CBM™

CBM™

CBM™

CBM™

FLOPPY DISK

FLOPPY DISK

FLOPPY DISK

FLOPPY DISK

# CBM FLOPPY DISK USER MANUAL

MODEL 2040

# COMMODORE CBM DUAL DRIVE FLOPPY DISK MODEL 2040

# USER MANUAL

P/N 320830-4

**Commodore Business Machines**
3330 Scott Boulevard
Santa Clara, California 95050

# Table Of Contents

Section 1

# GENERAL INFORMATION

With the purchase of your Commodore 2040 Dual Drive Floppy Disk you have greatly enhanced the computing power of your Commodore CBM 2001 Series computer. This User Manual contains the information you need to connect, check out, and operate your Dual Drive Floppy Disk. To get the most out of your system, you should also consult your computer's User Manual, and if necessary, one of the BASIC manuals listed in Appendix A.

For ease of reference, the Commodore 2040 Dual Drive Floppy Disk will be referred to in this manual as "the 2040."

## DESCRIPTION

The 2040, shown in Figure 1, is a dual-drive diskette storage device. It consists of read/write control, drive motor electronics, two drive mechanisms, two read/write heads, track positioning mechanisms, and removable diskettes. The 2040 conforms to IEEE-488 interface requirements. Because your 2040 is an "intelligent" peripheral, it requires no space in your computer's memory. This means you have just as much computer memory available to you as when you do not have the 2040 attached.

### Front Panel

The 2040's front panel (Figure 2) consists of an identification panel across the top; slots in which to insert two diskettes; and doors to close after inserting the diskettes. When you close the door, the diskette is clamped onto the diskette spindle hub. Also on the front panels are three LED indicator lights. The one on the right is called the Drive Ø Active Indicator, and it lights when Drive Ø is active. The LED on the left does the same for Drive 1. The indicator in the middle lights whenever a disk error occurs.

Figure 1. Model 2040 Dual Drive Floppy Disk

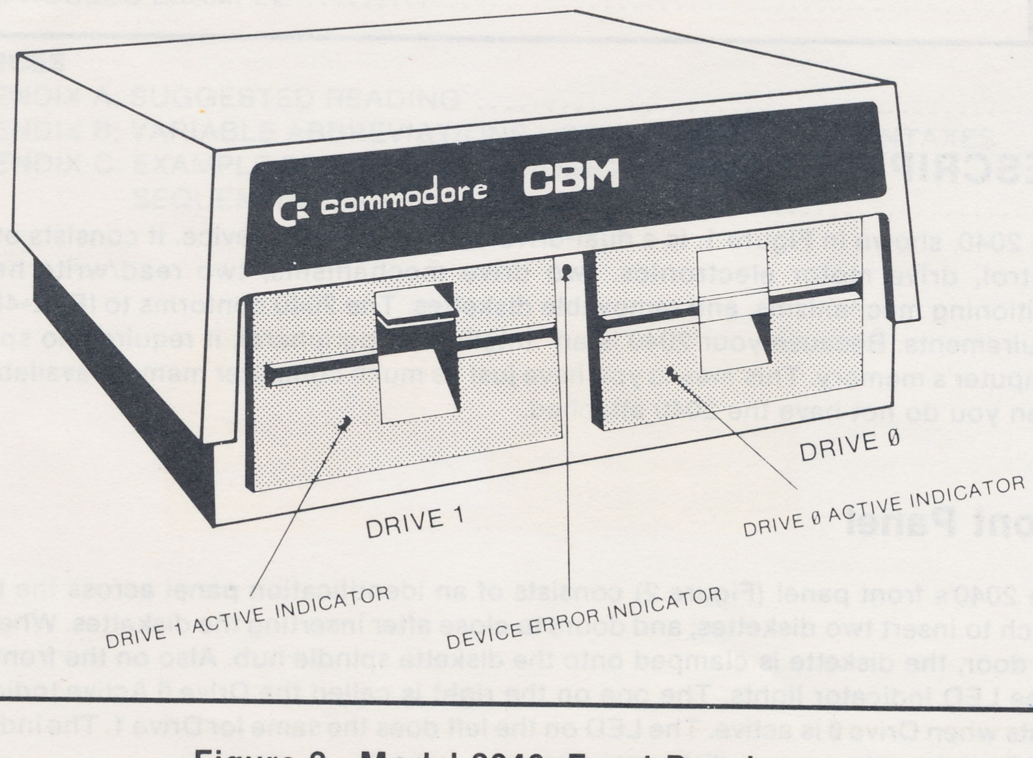

Figure 2. Model 2040: Front Panel

## Back Panel

The back of the 2040 as shown in Figure 3 contains an IEEE-488 interface connector. Near the panel's lower edge is the rocker switch for turning power on and off to the 2040. There is also a "slow blow" fuse, and the cord to the AC power outlet.

## Interior Configuration

The inside of your 2040 contains two Shugart 390 disk drives. All the logic for your 2040 is housed on the underside of the lid. The mechanical devices are, for the most part, located beneath the diskettes. See Figure 4 for a simplified schematic of the interior of your 2040. (Only one disk drive is shown.)

## The Diskette

The diskette (also known as a minifloppy, floppy diskette, minidiskette, etc.) is similar to the standard flexible disk. There are several reputable manufacturers of the 5¼-inch diskettes. You should make sure that you buy diskettes for *soft sectored format*. Your authorized Commodore dealer can supply your needs. Figure 5 shows a diskette in its jacket.

## Specifications

Table 1 contains the specifications for the Commodore 2040 Dual Drive Floppy Disk.

