FNAMES, 1, 15)

97♠ INPUT "STREET ADDRESS" ":ADDRESS%

980 ADDRESS&=SEG&(ADDRESS&,1,22)

99₽ INPUT "CITY" ":CITY\$

1000 CITY8=SEG8(CITY8,1,15)

1010 INPUT "STATE ABBREVIATION" ":STATES

1020 STATE 8=SEG8 (STATE8, 1, 2)

1030 INPUT "ZIP CODE" ":ZIP

104D RETURN

The next subroutine searches the file for a record matching the values entered for LNAMES and FNAMES. Because the file is sorted, a technique known as a binary search can be used to search the file. In a binary search, the body of data to be searched is divided in half and a comparison is made to determine if the matching value would be located in the upper or lower half of the data. The half containing the matching value is then itself divided in half, and another comparison is made to determine which part would contain the matching value. This process of dividing the data in half is repeated until a matching value is found or it is determined that no matching value exists. A binary search is normally faster than a sequential search from the beginning of a file.

1050 REM SEARCH FILE FOR DUPLICATE RECORD

1060 HI=NAMECOUNT+1

107D LO=1

1080 MID=INT((HI+LO)/2)

1090 LASTMID=MID

1100 INPUT @2, REC MID: L8, F8, A8, C8, S8, Z

1110 IF (L8aF8)=(LNAMESaFNAMES) THEN 1180

.1120 IF (LSaFS) < (LNAMESAFNAMES) THEN 1230 ELSE 1250

113**D** MID=INT((HI+L0)/2)

1140 IF LASTMID=MID THEN 1210

1150 LASTMID=MID

1160 GOTO 1100

1170 REM MATCH FOUND

118**D** MATCH8="YES"

1190 RETURN

1200 REM NO MATCH FOUND

121**P** MATCH8="NO"

122P RETURN

1230 LO=MID

124**P** GOTO 113**P**

125P HI=MID

126D GOTO 1130

The final subroutine adds a new name in sorted order. The subroutine begins by reading the last record in the file and comparing the name in that record with the name to be added to the file. If the name in the last record of the file is greater than the name to be added, that record is moved to the next larger record position (increasing the file size by one record). This process is repeated until a record is found whose name is not greater than the name to be added. The new mailing information is written to the empty "slot" that was created by the search (and move) procedure.

1270 REM ADD A RECORD IN SORTED ORDER

128D FOR C=LASTREC TO 1 STEP -1

1290 INPUT a2, REC C: L8, F8, A8, C8, S8, Z

1300 IF (LSOFS)<(LNAMESOFNAMES) THEN 1340

131**0** PRINT α2, REC C+1: L8, F8, A8, C8, S8, Z

132**D** NEXT C

1330 C=0

1340 PRINT @2, REC C+1: LNAMES, FNAMES, ADDRESSS, CITYS, STATES, ZIP

1350 RETURN

Compact Computer Version

The first part of the program displays a menu of options and enables you to choose the action to perform.

100 :Sorted File Example-Compact Computer Version

110 DISPLAY "N)ew A)dd F)ind S)how D)el E)nd"

12**0** ANS=KEYS

130 IF ANS="N" OR ANS="n" THEN 210

140 IF ANS="A" OR ANS="a" THEN 340

150 IF ANS="F" OR ANS="f" THEN 510

160 IF ANS="S" OR ANS="S" THEN 640

170 IF ANS="D" OR ANS="d" THEN 740

180 IF ANS="E" OR ANS="e" THEN 780

190 DISPLAY BEEP: GOTO 110

The next part of the program creates the mailing list file. The section begins by prompting for the number of names to be recorded in the file initially, and storing that value in record **Q**. Then the mailing information for the first name is written to record 1 (the subroutine beginning at line 81**0** is used to prompt for the mailing data.) The data for the first name is written separately from any other names to be recorded in the file because it is not necessary to sort the file for only one name. Lines 28**0** through 32**0** get the mailing information for the remaining names and write it to the file in sorted order. The subroutine beginning at line 1181 is used to insert each name in sorted order.

200 !Create a new file

210 INPUT "Number of names_µ ";NAMECOUNT

22D IF NAMECOUNT <= D THEN 21D

23D DISPLAY "Creating a new file..."

240 OPEN a2, "101. MAILFILE", UPDATE, RELATIVE, INTERNAL, VARIABLE 98

250 PRINT a2, REC 0, NAMECOUNT

26**0** GOSUB 811

27Φ PRINT α2, REC 1, LNAMES; FNAMES; ADDRESSS; CITYS; STATES; ZIP

280 FOR COUNT=2 TO NAMECOUNT

299 GOSUB 810
300 LASTREC=COUNT-1
310 GOSUB 1080
320 NEXT COUNT
330 CLOSE \(\alpha 2: \text{GOTO} \) 110

The next section adds a mailing address to the established file. Lines 360 and 370 open the file and get the number of records currently stored in the file. Line 380 calls the subroutine at line 810 to prompt for the information to add to the file. Line 390 calls the subroutine at 960 to determine if the name is already stored in the file. If the name does not match any names already in the file, the name is added to the file, using the subroutine at line 1080 to add the record in sorted order.

34D !Add to the file

350 DISPLAY "Add name to file"

36 D OPEN α2, "101. MAILFILE", UPDATE, RELATIVE, INTERNAL, VARIABLE 98

37D INPUT a2, REC 1, NAMECOUNT

38P GOSUB 810

39D GOSUB 96D

400 IF MATCH8="YES" THEN 49€

41D LASTREC=NAMECOUNT

42 DISPLAY "Updating file..."

43 O GOSUB 1080

44D NAMECOUNT=NAMECOUNT+1

450 PRINT @2, REC O, NAMECOUNT

46**D** INPUT "Add another name_μ (Y/N) ";ANδ

470 IF ANS="Y" OR ANS="y" THEN 380

480 CLOSE @2:GOTO 110

490 DISPLAY BEEP, "Name already in file!":PAUSE

500 GOTO 480

Lines 510 through 630 search the file for a specific name. If the name is found, the mailing information is displayed. If not, the message No match found: is displayed. The subroutine at line 960 is used to determine if a matching name exists in the file.

- 510 !Find a name
- 52D DISPLAY "Find a name"
- 53Φ OPEN α2,"1Φ1.MAILFILE", INPUT, RELATIVE, INTERNAL, VARIABLE 98
- 54D INPUT a2, REC 4, NAMECOUNT
- 55₽ INPUT "Last nameu ";LNAME&
- 56**0** INPUT "First name_μ ";FNAME&
- 57 P GOSUB 96 D
- 580 IF MATCH8="NO" THEN 620
- 590 DISPLAY L8;", ";F8;" ";A8:PAUSE
- 600 DISPLAY C8;", ";S8;" ";Z:PAUSE
- 610 CLOSE @2:GOTO 110
- 620 DISPLAY BEEP, "No match found:":PAUSE
- 63D GOTO 61D

Lines 64D through 73D read and display the information in the file. The <u>ENTER</u> key must be pressed to display the next line of information.

- 640 !Show file data
- 650 DISPLAY "Show file data"
- 66Φ OPEN α2,"1Φ1. MAILFILE", INPUT, RELATIVE, INTERNAL, VARIABLE 98
- 670 INPUT a2, REC 0, NAMECOUNT
- 68D FOR COUNT=1 TO NAMECOUNT
- 69Φ INPUT α2, LNAMES, FNAMES, ADDRESSS, CITYS, STATES, ZIP
- 700 DISPLAY LNAMES;", ";FNAMES;" ";ADDRESSS:PAUSE
- 710 DISPLAY CITY8;", ";STATE8;" ";ZIP:PAUSE
- 720 NEXT COUNT
- 730 CLOSE @2:GOTO 110

The next section deletes the file from the diskette.

- 74**D**:Delete the file
- 750 DISPLAY "Deleting data file..."
- 76D DELETE "101.MAILFILE"
- 770 GOTO 110

The final section ends the program.

78**D**:End program

790 END

The subroutines begin with line 800. The first subroutine prompts for the data to be stored in the file and ensures that it does not exceed the length allocated for each data value. The record length of 98 bytes is allocated as follows: 31 bytes for the last name, 16 bytes for the first name, 23 bytes for the street address, 16 bytes for the city, 3 bytes for the state, and 9 bytes for the zip code. (Remember, each string value in INTERNAL format requires one extra byte for a length byte.)

8DD !**** SUBROUTINES *****

810 :Input data subroutine

820 DISPLAY ERASE ALL, "Last nameu";

830 ACCEPT AT(12) SIZE(30) VALIDATE(ALPHA), LNAMES

84D DISPLAY ERASE ALL, "First nameu";

850 ACCEPT AT(13) SIZE(15) VALIDATE(ALPHA), FNAME 8

860 DISPLAY ERASE ALL, "Street Address";

870 ACCEPT AT(17) SIZE(22) VALIDATE(ALPHANUM), ADDRESSS

880 DISPLAY ERASE ALL, "Cityu";

89♠ ACCEPT AT(7) SIZE(15) VALIDATE(ALPHANUM), CITY\$

900 DISPLAY ERASE ALL, "State abbreviationu";

910 ACCEPT AT(21) SIZE(2) VALIDATE(ALPHA), STATE8

92**D** DISPLAY ERASE ALL, "Zip code_μ";

930 ACCEPT AT(11) SIZE(5) VALIDATE(DIGIT), ZIP

940 DISPLAY ERASE ALL

95D RETURN

The next subroutine searches the file for a record matching the values entered for LNAMES and FNAMES. Because the file is sorted, a technique known as a binary search can be used to search the file. In a binary search, the body of data to be searched is divided in half and a comparison is made to determine if the matching value would be located in the upper or lower half of the data. The half containing the matching value is then itself divided in half,

and another comparison is made to determine which part would contain the matching value. This process of dividing the data in half is repeated until a matching value is found or it is determined that no matching value exists. A binary search is normally faster than a sequential search from the beginning of a file.

960 :Search file for duplicate record

970 HI=NAMECOUNT+1

98D LO=1

99**D** MID=INT((HI+LO)/2)

1000 LASTMID=MID

1010 INPUT a2, REC MID, L8, F8, A8, C8, S8, Z

1020 IF (L80F8)=(LNAME80FNAME8) THEN MATCH8="YES":RETURN

1030 IF (L80F8) < (LNAME 80 FNAME 8) THEN LO=MID ELSE HI=MID

104D MID=INT((HI+LO)/2)

1050 IF LASTMID=MID THEN MATCH8="NO":RETURN

106D LASTMID=MID

1070 GOTO 1010

The final subroutine adds a new name in sorted order. The subroutine begins by reading the last record in the file and comparing the name in that record with the name to be added to the file. If the name in the last record of the file is greater than the name to be added, that record is moved to the next larger record position (increasing the file size by one record). This process is repeated until a record is found whose name is not greater than the name to be added. The new mailing information is written to the empty "slot" that was created by the search (and move) procedure.

1080 :Add record in sorted order

109D FOR C=LASTREC TO 1 STEP -1

1100 INPUT a2, REC C, L8, F8, A8, C8, S8, Z

111D IF (LSOFS)<(LNAMESOFNAMES) THEN 115D

112**D** PRINT α2, REC C+1, L8, F8, A8, C8, S8, Z

113**D** NEXT C

114D C=D

1150 PRINT a2, REC C+1, LNAMES, FNAMES, ADDRESSS, CITYS, STATES, ZIP

116D RETURN

Reading a Diskette's Catalog

During the initialization process, the computer reserves the first 2 sectors on the diskette for its catalog. These sectors are used to store information about the diskette such as the diskette's name and the number of used and available sectors. Each time a new file is stored on the diskette, the computer increases the catalog size by one sector. (The additional sector is used to record information about the file such as its name and location on the diskette.)

A diskette's catalog can be accessed as a relative file if the <u>filename</u> in the <u>device.filename</u> parameter of the OPEN statement is omitted. Thus, specifying only the <u>device</u> in an OPEN statement is interpreted by the computer as referring to the diskette's catalog. For a Home Computer, the <u>device</u> can be specified by the letters DSK and a drive number, or by the word HEXBUS, a period, and a number from 111 through 114, where the last digit corresponds to the drive number. For the Compact Computer, the <u>device</u> is a number from 101 through 104, where the last digit corresponds to the drive number.

The catalog can be opened only in INPUT mode, and must be accessed as a RELATIVE file in INTERNAL format with 38-byte fixed-length records. The following are examples of valid OPEN statements for reading the catalog of a diskette in drive 1.