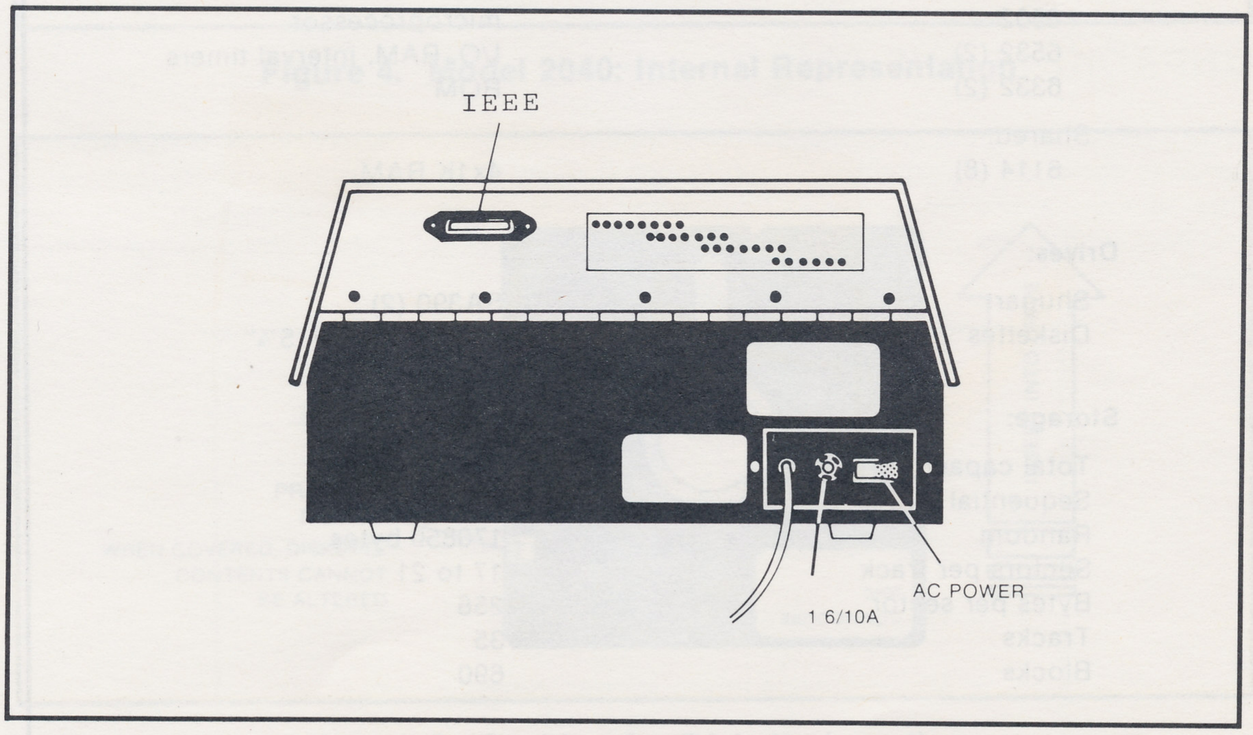

**Figure 3. Model 2040: Rear View**

## Table 1

### Specifications: Model 2040 Dual Drive Floppy Disk

| | |
|---|---|
| **Physical:** | |
| Material: | 18 ga. steel |
| Dimensions: | |
| Height | 6.5" |
| Width | 15" |
| Depth | 14.35" |
| **Electrical:** | |
| Power requirements: | |
| Voltage | 120 VAC |
| Frequency | 60 Hertz |
| Power | 50 watts |
| **IC's:** | |
| Controller: | |
| 6504 | microprocessor |
| 6530 | I/O, RAM, ROM |
| 6522 | I/O, interval timers |
| Interface: | |
| 6502 | microprocessor |
| 6532 (2) | I/O, RAM, interval timers |
| 6332 (2) | ROM |
| Shared: | |
| 6114 (8) | 4x1K RAM |
| **Drives:** | |
| Shugart | SA390 (2) |
| Diskettes | Standard mini, 5¼" |
| **Storage:** | |
| Total capacity | 176640 bytes |
| Sequential | 170180 bytes |
| Random | 170850 bytes |
| Sectors per track | 17 to 21 |
| Bytes per sector | 256 |
| Tracks | 35 |
| Blocks | 690 |