Examples—Home Computer

OPEN a7: "HEXBUS. 101.", INPUT, RELATIVE, INTERNAL

OPEN @2: "DSK1.", INPUT, RELATIVE, INTERNAL

Example—Compact Computer

OPEN a9," 101.", INPUT, RELATIVE, INTERNAL

The capability to open the catalog as a relative file enables Home Computer owners to read and display a catalog's contents without having to use the Disk

Manager 3 cartridge. For Compact Computer owners who do not have the Disk Utilities package (sold separately by Texas Instruments), this method must be used to access catalog information.

The catalog of a diskette can consist of as many as 128 records, numbered $\mathbf{0}$ through 127. The size of the catalog depends upon the number of files stored on the diskette. The catalog of a newly initialized diskette contains one record (describing the diskette). Each time a file is stored on the diskette, the catalog is increased by one record. Each time a file is deleted from the diskette, the catalog is reduced by one record.

Record @ contains the following information about the diskette:

- :o: The name of the diskette.
- io: The record type (always a zero for record @).
- !o! The total number of sectors on the diskette.
- io: The number of available sectors on the diskette.

Records 1 through 127 are used to record information about files, one record for each file stored on the diskette. These records contain the following information:

- to: The name of the file. If the length of this field is \boldsymbol{Q} , then the last file name has been read.
- !o! A numeric code indicating the file type (the code is a negative value if the file is protected).

Code	<u>Meaning</u>
1	DISPLAY/FIXED data file
2	DISPLAY/VARIABLE data file
3	INTERNAL/FIXED data file
4	INTERNAL/VARIABLE data file
5	Home Computer BASIC program or other Home Computer
	"memory-image" data file

- io: The total number of sectors allocated for the file.
- io: The total number of bytes per record. (A type-5 file always has bytes per record, because this measurement does not relate to memory-image data.)

Note: The Home Computer stores programs as type-5 files with a record length of **Q** bytes. The Compact Computer does not assign a unique file type to programs. Instead, the Compact Computer stores programs as SEQUENTIAL data files in DISPLAY format, with the entire Compact Computer program stored as a single variable-length record. Thus, for example, a short Compact Computer program might store as a 5**Q**-byte record, whereas a long Compact Computer program might store as a 12,**000**-byte record.

READING THE CATALOG WITH A HOME COMPUTER

The following sample program illustrates how to read and display the catalog of a diskette using a Home Computer.

The program begins by clearing the screen and assigning descriptions of the five possible file types to elements of the array TYPES. The program then requests the number of the disk drive containing the diskette to be cataloged, and checks to see if a valid drive number is entered.

100 REM DISPLAY CATALOG—HOME COMPUTER VERSION

110 CALL CLEAR

12**P** DIM TYPE**8**(5)

13D TYPE\$(1)="D/F"

14D TYPE 8(2) = "D/V"

15D TYPE 8(3) = "I/F"

16D TYPE\$(4)="I/V"

17D TYPE8(5)="PGM"

18 PINPUT "DISK DRIVEμ (1-4) ":D

19**0** D=INT(D)

200 IF D<1 THEN 180

210 IF D34 THEN 180

The next four lines open the catalog as a file, read the diskette information from record ϕ , and display the data on the screen.

```
22D OPEN al: "DSK"aSTR&(D)a".", INPUT, RELATIVE, INTERNAL
23D INPUT al: NAME&, CODE, SECTORS, AVAIL
24D DISPLAY: "DRIVE"; STR&(D); "DISKNAME="; NAME&
25D DISPLAY "AVAILABLE="; STR&(AVAIL); "USED="; STR&(SECTORS-AVAIL)::
```

The remainder of the program reads the records describing each file, formats the data, and displays the information on the screen. All file data values have been read when the length of NAMES is zero.

```
26D GOSUB 44D
27D FOR LOOP=1 TO 127
280 INPUT al: NAMES, CODE, SECTORS, BYTES
29 D IF LEN(NAME 8) = D THEN 41 D
300 DISPLAY :NAMES; TAB(12); STRS(SECTORS); TAB(17); TYPES(ABS(CODE));
31D IF ABS(CODE)=5 THEN 34D
315 REM TS IN NEXT LINE IS EQUAL TO 3 SPACES
32D T8=" "aSTR8(BYTES)
330 DISPLAY SEG8(T8, LEN(T8)-4,5);
34P IF CODE & D THEN 36D
35p DISPLAY TAB(26);"Y";
36D IF LOOP/17 &INT(LOOP/17) THEN 400
37D DISPLAY ::
38D INPUT "PRESS ENTER TO CONTINUE: ": ANS
39P GOSUB 43D
400 NEXT LOOP
410 CLOSE al
42P END
43D DISPLAY
435 REM TYPE 3 SPACES BETWEEN FILENAME AND SIZE
440 DISPLAY "FILENAME SIZE TYPE LRL P"
45D DISPLAY "------------;
```

460 RETURN

To use the catalog program, RUN it and enter the number of the disk drive containing the diskette to be cataloged when the prompt DISK DRIVE $_{\mu}$ (1-4) appears. The catalog information is displayed 17 lines at a time.

i (M.). Na 1980 na mangana na kabangsan kabangsan kabangsan kabangsan kabangsan kabangsan kabangsan kabangsan kabangsa

READING THE CATALOG WITH A COMPACT COMPUTER

The following program illustrates how to read and display the catalog of a diskette using a Compact Computer.

The program begins by assigning descriptions of the five possible file types to elements of the array TYPE. The program then requests the number of the disk drive containing the diskette to be cataloged. (Drive 1 can be selected by simply pressing ENTER.)

100 !Display Catalog--Compact Computer Version

11**D** DIM TYPE8(5)

12**D** TYPE\$(1)="Dis/Fix"

13**D** TYPE\$(2)="Dis/Var"

14**0** TYPE\$(3)="Int/Fix"

15D TYPE\$(4)="Int/Var"

16D TYPE\$(5)="HC Program"

17D DISPLAY "Disk Drive_μ (1-4) 1";

18 D ACCEPT AT(19) SIZE(-1) VALIDATE("1234"), D8

The next six lines open the catalog as a file, read the diskette information from record $\boldsymbol{\mathcal{O}}$, convert it from Home Computer format to Compact Computer format, and display the data. Because all numbers in the catalog are written in Home Computer INTERNAL format, it is necessary to read them as strings and use the CONVERT subprogram to convert them to numbers.

```
19D OPEN al, "10" aDSa". ", INPUT, RELATIVE, INTERNAL
2DD INPUT al, NAMES, CS, SS, AS
21D CALL CONVERT(SS, SECTORS)
22D CALL CONVERT(AS, AVAIL)
23D DISPLAY ERASE ALL, "Disk="; NAMES;" A="; STRS(AVAIL);" U="; STRS(SECTORS-AVAIL)
24D ANS=KEYS: IF ANS="" THEN 24D
```