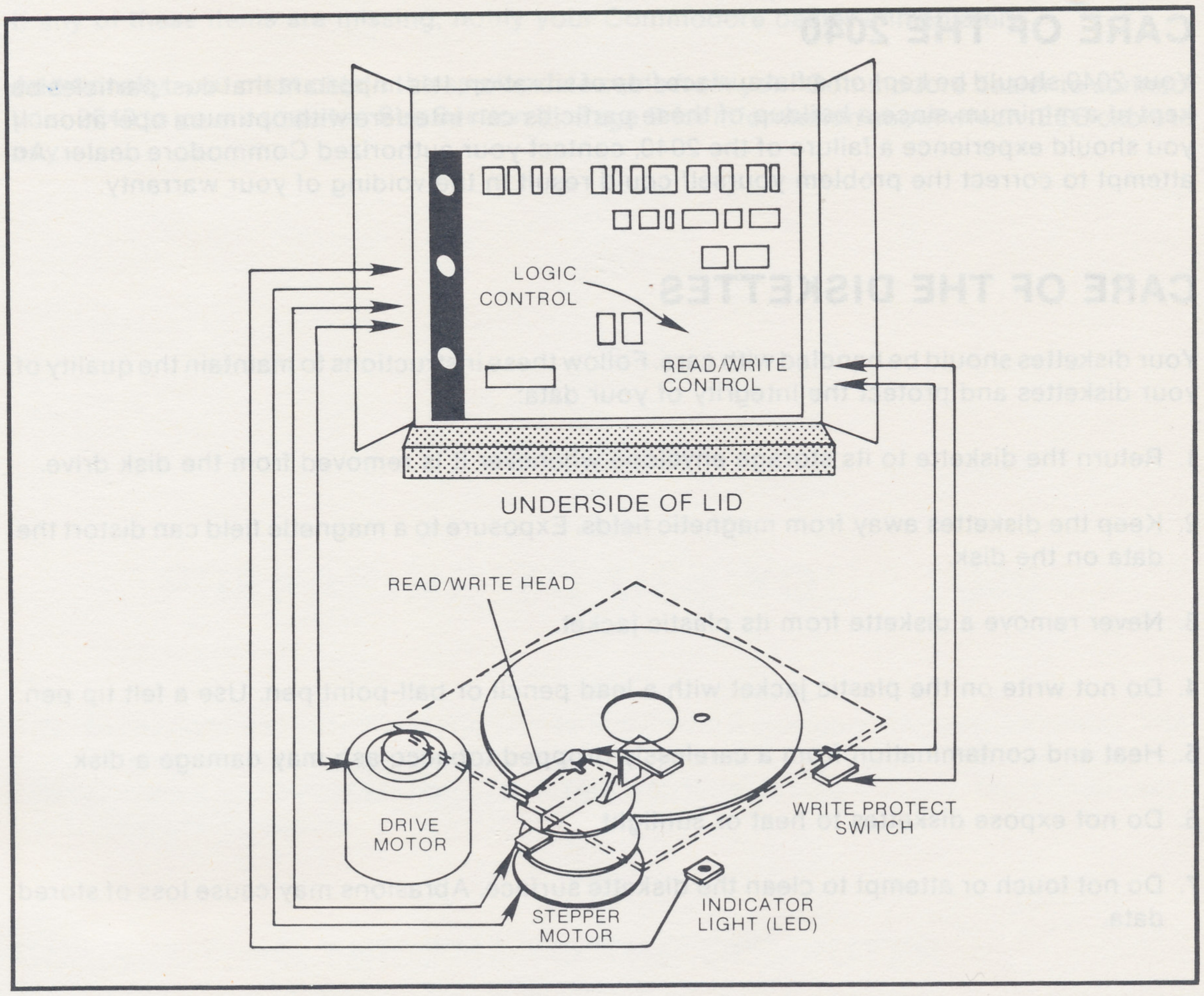

**Figure 4. Model 2040: Internal Representation**

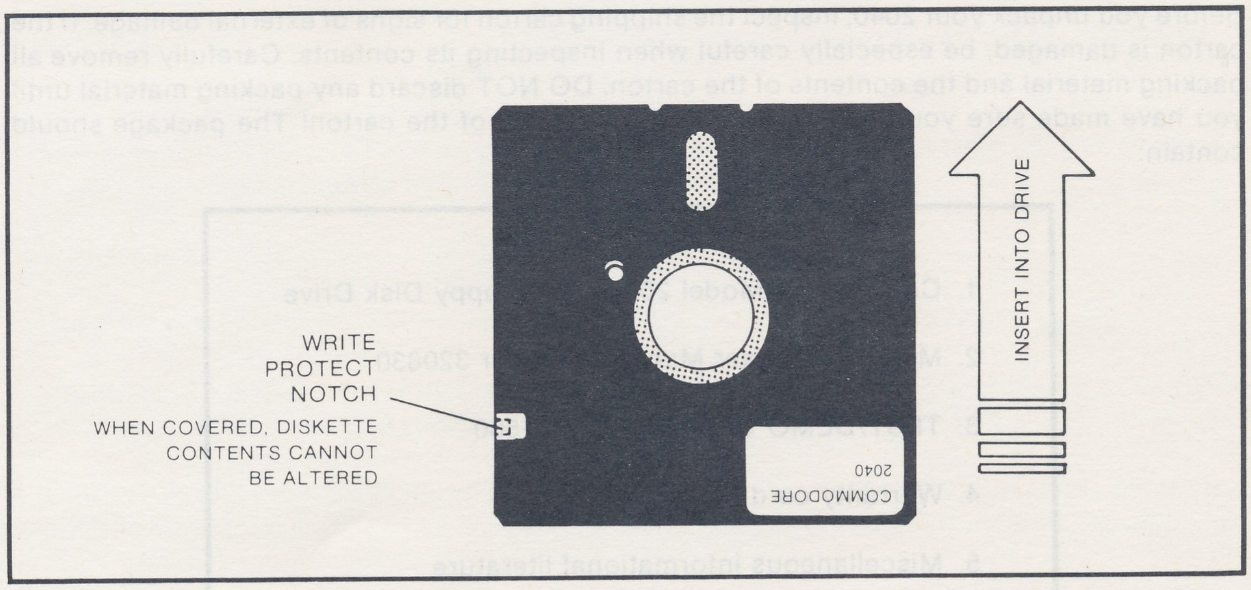

**Figure 5. Position for Diskette Insertion**

## CARE OF THE 2040

Your 2040 should be kept on a flat surface free of vibration. It is important that dust particles be kept at a minimum since a buildup of these particles can interfere with optimum operation. If you should experience a failure of the 2040, contact your authorized Commodore dealer. An attempt to correct the problem yourself could result in the voiding of your warranty.

## CARE OF THE DISKETTES

Your diskettes should be handled with care. Follow these instructions to maintain the quality of your diskettes and protect the integrity of your data:

1. Return the diskette to its storage envelope whenever it is removed from the disk drive.
2. Keep the diskettes away from magnetic fields. Exposure to a magnetic field can distort the data on the disk.
3. Never remove a diskette from its plastic jacket.
4. Do not write on the plastic jacket with a lead pencil or ball-point pen. Use a felt tip pen.
5. Heat and contamination from a carelessly dropped tobacco ash may damage a disk.
6. Do not expose diskettes to heat or sunlight.
7. Do not touch or attempt to clean the diskette surface. Abrasions may cause loss of stored data.

## UNPACKING THE 2040

Before you unpack your 2040, inspect the shipping carton for signs of external damage. If the carton is damaged, be especially careful when inspecting its contents. Carefully remove all packing material and the contents of the carton. DO NOT discard any packing material until you have made sure you have located all the contents of the carton! The package should contain:

1. Commodore Model 2040 Dual Floppy Disk Drive
2. Model 2040 User Manual, Number 320830
3. TEST/DEMO Diskette, P/N 321500
4. Warranty card
5. Miscellaneous informational literature

If any of these items are missing, notify your Commodore dealer immediately.

Additionally, you must obtain the appropriate cable from your Commodore dealer to connect your 2040 to your computer. See Section II, Page 9 for information about which IEEE cable to buy.

## CARE OF THE 2040

Your 2040 [illegible] you should experience a failure of the 2040, contact your authorized Commodore dealer. Any attempt to correct the problem yourself could result in the voiding of your warranty.

## CARE OF THE DISKETTES

Your diskettes should be handled with care. Follow these instructions to maintain the quality of your diskettes and protect the integrity of your data:

1. Return the diskette to its storage envelope whenever it is removed from the disk drive.
2. Keep the diskettes away from magnetic fields. Exposure to a magnetic field can distort the data on the disk.
3. Never remove a diskette from its plastic jacket.
4. Do not write on the plastic jacket with a lead pencil or ball-point pen. Use a felt tip pen.
5. Heat and contamination from a carelessly dropped tobacco ash may damage a disk.
6. Do not expose diskettes to heat or sunlight.
7. Do not touch or attempt to clean the diskette surface. Abrasions may cause loss of stored data.

## UNPACKING THE 2040

Before you unpack your 2040, inspect the shipping carton for signs of external damage. If the carton is damaged, be especially careful when inspecting its contents. Carefully remove all packing material and the contents of the carton. DO NOT discard any packing material until you have made sure you have located all the contents of the carton. The package should contain:

1. Commodore Model 2040 Dual Floppy Disk Drive
2. Model 2040 User Manual, Number 320830
3. TEST/DEMO Diskette, P/N 321500
4. Warranty card
5. Miscellaneous informational literature

## Section 2

# PREPARING TO USE YOUR 2040

Before starting to use your 2040, you should make sure it is in good working condition. This includes properly connecting it directly to your computer or to another IEEE-488 peripheral device, giving it a power-on and initial checkout, and finally giving it a performance test using the TEST/DEMO diskette.

## CONNECTING THE 2040 TO THE COMPUTER

One of two connector cables is required to connect the 2040 to your computer.

1. PET-to-IEEE cable P/N 320202
   Use this cable, if the 2040 is to be the only (or first) IEEE device connected to your computer.
2. IEEE-to-IEEE cable, P/N 905080
   Use this cable if the 2040 is to be connected ("daisy-chained") to another peripheral device such as the Commodore Model 2022 or 2023 Printer.

Follow these steps to connect the 2040 to your computer:

1. Turn off the AC power to your computer.
2. Place the 2040 in a convenient location as close as possible to the computer. DO NOT connect the 2040 to an AC outlet at this time.
3. Connect the PET-to-IEEE cable between the IEEE-488 interface connector on the computer and the connector on the 2040. If additional IEEE devices are to be connected, the IEEE-to-IEEE cable(s) must be used. See Figure 6.
4. Connect the 2040 power cord to an AC outlet. DO NOT turn on power at this time.