The next section of the program reads the information describing each file, converts it to Compact Computer format, and displays the data. The program displays the filename, file type, number of sectors used by the file, and the protection status, if any, for the file. After displaying the data for each file, the program checks the keyboard to determine whether to display the data for the next file, the previous file, or to stop the program (lines 350-370). To display the information for the next file, press the DOWN ARROW key. To display the information for the previous file, press the UP ARROW key. Pressing any other key stops the program and ends the catalog listing.

```
25D IF R 1 THEN R=1
260 DISPLAY ERASE ALL
27 INPUT al, REC R, NAMES, CS, SS, BS
28DIF LEN(NAME&)=DTHEN DISPLAY "Catalog Done:":R=R-1:GOTO 36D
2910 CALL CONVERT(C8, CODE)
3 CALL CONVERT(S8, SECTORS)
310 CALL CONVERT(BS, BYTES)
32D DISPLAY NAMES;",";TYPES(ABS(CODE));
33♥ IF ABS(CODE) &5 THEN DISPLAY " ";STR%(BYTES);
34 DISPLAY ",";STR (SECTORS);
350 IF CODE D THEN DISPLAY ",P" ELSE DISPLAY ""
36 D ANS=KEYS
37 ► IF ANS=CHRS(233) THEN R=R+1:GOTO 26 ►
38D IF ANS=CHRS(232) THEN R=R-1:GOTO 25D
39D CLOSE al
400 END
```

The last part of the program is a subprogram to convert a string value from Home Computer INTERNAL numeric format to a Compact Computer numeric value. This subprogram is necessary because the two computers use different formats for storing numbers. The Disk Drive/Controller always writes numbers in the catalog in Home Computer INTERNAL format.

```
410 SUB CONVERT (X8, VALUE)
420 DIM X(8)
43D VALUE=D
44P FOR C=1 TO 8
45\mathbf{O} \times (C) = ASC(SEG8(X8,C,1))
46♥ NEXT C
470 VSIGN=1
48\mathbf{P} IF X(1)=\mathbf{D} OR X(2)=\mathbf{D} THEN SUBEXIT
48P IF X(1)<128 THEN 53P
49© VSIGN=-1
5DD T=-(X(1)*256+X(2)-65536)
51\Phi X(1)=INT(T/256)
52D X(2)=T-X(1)*256
53P FOR C=2 TO 8
54D VALUE=VALUE*111+X(C)
55D NEXT C
56 \mathbf{P} VALUE=VALUE*VSIGN*1\mathbf{QQ}_{\mathbf{U}}(X(1)-7\mathbf{Q})
57D SUBEND
```

(**Note to typesetter: u = exponentiation)

APPENDIX A HOME COMPUTER I/O ERROR CODES

The non-disk-related BASIC error codes for the Home Computer are given in the owner's manual supplied with the computers. The following codes are for errors that relate to the $HEX-BUS^{TM}$ Disk Drive/Controller.

Error codes for the Home Computer are two-digit numbers. The first digit indicates the command or statement involved in the error. The second digit indicates the type of error.

<u>First</u>	
<u>Digit</u>	Command or Statement
Ø	OPEN
1	CLOSE
2	INPUT
3	PRINT
4	RESTORE
5	OLD
6	SAVE
7	DELETE
9	EOF

Second	
Digit	Type of Error
0	DEVICE NOT FOUND-The specified disk drive could not be
	accessed. Check that the disk drive is turned on and
	correctly connected to the computer.
1	DISK or FILE WRITE-PROTECTEDThe diskette is
	write-protected, or the file is assigned a protected
	status.
2	BAD OPEN or HEXBUS ATTRIBUTE—One or more OPEN options
	were invalid or didn't match the file's actual
	characteristics. The device name used with HEXBUS was
	invalid (for example using SAVE HEXBUS.DSK1.filename
	instead of SAVE HEXBUS.101.filename).

- ILLEGAL OPERATION—Should not be generated by the disk system. It can be caused on other devices, however, by attempting to perform an illegal file operation such as requesting INPUT from a printer.
- OUT OF SPACE—The diskette is full, or you are trying to open more than 4 files at one time.
- 5 ATTEMPT TO READ PAST END OF FILE—An input statement has been executed after the end of a file has been reached.
- DEVICE ERROR—The disk drive is not connected or turned on, the engage/release lever on the drive is not closed, the diskette is damaged or not initialized, or the disk drive is not functioning properly.
- FILE ERROR—An attempt was made to write to or read from a file that is not open, an attempt was made to open a file that is already open, the indicated file or diskette doesn't exist, or you are trying to read a BASIC program file as if it were a data file.

APPENDIX B DISK MANAGER 3 ERROR CODES

The following list details the Disk Manager 3 error codes and messages.

CODE DEFINITION

DISK NOT INITIALIZED (No code is associated with this error.)

!o! This error is generated by the Disk Manager 3 cartridge when the Disk Controller is unable to read any data from a diskette. The error can be caused by an uninitialized or damaged diskette, the lack of a diskette in the disk drive, or an inproperly inserted diskette.

O DEVICE NOT FOUND ACTIVE

!o! The <u>HEX-BUS</u> Interface peripheral is turned off or does not have power.

1 FILE/DISK WRITE PROTECTED

- :o: Attempted to catalog or log errors to a write-protected diskette.
- !o! Attempted to backup, copy, rename, modify protection status, or delete a file from a diskette that is write-protected.
- !o! Attempted to copy, rename, modify protection status, or delete a write-protected file.
- :o! Diskette is not correctly inserted in the disk drive.

2 INVALID OPTION SPECIFIED

- to! Incorrect or invalid option specified in device.filename.
- to: Filename too long or missing in device.filename.

4 OUT OF SPACE

- !o! Attempted to copy a file to a diskette whose catalog is full. (The catalog of a diskette can contain a maximum of 127 files.)
- io: No available space on a diskette.

- 6 DEVICE ERROR
 - :o! A failure has occurred in the Disk Drive/Controller peripheral. This error can occur when catalog information was lost, the peripheral detected a transmission error or a diskette failure, etc.
 - !o! Attempted to access disk drive 2, 3, or 4 when a disk drive 2, 3, or 4 is not connected to the Disk Drive/Controller.
- 7 NONEXISTENT FILE OR DISK
 !o! The specified file does not exist.
- NO DISK OR NO DRIVE

 !o! Attempted to access disk drive 2, 3, or 4 when a disk drive 2, 3, or 4 is not connected to the Disk Drive/Controller.

- 17 INVALID INPUT PARAMETERS
 - that 35, 40, or 77.
- 21 SEEK ERROR

to! The Disk Controller has detected a read/write head seek error.

- 31 RECORD NOT FOUND ON INPUT
 - :o: The Disk Controller has detected a missing sector address mark.
 - !o! The disk drive does not contain a diskette, or the diskette is damaged or not initialized.
 - !o! The engage/release lever on the drive is not closed.
- 32 CRC ERROR ON INPUT
 - !o! The Disk Controller has detected a CRC (Cyclic Redundancy Check)
 error.
- 33 LOST DATA ON INPUT
 - :o! The Disk Controller has detected a device error during a read operation.
- 38 BAD COMPARE ON WRITE
 - !o! The Disk Controller has detected a device error during a write operation.
- 41 RECORD NOT FOUND ON PRINT
 - :o: The Disk Controller has detected a device error during a write operation.
- 42 CRC ERROR ON PRINT
 - !o! The Disk Controller has detected a CRC (Cyclic Redundancy Check) error during a write operation.
- 43 LOST DATA ON PRINT
 - :o: The Disk Controller has detected a device error during a write operation.

- 44 DISK WRITE PROTECTED
 - !o! Attempted to FORMAT or modify the name of a write-protected diskette.
 - io: Diskette is not correctly inserted in the disk drive.
- 51 MISSING ADDRESS MARK
 - io: The Disk Controller has detected a missing sector address mark.
- 60 BUS TRANSMISSION ERROR
 - !o! Lost communication with the specified device.
 - !o! Specified device is not connected to the I/O bus.
- 61 NO DATA (SECTOR NOT FOUND)
 - :o: The Disk Controller has detected a missing sector identification mark.
- 99 COMPREHENSIVE TEST ERROR
 - :o: Comprehensive test data written during test 5 could not be verified during test 6.
 - DISK NOT INITIALIZED (There is no code associated with this error.)
 - !o! This error is generated by the Disk Manager 3 cartridge when the Disk Controller is unable to read any data from a diskette. The error can be caused by an uninitialized or damaged diskette, the lack of a diskette in the disk drive, or an inproperly installed diskette.

APPENDIX C COMPACT COMPUTER 40 I/O ERROR CODES

The following list details the Compact Computer 4D input/output (I/O) error codes that relate to the HEX-BUSTM Disk Drive/Controller.

Compact Computer 4D I/O errors are displayed in one of the following forms.

!o! I/O error ccc afff

to: I/O error ccc "ddd"

 \underline{ccc} is one of the I/O error codes listed below, \underline{fff} is the file number assigned in the OPEN statement, and \underline{ddd} is the device number used in the OPEN statement.

CODE DEFINITION

- 1 DEVICE/FILE OPTIONS ERROR
 - to: Incorrect or invalid option specified in device.filename.
 - io: Filename too long or missing in device.filename.
- 2 ERROR IN ATTRIBUTES
 - !o! An OPEN statement specified incorrect attributes (<u>file-type</u>, file-organization, <u>open-mode</u>) for an existing file.
- 3 FILE NOT FOUND
 - !o! The file specified in one of the following operations does not exist.

OPEN statement using the INPUT attribute

OLD "device.filename"

RUN "device.filename"

CALL LOAD("device.filename")

- 4 DEVICE/FILE NOT OPEN
 - io: Attempted to access a closed file with a CLOSE, INPUT α , LINPUT α , PRINT α or RESTORE α operation.
 - :o: File specified in EOF function is closed.

5 DEVICE/FILE ALREADY OPEN

- :o: Attempted to OPEN or DELETE an open file.
- :o: Attempted to FORMAT a diskette while a diskette file is open (the open file need not be on the same diskette referenced by the format command).

6 DEVICE ERROR

!o! A failure has occurred in the Disk Drive/Controller peripheral. This error can occur when catalog information was lost, the peripheral detected a transmission error or a diskette failure, etc.

7 END OF FILE

!o! Attempted to read past the end of the file.

8 DATA/FILE TOO LONG

- :o: Attempted to output a record that was longer than the capacity of the device.
- !o! A file exceeded the maximum file length for a device.
- io: The data length field in a CALL I/O command was too large.

9 WRITE PROTECT ERROR

- !o: Attempted to FORMAT a write-protected diskette.
- !o! Attempted to OPEN a write-protected file in OUTPUT, APPEND, or UPDATE mode.
- !o! Attempted to DELETE a file from a write-protected diskette.
- :o: Attempted to execute a CALL I/O write command to a write-protected diskette.
- :o: Attempted to SAVE a program to a write-protected diskette.
- :o! Diskette is not correctly inserted in the disk drive.

NOT REQUESTING SERVICE

!o: Response to a service request poll when the specified device did not request service. (This code is used in special applications and should not be encountered during normal execution of BASIC programs.)

11 DIRECTORY (CATALOG) FULL

:o: Attempted to OPEN a new file on a diskette whose catalog is full.

(The catalog of a diskette can contain a maximum of 127 files.)

12 BUFFER SIZE ERROR

- !o! When an existing file was opened for INPUT, UPDATE, or APPEND, the specified record length (VARIABLE \underline{XXX}) was different than the allowed length of the largest record in the existing file.
- !o! The VERIFY command found the program in memory was smaller than the program on the diskette.

13 UNSUPPORTED COMMAND

- !o! Attempted an operation not supported by the peripheral. (This code is used in special applications and should not be encountered during normal execution of BASIC programs.)
- DEVICE/FILE NOT OPENED FOR OUTPUT

 !o! Attempted to write to a file or device opened for input.
- DEVICE/FILE NOT OPENED FOR INPUT

 !o! Attempted to read from a file or device opened for output or append.
- 16 CHECKSUM ERROR

 !o! The checksum calculated on the input record was incorrect.
- 17 FILE ORGANIZATION INCORRECT
 - !o! Attempted to open for relative access a file that was created as a sequential file with variable-length records.

23 FILE TYPE ERROR

!o! When an existing file was opened, DISPLAY format was specified for a file in INTERNAL format, or INTERNAL format was specified for a file in DISPLAY format.