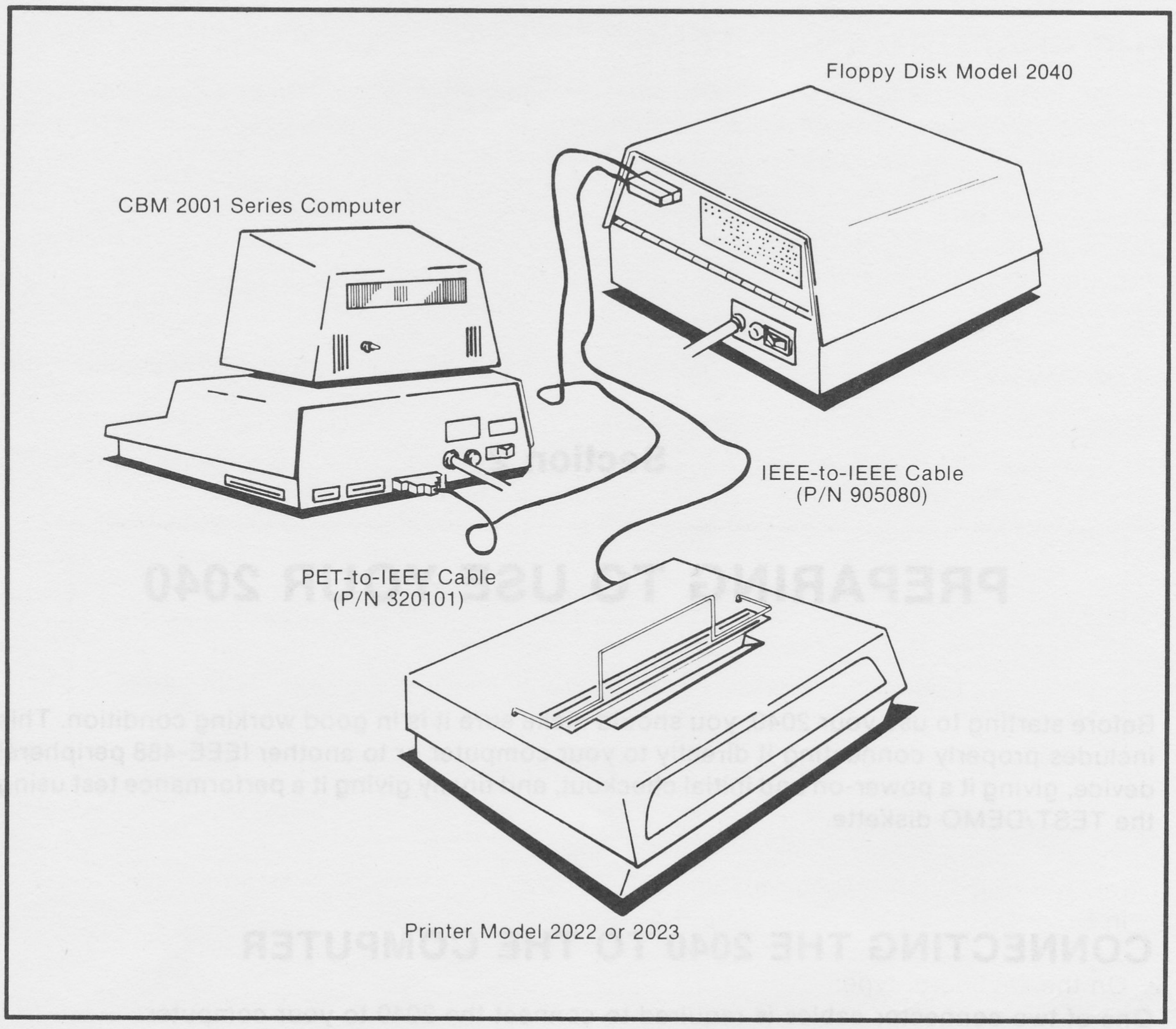

**Figure 6. Floppy Disk Hookup**

## PERFORMING THE POWER-ON TEST

You are now ready to proceed with the power-on part of the checkout:

1. Turn on AC power to the computer and verify that it is working correctly.
2. Open both drive doors on the 2040. Make sure that there are no diskettes in either drive. If you find a diskette, take it out.
3. Turn on AC power to the 2040. All three indicator lights (LEDs) on the front panel light briefly then go out. If any indicator light remains lit for more than about three seconds, turn off the power to the 2040. Wait five minutes and try again. If any light still remains lit, contact your Commodore dealer immediately.

## THE PERFORMANCE TEST

When the previous steps have been completed successfully, you may proceed with the performance test. Don't worry if you don't fully understand exactly what is happening in this test. All commands and results are explained in Section 3. At this point, just enter the commands by rote to get a feel for what you can do with your 2040. If unexpected results are obtained during any step of the test, stop and start over again. The most likely cause of a problem is an improperly entered command. This is to be expected until you become familiar with your 2040.

All commands that you type at the keyboard should be followed by a carriage return: the RETURN key on your keyboard.

NOTE: You must enter the commands exactly as shown on the instructions below. Do not insert any spaces in the commands unless shown in the instructions. If the error indicator lights up, you may be able to continue the example anyway. Re-enter your last command. If the light goes out, your correction was successful and you may continue.

1. Insert the TEST/DEMO diskette into Drive Ø as shown in Figure 7. Insert a blank diskette in Drive 1. Remember to close both drive doors.

2. On the keyboard, type:

```
OPEN 1,8,15
```

This command opens logical file 1 on device 8 (your 2040). The secondary address of 15 opens the command channel to the 2040. The screen displays your entry followed by READY.

NOTE FOR BUSINESS KEYBOARD USERS ONLY: If you have a business keyboard, you may wish to set your computer for upper case character entry. You do this by typing:

```
POKE 59468,12
```

and pressing the RETURN key. Although it is not absolutely necessary to give this command prior to communicating with your 2040, it does permit easy entry. In addition, the examples in this manual can be duplicated exactly when you use only upper case.