24 VERIFY ERROR

26 UNINITIALIZED MEDIUM

- :o: Attempted to open a file on uninitialized diskette.
- :o: Attempted to open a file on diskette that has been accidentally erased or destroyed.

31 INVALID FILENAME

!o! Filename was missing, too long, or contained periods or embedded blanks.

32 MEDIUM FULL

io! No available space on a diskette.

33 ATTEMPT TO OPEN TOO MANY FILES

!o! Attempted to open more than 4 diskette files at one time.

34 INVALID DATA OR DATA LENGTH

- !o! The data field of a CALL I/O command contains invalid values.
- :o: The data length of a CALL 1/0 command does not meet the requirements of the command.

35 FILENAME ALREADY EXISTS

!o! Attempted to create a new file on a diskette with a name that was identical to an existing file on the diskette.

97 NO DISKETTE OR NO DRIVE

!o! Attempted to access disk drive 2, 3, or 4 when a disk drive 2, 3, or 4 is not connected to the Disk Drive/Controller.

98 INVALID INPUT PARAMETERS

!o! A CALL I/O command has transmitted an invalid sector number.

- 99 SEEK ERROR
 - !o! The Disk Controller has detected a read/write head seek error.
- 111 MISSING ADDRESS MARK
 - to: The Disk Controller has detected a missing sector address mark.
- NO DATA (SECTOR NOT FOUND)
 - :o: The Disk Controller has detected a missing sector identification mark.
- 112 RECORD NOT FOUND ON READ
 - !o! The disk drive does not contain a diskette, or the diskette is damaged or not initialized.
 - :o: The engage/release lever on the drive is not closed.
- 113 CRC ERROR ON INPUT
 - !o: The Disk Controller has detected a CRC (Cyclic Redundancy Check)
 error.
- 114 LOST DATA ON INPUT
 - !o! The Disk Controller has detected a device error during a read operation.
- 115 BAD COMPARE ON WRITE
 - !o! The Disk Controller has detected a device error during a write operation.
- 116 RECORD NOT FOUND ON WRITE
 - :o: The Disk Controller has detected a device error during a write operation.
- 117 LOST DATA ON WRITE
 - !o! The Disk Controller has detected a device error during a write operation.

254 ILLEGAL IN SLAVE MODE

- :o: Attempted a normal (master) I/O bus operation while the computer was in peripheral (slave) mode. (This error occurs during some special applications and should not be encountered during normal execution of a BASIC program.)
- !o! NOTE: Improper modification of memory by the POKE, RELMEM, EXEC, or DEBUG subprograms can result in the computer being placed in peripheral (slave) mode.

255 TIME-OUT ERROR

- !o! Lost communication with the specified device.
- :o: Specified device is not connected to the I/O bus.

APPENDIX D HOME COMPUTER DISKETTE/CASSETTE OPERATIONS

TRANSFERRING PROGRAMS BETWEEN CASSETTES AND DISKETTES

To transfer a program from a cassette tape to a diskette, load the program from the cassette into the computer's memory using the OLD command. Then save the program to the diskette using the SAVE command. To transfer a program from a diskette to a cassette, load the program from the diskette and save it to the cassette tape.

For example, the following sequence transfers a program from a cassette to the diskette in disk drive 1. The program is stored on the diskette under the name PROGRAM1.

OLD CS1
(follow normal cassette instructions)
SAVE DSK1.PROGRAM1

The following sequence transfers a program named PGM9 from the diskette in disk drive 1 to a cassette tape.

OLD DSK1.PGM9
SAVE CS1
(follow normal cassette instructions)

TRANSFERRING DATA FILES FROM CASSETTE TO DISKETTE

To transfer data from a cassette to a diskette, read the data from the cassette file and write it to a diskette file.

The following program illustrates transferring data from a cassette to the diskette in drive 1. To successfully transfer data between the two devices, you must know the parameters of the OPEN statement used to create the file, and the pattern of the data in the file. In this example, the cassette file

is a SEQUENTIAL file (required for cassette files) in DISPLAY format (the default) with FIXED records (required for cassette files) with a length of 128 bytes. Each record contains two string values and two numeric values, in that order.

100 OPEN al: "CS1", SEQUENTIAL, INPUT, FIXED 128

11D OPEN a2: "DSK1.DATA1", SEQUENTIAL, OUTPUT, FIXED 128

120 INPUT al:S18,S28,N1,N2

130 PRINT @2:S18;S28;N1;N2

14D GOTO 12D

INPORTANT: The EOF function cannot be used with cassette files. Therefore, the program does not have a method of testing when the last data value has been read. When the program attempts to input data past the end of the file, the program will end with an I/O error. When this error occurs, enter the commands

CLOSE al

CLOSE a2

to close the data files and ensure that no information is lost. If you fail to close the files, data may be lost from the diskette file.

APPENDIX E TABLE OF DISKETTE STORAGE CAPACITIES

The number of sectors placed on a diskette during initialization depends upon the storage format selected for the diskette. The TI Disk Drive/Controller supports the following diskette storage formats:

!o! Single-sided single-density (SS/SD)

:o! Single-sided double density (SS/DD)

:o: Double-sided single-density (DS/SD)

:o: Double-sided double density (DS/DD)

The number of sectors created by each format is shown below.

					<u>Diskette Format</u>		
				SS/SD	· SS/DD	DS/SD	DS/DD
Track	Count	4 0	1	358	638	718	1278

As part of the initialization process, the Disk Drive/Controller checks that data can be reliably written to and read from each sector. Sectors that are found to be unreliable are marked "used" in the diskette's catalog so that they will not be used to store files. Unusable sectors are usually caused by flaws in a diskette. The sum of the number of "available" and "used" sectors will match the value given in the table above.

MAINTENANCE AND SERVICE INFORMATION

In Case of Difficulty

If you have difficulty with your disk system, the following instructions will help you analyze the problem. If the suggested remedies are not successful, contact the Consumer Relations Department by mail or telephone (refer to "If You Have Questions or Need Assistance" later in this Appendix).