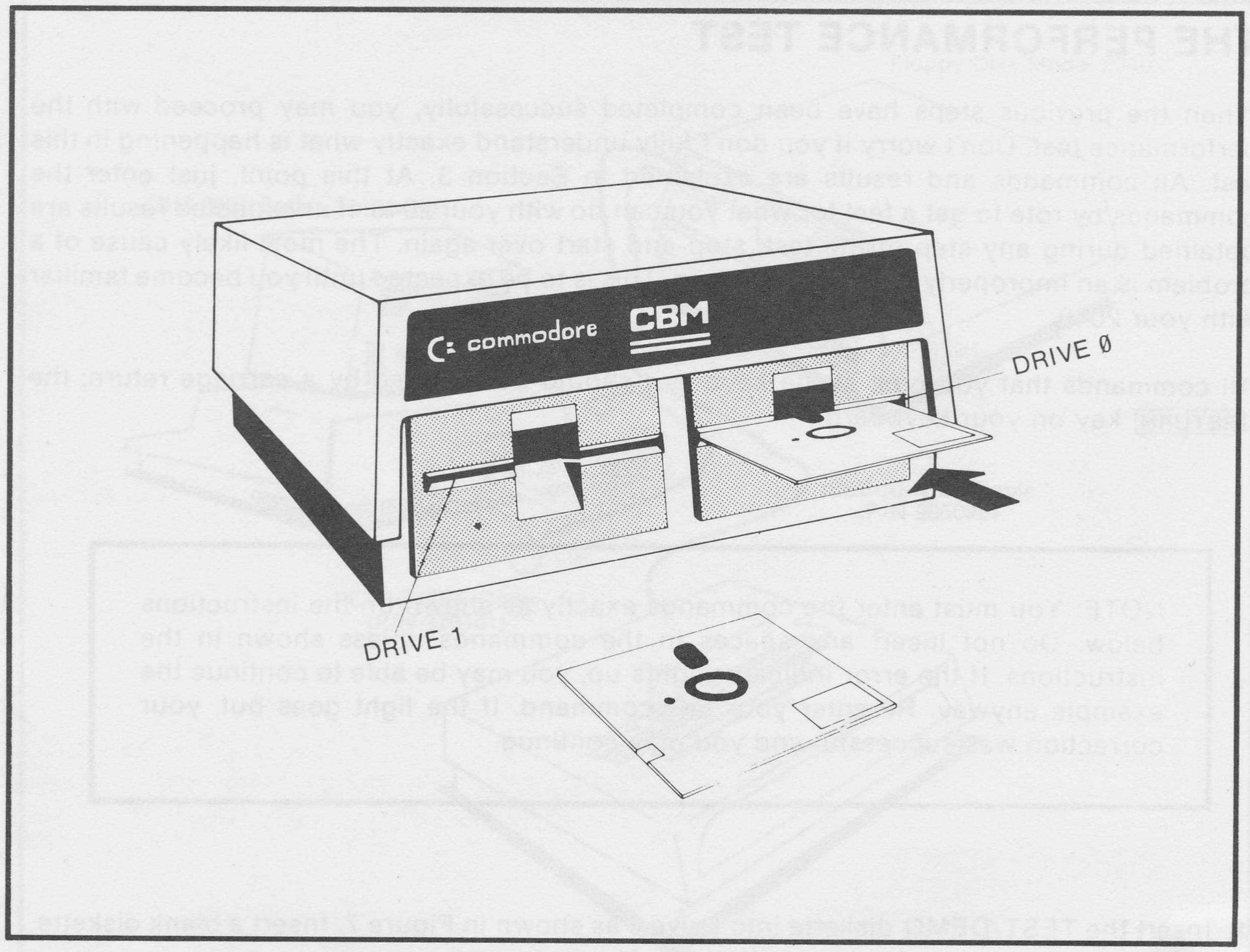

**Figure 7. Inserting the Diskettes**

3. On the keyboard, type:

```
PRINT #1,"NEW1:TEST,99"
```

PRINT#1 sends the command string to the 2040. It formats (soft-sectors) the diskette in Drive 1, gives the diskette the name TEST, and an ID of 99. The computer displays your entry and then READY. The Drive 1 Active indicator LED lights and the drive motor runs for about one minute. Then the motor stops and the light goes out.

4. On the keyboard, type:

```
PRINT#1,"IØ"
```

This command initializes Drive Ø. (Drive 1 was initialized as part of the formatting procedure in number 3). The initialization procedure places the magnetic head of the drive in the proper position above the diskette. The computer displays your entry, then the word READY. The Drive Ø Active indicator lights up and the Drive Ø motor runs briefly. Then the motor stops and the light goes out.

5. On the keyboard, type:

```
LOAD "$Ø",8
```

This command loads the file directory from the TEST/DEMO diskette into the computer's internal memory.

The computer display now shows:

```
LOAD "$Ø",8
SEARCHING FOR $Ø
LOADING
READY.
```

At the same time, the Drive Ø indicator lights and the Drive motor runs briefly.

6. On the keyboard, type:

```
LIST
```

This command requests the display of the TEST/DEMO diskette file directory which was placed in the computer's memory during step 5. The screen now displays your entry and the entire file directory for TEST/DEMO.

Notice that the BASIC command LIST involves no interaction with the 2040 since the directory is retrieved from your computer's memory.

7. On the keyboard, type:

```
LOAD"Ø:DIAGNOSTIC BOOT",8
```

This command loads the Diagnostic Boot program from the diskette in Drive Ø with the computer's memory.

The computer displays your entry and:

```
SEARCHING FOR Ø:DIAGNOSTIC BOOT
LOADING
READY.
```

Simultaneous with the screen display, the Drive Ø indicator lights and the Drive Ø motor runs. Before going on to step 8, make sure you remove the diskettes when directed to by the screen display.

8. On the keyboard, type:

```
RUN
```

The program is executed. It checks for malfunctions in I/O ICs, RAMs, and ROMs. Follow the directions on the screen. If all three indicators flicker continuously, the test is passed. If all indicators remain lit in a steady pattern, the directions on the screen can be used to find the problem area. Contact your Commodore dealer for help. After 30 seconds of operation, reset the 2040 by turning the rocker switch on the 2040 off, then back on.

9. On your keyboard, press:

   **RUN STOP**

   Then press the **SHIFT** key simultaneously with the **CLR HOME** key. Now return the diskettes to their respective drives.

10. On the keyboard, type:

```
PRINT#15,"IØ"
LOAD "Ø:PET DISK",8
```

   This program loads the file containing the PET DISK program from the diskette in Drive Ø to the computer's memory.

   The computer displays your entry and:

```
SEARCHING FOR Ø:PET DISK
LOADING
READY.
```

   The Drive Ø motor runs simultaneously with the display shown above.

11. On the keyboard, type:

```
RUN
```

   The PET DISK demonstration program is displayed continuously until you terminate it. Notice that you can hear the Drive Ø motor run intermittantly as one program calls the next.

12. On the keyboard, press:

   **RUN STOP**

   Then press the **SHIFT** key simultaneously with the **CLR HOME** key. This completes the Model 2040 Dual Drive Disk performance test.

   You now know that your 2040 works. You have had a taste of operating it. Now you are ready to really make it perform.

## Section 3

# USING YOUR 2040

Your 2040 adds to your computing power with added storage and file handling capability. You control it directly with BASIC commands given from the keyboard, from BASIC statements within programs, and with special disk commands. In this section you will learn how to apply these commands and statements to the 2040. The section is organized in such a way that the functions and syntaxes of the BASIC commands and the special disk commands are described first. Then instructions on how to use these commands to perform 2040-related tasks are given. For the most part, the instructions in this section apply to program (PRG) files that contain BASIC programs.

Before you attempt to use your 2040, make sure you know how to do the following:

1. Operate your Commodore computer
2. Do elementary programming in BASIC
3. Open and close files

You should refer to your computer's User Manual for this information.

This section uses certain conventions to indicate specific actions or requirements:

| Example | Description |
|---|---|
| *dn* | Italicized lowercase letters in a syntax indicate that you should enter something (a variable name) in place of the letters. |
| [ ] | Brackets indicate optional usage. |

NOTE: The BASIC statements described in this manual apply specifically to use with the 2040. Certain of the commands and statements may follow a slightly different syntax or produce different results from those described here when they are used for the computer or for other peripheral devices. Consult the appropriate manual for the exact usage of these commands and statements in other contexts.

## 2040-ASSOCIATED BASIC COMMANDS

Several BASIC commands that allow you to communicate with and transfer data to and from your 2040 are described in this section. These commands are:

| | |
|---|---|
| **OPEN** | **VERIFY** |
| **CLOSE** | **LOAD** |
| **SAVE** | **PRINT#** |

Let's take a quick look at the syntax of each of these commands and find out what each one does. Then, in subsequent sections, you will see how to use these commands to perform certain 2040-related tasks. For more about these commands, and peripheral devices in general, read Chapter 7 of your computer's User Manual.

### The OPEN Command

This command sets up a correspondence between a file number and the 2040. It also opens a specified channel between the computer and the 2040. The syntax of OPEN is:

**OPEN *lfn,dn,sa***

| | | |
|---|---|---|
| Where: | ***lfn*** | is a logical file number. You assign this number arbitrarily and it may be any number between 1 and 255. |
| | ***dn*** | is the device number. This number must be 8 since that is the number assigned to it at the factory. |
| | ***sa*** | is the secondary address. The secondary address is used to open channels between the 2040 and the computer. At this point, we will deal with Channel 15 exclusively. Channel 15 is the command and error message channel. Later, we will explore the use of other secondary addresses. |

## The CLOSE Command

The CLOSE command closes a file opened by the OPEN command. Its syntax is:

**CLOSE *lfn***

| | | |
|---|---|---|
| Where: | ***lfn*** | is the logical file number of a file opened by the OPEN command. |

You should always close a file after working with it. You are not allowed to have more than ten open files in the computer and five in the 2040, so it is well to make a habit of closing files as soon as possible. This way you will always have the maximum number of files available for use.

## The SAVE Command

The SAVE command transfers programs from the computer's memory to the specified diskette. SAVE always assigns a file type of PRG (for program) to files that are written to the diskette with it. The syntax of SAVE is:

**SAVE *"dr:fn",dn***

| | | |
|---|---|---|
| Where: | ***dr*** | is the disk drive number. It must be Ø or 1. |
| | ***fn*** | is any file name of 16 characters or less you wish to assign to the file to be transferred to the diskette. Blanks are counted as characters. |
| | ***dn*** | is the device number and it must be 8. |

Another syntax of SAVE is:

**SAVE *"@dr:fn",dn***

| | | |
|---|---|---|
| Where: | **@** | means to replace the contents of an existing diskette file. |
| | ***fn*** | is the name of the diskette file whose contents is to be replaced. |

## The VERIFY Command

The VERIFY command performs a byte-by-byte comparison of the file just written to the diskette against the file in the computer's memory. The syntax of the VERIFY command is:

**VERIFY *"dr:fn",dn***

| | | |
|---|---|---|
| Where: | ***dr*** | is the same drive number as used in the immediately preceding SAVE command. |
| | ***fn*** | is the same file name as used in the immediately preceding SAVE command. |
| | ***dn*** | is the device number, in this case 8. |

Another syntax of VERIFY is:

**VERIFY "*",*dn***

| | | |
|---|---|---|
| Where: | * | causes the program just saved to be verified. |

You should give the VERIFY command every time you use the SAVE command to place a file on a diskette. If you do this, you will always know whether or not the file was copied correctly.

## The LOAD Command

The LOAD command transfers program (PRG) files from the specified diskette to the computer's memory. The syntax of LOAD is:

**LOAD *"dr: fn",dn***

| | | |
|---|---|---|
| Where: | ***dr*** | is the drive number from which you are loading data. It must be 0 or 1. |
| | ***fn*** | is the file name previously specified in the SAVE command and/or stored in the disk directory. |
| | ***dn*** | is the device number and it must be 8. |

After a successful LOAD, you must give a new OPEN command in order to continue communicating with the 2040.

## The PRINT# Command

The PRINT# command transmits a diskette command string to the 2040. The syntax of PRINT# is:

**PRINT# *fn,"commandstring"***

Where: commandstring contains disk handling or disk file handling commands. These disk commands are:

| | |
|---|---|
| **N -** | (NEW). Formats (soft-sectors) the diskette and initializes it. |
| **I -** | (INITIALIZE). Aligns the drive head, reads the directory track, loads disk data into the Disk Operating System (DOS). |
| **V -** | (VALIDATE). Creates the Block Availability Map (BAM) according to valid disk data and initializes the diskette. |
| **D -** | (DUPLICATE). Duplicates an existing disk. |
| **C -** | (COPY). Copies and/or merges files from one disk to another or on the same disk. |
| **R -** | (RENAME). Changes the name of an existing valid file. |

**S -** (SCRATCH). Deletes a file.

You have a general idea of what each of six BASIC commands and seven diskette handling commands do. Now you will see how to use these commands to perform various tasks.

## PREPARING TO USE A DISKETTE

Every time you place a diskette in one of your disk drives you must follow a certain procedure to prepare it and the disk drive for use. Some form of this initialization must occur every time you insert a diskette. Initialization or reinitialization occurs when you give the INITIALIZE, VALIDATE, DUPLICATE, or NEW diskette commands.

> REMEMBER: Diskette commands must be transmitted in PRINT# commands. All PRINT# commands require that an OPEN command be in effect.

### Preparing a Blank Diskette

A blank diskette has no sector formatting on it. Before it can be used it must be formatted. Follow this procedure:

1. Turn on the computer and the 2040. Insert the diskette in either disk drive *after* the power is turned on. Never turn the power on to the 2040 with a diskette in place.
2. Type:

   ```
   OPEN 1,8,15
   ```

   The 1 is a logical file number; the 8 is the 2040's device number; the 15 is the secondary address that opens the command and error channel to the 2040.
3. Next, use the disk command NEW to format (soft-sector) the diskette and to initialize the disk drive. The syntax of the NEW command string is:

   **"N[EW]*dr:dskname,id*"**

| | | |
|---|---|---|
| Where: | ***dr*** | is the drive number, Ø or 1 |
| | ***dskname*** | is the name you wish to assign to the disk. It may be up to 16 characters long. |
| | ***id*** | is a unique two-character identifier. |

The NEW disk command writes headers for each data block on all 35 tracks. It clears the disk directory and reinitializes the Block Availability Map (BAM).

**Example:**

```
OPEN 1,8,15
PRINT#1,"NØ:TESTDISK, XX"
```

## Preparing To Reuse An Old Diskette

You can reuse an old diskette just as if it were a blank diskette. Simply follow the same procedure as you would if it were a blank diskette.

**Example:**

```
OPEN 1,8,15
PRINT#1,"N1:JONES,99"
```

All of the old data, the disk name, and ID are removed. The new disk name and ID replace the old ones.

> WARNING: There is no way to recover directly old data from a diskette once the NEW disk command has been executed.

If you wish, you can remove all of the old data from an old diskette, but retain or change the diskette name as desired. The two-character ID remains the same. There are two reasons for "reNEWing" a diskette this way: it takes less time—20 seconds against 80 seconds; you can keep track of individual diskettes by assigning unique IDs and then never changing those IDs. If you prefer to do this rather than treating the diskette as a blank diskette, use the following non-formatting NEW command:

**"N[EW]*dr:dskname*"**

| | | |
|---|---|---|
| Where: | ***dr*** | is the drive number, Ø or 1. |
| | ***dskname*** | is either the existing diskette name or a new one. |

Notice that ID is not included in this disk command.

**Example:**

```
OPEN 1,8,15
PRINT#1,"NEW 1:NEW DATA"
```

However you choose to reformat an old diskette, the disk command clears the disk directory and all disk files.

## Preparing a "Currently-in-Use" Diskette

A "currently-in-use" diskette may be pre-formatted but contain no files yet, or it may already contain files. It is not necessary to go through the formatting procedure for this kind of diskette. In the previous sub-sections, the initialization procedure was included in the NEW command. In this case, however, you have to use the INITIALIZE command.