- Power—Be sure all devices are plugged in. Then turn the system on, beginning with the peripheral at the outermost end of the HEX-BUSTM chain. The last unit you should turn on is the computer. All units must be turned on for proper operation.
- Cables—Check that the proper cables are being used. Check for loose or broken leads and connectors. Be sure that the cables are plugged in securely. Be sure the <u>HEX-BUS</u> cables are plugged in on both rows of pins and that the raised tab is on top.
- 3. <u>Software—Be</u> sure all commands and statements are used as described in this manual or in the manual supplied with the computer. If the disk system works properly with the Disk Manager 3 cartridge but not with a program, the problem is probably with the program. Especially check the use of OPEN, INPUT a, and PRINT a statements.
- 4. <u>Computer</u>—Disconnect all peripherals and check to see that the computer works properly with no peripherals attached. The carefully reconnect each peripheral and check its operation.
- 5. <u>Diskettes</u>—Be sure that diskettes are initialized and inserted with the label facing up. Be sure that the diskettes that you are using are "soft-sectored." Soft-sectored diskettes have only one index hole in the disk film. Hard-sectored diskettes have multiple index holes in the disk film.

6. <u>Diagnostic Tests</u>—Home Computer owners can ensure that the diskette, controller, and disk drives are working properly by running the diagnostic tests in the Disk Manager 3 cartridge.

Service Centers

If your Disk Drive/Controller peripheral requires service, instead of returning the unit to a service facility for repair or replacement, you may elect to exchange the unit for a factory-reconditioned Disk Drive/Controller of the same model (or equivalent model specified by TI) by going in person to one of the service centers which have been established across the United States. A handling fee will be charged by the service center for in-warranty exchanges of the Disk Drive/Controller peripheral. Out-of-warranty exchanges will be charged at the rates in effect at the time of the exchange.

To determine if there is a service center in your area, look for Texas Instruments Service Center in the white pages of your telephone directory, or look under one of the following two headings in the yellow pages: "Calculator and Adding Machines" or "Computers—Service and Repair." Please call the service center for availability and exchange fee information. Write the Consumer Relations Department for further details and the location of the nearest service center.

If You Have Questions or Need Assistance

If you have questions concerning Disk Drive/Controller repair, please call our Customer Relations Department at (800) 858-4565 (toll free within the contiguous United States). If you wish to make peripheral or software purchases and are unable to find these items locally, call our Software Sales Department toll-free at (800) 858-4075. The operators at this number cannot provide technical assistance.

For technical questions, such as programming information, specific applications, etc., you can call (806) 741-2663. Please note that this is not a toll-free number, and collect calls cannot be accepted.

As an alternative, you can write to:

Consumer Relations Department
Texas Instruments Incorporated
P.O. Box 53
Lubbock, Texas 794**0**8

Because of the number of suggestions which come to Texas Instruments from many sources containing both new and old ideas, Texas Instruments will consider such suggestions only if they are freely given to Texas Instruments. It is the policy of Texas Instruments to refuse to receive any suggestions in confidence. Therefore, if you wish to share your suggestions with Texas Instruments, or if you wish us to review any programs that you have developed, please include the following statement in your letter:

"All of the information forwarded herewith is presented to Texas Instruments on a nonconfidential, nonobligatory basis; no relationship, confidential or otherwise, expressed or implied, is established with Texas Instruments by this presentation. Texas Instruments may use, copyright, distribute, publish, reproduce, or dispose of the information in any way without compensation to me."

WARRANTY INFORMATION

THREE-MONTH LIMITED WARRANTY

This Texas Instruments Disk Drive/Controller peripheral warranty extends to the original consumer purchaser of the accessory.

Warranty Duration

This Disk Drive/Controller peripheral is warranted for a period of three (3) months from the date of the original purchase by the consumer.

Warranty Coverage

This Disk Drive/Controller peripheral is warranted against defective materials or workmanship. This warranty is void if the accessory has been damaged by accident, unreasonable use, neglect, improper service or other causes not arising out of defects in materials or workmanship.

Warranty Disclaimers

Any implied warranties arising out of this sale, including but not limited to the implied warranties of merchantability and fitness for a particular purpose, are limited in duration to the above three-month period. Texas Instruments shall not be liable for loss of use of the hardware or other incidental or consequential costs, expenses, or damages incurred by the consumer or any other user.

Some states do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you in those states.

Legal Remedies

This warranty gives you specific legal rights, and you may also have other rights that vary from state to state.

Warranty Performance

During the above three-month warranty period, your Disk Drive/Controller peripheral will be repaired or replaced with a new or reconditioned unit of the same or equivalent model (at TI's option) when the unit is returned by prepaid shipment to a Texas Instruments Service Facility listed below. The repaired or replacement unit will be warranted for three months from date of repair or replacement. Other than the postage requirement, no charge will be made for the repair or replacement of in-warranty units.

Texas Instruments strongly recommends that you insure the unit for value prior to shipment.

Texas Instruments consumer service facilities

U.S. Residents

Texas Instruments Service Facility 23**0**3 North University Lubbock, Texas 79415

Canadian Residents

Geophysical Services Incorporated
41 Shelley Road
Richmond Hill, Ontario, Canada L4C5G4

Consumers in California and Oregon may contact the following Texas Instruments offices for additional assistance or information:

Texas Instruments Consumer Service 831 South Douglas Street El Segundo, California 9**0**245 (213) 973-18**0**8

Texas Instruments Consumer Service 67**©D** Southwest 1**©**5th Kristin Square, Suite 11**D** Beaverton, Oregon 97**©**05 (5**D**3) 643-6758

GLOSSARY

ASCII—An acronym for American Standard Code for Information Interchange. An almost universally accepted code for representing the upper—and lower—case letters of the alphabet, the 100 digits, and the various punctuation marks and symbols as unique numbers between 1000 and 1000 and 1000 This code permits the computer to store these characters in its memory as 1000 Below Information Interchange. An almost universally accepted code for Information Interchange. An almost universally accepted code for Information Interchange. An almost universally accepted code for representing the upper—and lower—case letters of the alphabet, the 1000 digits, and the various punctuation marks and symbols as unique numbers between 1000 and 100

automatic density recognition—The capability of the Disk Drive/Controller to automatically recognize whether a diskette has been initialized for single—or double—density operation.

backup—The process of regularly making copies of data and program files stored on diskettes. The purpose of making backup copies is to provide a means of recovering information if the original copy of a file is damaged.

BASIC—An acronym for Beginner's All-purpose Symbolic Instruction Code, a very successful and popular computer language developed at Dartmouth College in 1963-64. The language is characterized by the use of English words for commands to the computer.

binary—The base—2 number system used at the most basic level of a digital computer. The only digits used in the binary number system are **Q**s and ls. These digits are arranged in positions marking ones, twos, fours, eights, and other powers of two. Binary representation is used in computers because it is convenient to represent ls and **Q**s by high and low electrical signals.

bit—A binary digit, which can be a **o** or 1. Bits are used to represent numbers in binary or base-2 notation.