The INITIALIZE command aligns the read/write head with track 1 on the specified diskette. It then moves to track 18, reads the disk name and ID, and loads the information into the Disk Operating System (DOS) memory. The syntax of the INITIALIZE disk command string is:

**"I[NITIALIZE][dr]"**

| | | |
|---|---|---|
| Where: | *dr* | is the drive number, Ø or 1. If no drive number is specified, both drives are initialized. |

**Example 1:**

```
OPEN 1,8,15
PRINT#1,"IØ"
```

This command initialized drive Ø.

**Example 2:**

```
OPEN 1,8,15
PRINT#1,"INITIALIZE"
```

This command initializes both drives.

**Example 3:**

```
OPEN 1,8,15
PRINT#1,"NEW 1:ACCTS,CC"
PRINT#1,"IØ"
```

In this example, both drives are initialized. Notice that there is no need to give the INITIALIZE command for Drive 1, since initialization is included in the NEW command.

## WRITING A PROGRAM TO A DISKETTE

Once you have placed a program in the computer's memory, you can move it onto a diskette for storage. This is accomplished with the SAVE command as described on page 17. Any data transferred with a SAVE command is automatically designated by the system as a program (PRG) file.

The following example shows how a program is typed into the computer's memory, then moved from the computer's memory to the specified diskette.

**Example:**

```
10? "THIS IS A TEST"
SAVE "0:TESTPROG", 8
VERIFY "0:TESTPROG", 8
```

## READING A PROGRAM FROM A DISKETTE

Reading a program from a diskette into your computer's memory is as easy as putting it on the diskette in the first place. This task is accomplished with the LOAD command. You must specify the drive number, the program name, and the device number.

This example shows how the program in the previous example is loaded from the diskette to the computer's memory, then executed. (If you wish to do this example, first type NEW and depress RETURN to clear your computer's memory so that you can see that it really works.)

**Example I:**

```
LOAD "0:TESTPROG", 8
READY.
RUN
THIS IS A TEST
```

Example 2 shows a session with the computer, a tape cassette, the 2040, and a printer. The purpose of the session is to copy a program on the cassette tape to a diskette. The program is then read from the diskette to the computer's memory and finally printed on the printer, It is assumed that the BASIC program was previously stored on the cassette.

**Example 2:**

| | |
|---|---|
| **LOAD"DEMO"**<br><br>PRESS PLAY ON TAPE # 1<br>OK<br><br>SEARCHING FOR DEMO<br>FOUND DEMO<br>LOADING<br>READY. | You load the file from the cassette tape to the computer's memory. |
| **SAVE"1:DEMO",8**<br>**VERIFY"1:DEMO",8**<br>READY. | You move a copy of the file from the computer's memory to a diskette. |
| **NEW** | You erase everything from the memory. |
| **LOAD "1:DEMO",8**<br>SEARCHING FOR 1:DEMO<br>LOADING<br>READY. | You load the program back into the computer's memory. |
| **RUN**<br>THIS IS A DEMONSTRATION TO<br>SHOW HOW WE MOVE A PROGRAM<br>FROM ONE MEDIUM TO ANOTHER.<br><br>READY. | You run the program just to prove to yourself that it's really there! |
| **OPEN1,4**<br><br>READY.<br><br>**CMD1**<br>**LIST**<br>**PRINT#1**<br>**CLOSE 1** | You open the file to the printer and list the program on the printer.<br><br>The printer prints:<br><br>10 PRINT"THIS IS A DEMONSTRATION TO"<br>20 PRINT"SHOW HOW WE MOVE A PROGRAM"<br>30 PRINT"FROM ONE MEDIUM TO ANOTHER"<br>READY. |

## DISPLAYING THE DISKETTE DIRECTORY

You can display the directory for the diskette in the specified diskette drive with:

**LOAD *"$dr",dn***

| | | |
|---|---|---|
| Where: | ***dr*** | is the drive number |
| | ***dn*** | is the device number (in this case, 8, for the 2040). |

The disk directory includes a header that tells the name of the disk and its ID. The data shows the name, the number of blocks used, and the file type of each file. The numbers of unused blocks is also displayed. (A block contains 256 bytes.)

The following example shows a request to load the TEST/DEMO diskette's directory. You have already inserted the diskette and initialized the drive.

**Example:**

```
LOAD "$Ø", 8
SEARCHING FOR $Ø
LOADING
READY.
```

The TEST/DEMO's directory is moved into your computer's memory. Display it by typing:

```
LIST
```

## WORKING WITH YOUR DISKETTE FILES

You need to do more than just move your files from your computer to a diskette and back, and list them. This subsection describes the 2040 file handling disk commands. These are:

| | **Command** | **Function** |
|---|---|---|
| Diskette Level | **VALIDATE** | Creates a Block Availability Map (BAM) |
| | **DUPLICATE** | Duplicates a diskette |
| File Level | **COPY** | Copies files |
| | **RENAME** | Renames a file |
| | **SCRATCH** | Erases a file |

## The VALIDATE Disk Command

The VALIDATE disk command creates a Block Availability Map (BAM) according to the valid data on the diskette. The syntax of VALIDATE is:

**PRINT#*lfn*, *"V[ALIDATE] dr"***

You may abbreviate VALIDATE to V if you prefer. If you do not specify a drive number, the diskette in the last drive used during the current session is verified.

In addition to creating the BAM, VALIDATE deletes from the directory files that were never properly closed. If a READ error is encountered during a VALIDATE, the operation aborts and leaves the diskette in its previous state. If a VALIDATE error does occur, *you must re-initialize* before proceeding.

**Example:**

```
OPEN 1, 8, 15
PRINT#1,"VØ"
```

or

```
PRINT#1,"VALIDATE Ø"
```

## The DUPLICATE Disk Command

The DUPLICATE disk command creates an identical diskette that includes the diskette name, ID, and all files. It is important that both disks be in good condition since any read or write errors will cause the operation to abort.

The syntax of DUPLICATE is:

**PRINT#*lfn*, "D[UPLICATE]*ddr*=*sdr*"**

| | | |
|---|---|---|
| Where: | ***ddr*** | is the destination diskette |
| | ***sdr*** | is the source diskette |

WARNING: Make sure that you do not reverse the order of the drive numbers. If you do you will lose all your data and there is no way to recover it.

**Example:**

```
OPEN 1,8,15
PRINT#1,"DUPLICATE Ø=1"
```

or

```
PRINT#1,"DØ=1"
```

## The COPY Disk Command

The COPY disk command allows you to copy files from one diskette to another, or to duplicate files on the same diskette. You can also use this command to concatenate data files.

The syntax of COPY is:

**PRINT#*lfn*, "C[OPY] *ddr:dfn=sdr:sfn,sdr:sfn . . .***

| | | |
|---|---|---|
| Where: | ***ddr*** | is the destination drive. The file is to be copied onto the diskette in this drive. |
| | ***dfn*** | is the destination file name. This name may be either a new name or the same as the old file name. |
| | ***sdr*** | is the source drive the original file is on the diskette in this drive. |
| | ***sfn*** | is the source file name. |

You may include up to four source files to be concatenated into your destination file.

In Example I below, a file is copied from the diskette in Drive 1 to the diskette in Drive Ø. In Example 2, files from both drives are concatenated into a file on Drive 1.