<u>bus</u>—An ordered set of electical lines used by the computer to send and receive data, address, timing, and status information.

byte—An eight-bit binary number. Eight binary bits can represent any number from **©** through 255. The storage capacity of a computer or diskette is commonly measured in bytes.

catalog—An area on each diskette, used by the computer to record information about the diskette and the files stored on the diskette. (The catalog is also referred to as the directory of the diskette.)

character-One letter, digit, or symbol stored by a computer.

CRC error—An acronym for Cyclic Redundancy Check. A CRC code provides the Disk Drive/Controller with a means of checking that information read from a diskette is error—free.

data—Information, either textual or numeric. Data processing refers to the manipulation of data by a computer.

data entry--The process of entering information into a computer.

<u>default</u>—Information assumed by a computer when no specific information is entered.

density—The quantity of data that can be stored in a fixed amount of space on a diskette. Single-density diskettes initialized by the Disk Drive/Controller contain 9 sectors per track; double-density diskettes contain 16 sectors per track.

<u>device</u>—Any equipment connected to a computer that can receive data from or transmit data to the microprocessor of the computer. Examples of devices include the monitor screen, keyboard, disk drive, and printer.

<u>device name</u>—The letters or numbers used to identify a particular peripheral device.

<u>disk drive</u>—The mechanical device that rotates and magnetically reads from and writes to a diskette.

disk drive controller—The hardware interface between the computer and a disk drive that controls the operation of the disk drive.

diskette—A flexible plastic disk coated with a magnetic material that is used to store information by a computer. Also called a "floppy" diskette or "floppy" disk.

diskette jacket-The protective cover of a diskette.

double-density—A diskette storage format that places 16 sectors in each track of a diskette. (Other manufactures may place a different number of sectors on a double-density diskette.)

double-sided—A diskette storage format that permits information to be stored
on both sides of a diskette.

EOF—An acronym for End of File. The EOF function permits a BASIC program to check whether the last record of a data file has been reached.

<u>file</u>—A set of data stored as a unit on a storage medium such as a cassette tape or a diskette.

file access—The process of reading data from or writing data to a diskette.

filename—The name assigned to a file.

format—Refers to the arrangement or organization of tracks and sectors on a diskette. The index marks identifying the locations of the tracks and sectors are placed on a diskette when it is initialized. (The process of initializing a diskette is also called "formatting" a diskette.)

function keys——Special key sequences used when typing on the keyboard of a Home Computer to perform actions such as inserting and deleting characters. Refer to the owner's manual supplied with your computer for a list of the function keys.

nardware—Any of the physical components or pieces of computer equipment, including peripheral devices such as the TI Disk Drive/Controller.

<u>I/O</u>—An acronym for Input/Output. Generally refers to the exchange of data between a computer and a peripheral device.

I/O buffer—An area of computer memory used for the temporary storage of data being written to or read from a diskette. It is absolutely essential to CLOSE a data file to ensure that the contents of the I/O buffer are written to the diskette before turning the computer off (or leaving BASIC on a Home Computer).

<u>input</u>—The process of entering information into a computer, either through the keyboard or from a peripheral storage device such as the Disk Drive/Controller.

instruction——A computer language statement or command telling a computer what to do.

interface—Equipment designed to permit computer hardware to be connected together.

K—An abbreviation for kilo, meaning thousand. When used to designate memory or diskette capacity, a K is 1124 bytes of memory. The term is often loosely used to mean 1111.

<u>keyword</u>—One of the instructional words used in a high-level language such as BASIC. Examples of BASIC keywords are PRINT, END, and NEW.

memory—In a general sense, any of the many devices used to store information for a computer. In a specific sense, the random access memory (RAM) or read only memory (ROM) of the computer.

menu--A list of options available in a program.

output-Information produced by a computer-often on a monitor screen, printer, or diskette.

<u>peripheral</u>—Any accessory device that is not a basic part of the computer. Examples of peripherals are disk drives, printers, and modems. protected file--A file marked so that it cannot be modified or erased.

read--The process of retrieving information stored on a diskette.

read/write window—The oblong slot in the diskette jacket that allows a disk drive to access information on a diskette.

record——A unit of information stored in a data file. In a sequential file, records can be variable length or fixed length. Records in a relative file must be fixed length.

relative file access—One of two types of file access. The records in a relative file can be read from or written to in any order.

run-The process of starting a program.

sector—The smallest of two index units used to organize information stored on a diskette. Sectors are subdivisions of tracks. Each sector can store up to 256 bytes (characters) of data. The information that identifies the location of the sectors is placed on the diskette when it is initialized.

<u>seek</u>—The term used to describe the action of directing the read/write head of the Disk Drive/Controller to a specific track on a diskette.

sequential file access—One of two types of file access. The records in a sequential file must always be read from or written to in sequence, generally beginning with the first record in the file.

<u>single-density</u>—A diskette storage format that places 9 sectors in each track of a diskette. (Other manufactures may place a different number of sectors on a single-density diskette.)

<u>single-sided</u>—A diskette storage format that permits information to be stored on only one side of a diskette.

soft-sectored—One of two types of diskettes. A soft-sectored diskette has only one timing hole in the diskette (at the beginning of the first sector of a track.) A hard-sectored diskette has a timing hole at the beginning of every sector of a track. The TI Disk Drive/Controller requires soft-sectored diskettes.

software—A program that can be run on a computer.

track—The largest of two index units used to organize information stored on a diskette. Tracks are arranged in concentric circles and contain smaller index units known as sectors. The information that identifies the location of each track is placed on the diskette when it is initialized. The double-sided storage format used by the TI Disk Drive/Controller places 41 tracks on each side of a diskette. The single-sided storage format places 41 tracks on only one side of a diskette.

<u>verify</u>—The process of comparing data stored on a diskette with data stored in memory to ensure that no differences exist.

write--Used to refer to the process of recording information on a diskette.

write-protect notch—A 1/4-inch notch in the upper-right edge of the jacket of a diskette. Covering the write-protect notch with a piece of tape physically instructs the Disk Drive/Controller to reject any command to write to the diskette. The purpose of write-protecting a diskette is to prevent any loss of data on the diskette by commands that write over or modify files.