**Example I:**

```
PRINT#1,"C1:ACCT1=Ø:ACCT"
```

**Example 2:**

```
PRINT#1,"C1:JDATA=1:ACCT1,Ø:ADATA,Ø:BDATA"
```

## The RENAME Disk Command

The RENAME disk command renames an existing diskette file. No file must already exist with the file name specified in the command or an error will occur.

The syntax of RENAME is:

**PRINT#*lfn*, "R[ENAME]*dr:nfn=ofn*"**

| | | |
|---|---|---|
| Where: | ***dr*** | is the disk drive on which the diskette is located. |
| | ***nfn*** | is the new name of the file. |
| | ***ofn*** | is the old name of the file. |

## The SCRATCH Disk Command

The SCRATCH disk command erases unwanted files from the specified diskette and its directory. Files may be erased individually or several may be erased at one time.

The syntax of SCRATCH is:

**PRINT#*lfn*, *"S[dr][:]fn, [[dr][:]fn. . .[dr][:]fn]"***

| | | |
|---|---|---|
| Where: | ***dr*** | is the disk drive to be searched. |
| | **:** | alone means "last drive accessed," with ***dr*** refers to the specified drive, where not used means "both drives." |
| | ***fn*** | is the name of the file to be erased. |

You can erase one file, several files, or all the files on a diskette. To erase one file, enter the entire name of the file:

```
PRINT#1,"SØ:ACCT"
```

To erase several files with unrelated names, enter the entire name of each file to be deleted:

```
PRINT#1, "SØ:ACCT, Ø:CUSTOMER,Ø:INV"
```

To erase several files at one time where names begin the same way, use an asterisk (*) after the last common character. If you have files named BASICS, BASE DATA, and BASKET, place the asterisk after the S to delete all files begining with BAS:

```
PRINT#1,"S1:BAS*"
```

Or, if the files named above have totally unique names, you can use just the first letters of the files as shown below:

```
PRINT#1,"SØ:A*,:C*,:I*"
```

To erase several files with certain common characters or strings of characters, replace the unique characters with question marks (?). To erase the files MONITR.SRC and EDITOR.SCR on Drive Ø, replace the first through sixth characters with questions marks as shown below:

```
PRINT#1,"SØ:??????.SRC"
```

The examples below show how you can use the asterisk and question marks together:

**Examples:**

| **Designation** | **May refer to** |
|---|---|
| FILE?* | FILE 1 FILE2 FILE %¢ |
| P???FIL* | PET FILE PRG FILE-32 POKEFILES$$ %@ |
| ???* | TEXTILE OBJ A B |

You can erase all the files on a diskette. Only the diskette name and ID remain. Simply specify the disk drive and include an asterisk in the SCRATCH disk command.

**Example:**

```
PRINT#1,"SØ:*"
```

## DETERMINING THE FILE TYPE

If you want to store BASIC programs on diskette, the best way to do it is in PRG files. These are the files that we have dealt with up to this point in this manual. PRG (or program) files can be loaded with the LOAD command, and saved and assigned a file type of PRG with the SAVE command.

The rest of this section contains reference to two more kinds of files: Sequential (SEQ) and User (USR). LOAD and SAVE are not available for these file types.

## USING THE DATA CHANNELS

So far you have learned how to use the SAVE and LOAD commands to write and read program (PRG) files that contain BASIC programs. You used a secondary address of 15 to open the command channel and receive error messages. You also learned that you can have several logical files open at one time. But your CBM 2001 series computer and 2040 team has a lot more capability than just described. Data of many types can be stored and accessed. Data as well as command strings can be transmitted to the 2040 via the PRINT# command; and data from the diskette file can be transmitted to the computer with the INPUT# or GET# command. Read Chapter 7 of your User Manual for a complete discussion of these commands.

## The OPEN Command (Complete Version)

As before, you need to open the file and channel(s) to the file.

The syntax of the complete OPEN command is:

**OPEN *lfn, dn, sa, "dr:fn,ft,mode"***

| | | |
|---|---|---|
| Where: | ***lfn*** | is the logical file number |
| | ***dn*** | is the device number; in this case 8 |
| | ***sa*** | is the secondary address. It may be any number from 2 to 14 and may be used either for input or output as specified in mode. |
| | ***dr*** | is the driver number: Ø or 1 |

| | |
|---|---|
| ***fn*** | is the name of the file. |
| ***ft*** | is the file type. It may be SEQ (for sequential), USR (for user), or PRG (for program). |
| ***mode*** | describes how the channel is to be used. It may be either READ or WRITE. |

**Examples:**

```
OPEN 2, 8, 2, "Ø: FILE 1,SEQ,WRITE"

OPEN 3,8,9, "1:TEST DATA,PRG,WRITE"

OPEN 8, 8, 8, "Ø:NUM,USR,READ"
```

You can replace the contents of an existing file by preceding the drive number with an *at* sign (@) as shown here:

```
OPEN 3,8,5,"@Ø:JDATA,USR,WRITE"
```

If the specified file does not exist, then normal opening procedures are executed.

You can also assign some of the information that goes in the OPEN command to a variable name as shown below:

```
FL$="Ø:FILE A,SEQ,READ"
OPEN 1,8,14, FL$
```

or:

```
FL$="Ø:FILE A"
OPEN 1,8,14,FL$+",SEQ,WRITE"
```

The methods above are convenient when you need to open several channels to the same file name.

## Closing the Data Channel

The CLOSE command closes a file and the data or command channel associated with it. Whenever you close a file opened with a write channel, the closing of that file writes the final block of data to the disk and updates the disk directory. When you close a file opened with a read channel, that channel is simply closed down.

NOTE: When a drive is initialized with INITIALIZE, NEW, DUPLICATE, or VALIDATE, all channels associated with that drive are deleted in the 2040. These commands should not be executed when there are any files open since the files will be disrupted.

## Closing the Command Channel

The close of the command channel causes the close of all channels associated with the 2040. No other part of the logical file environment is affected.

The following example illustrates a situation in which several channels are closed down by a single CLOSE command.

**Example:**

```
OPEN 1,8,15                      → The command channel is opened.
OPEN 3,8,2,Ø:FILE1,SEQ,WRITE"    → Data channels are opened for writing.
OPEN 4,8,5,"Ø:FILE2,SEQ,WRITE"
PRINT#3,"IMPORTANT DATA"
PRINT#4,"MORE DATA"              → A channel is opened to the printer by
OPEN 3,4                           mistake.
?FILE OPEN ERROR?                → An error message is displayed on the
                                   screen.
```

Since there was an error, all logical files in the computer are closed, but the channels in the 2040 are still open. To close the 2040 channels, the operator types:

```
OPEN 1,8,15
CLOSE 1
```

Now all data channels in the 2040 are closed properly.

## Data Transfer to the Diskette

You can send data to a previously-opened diskette file with the PRINT# command. Unless you use a semicolon as a terminator for each PRINT# statement, the BASIC interpreter sends a carriage return and line feed to the diskette. These characters are written to the diskette as part of the data. It is important to be aware of this fact because when data is read from the file, these characters are also transferred.

**Example:**

```
PRINT#2,"JONESABC";CHR$(13);
```

The CHR$(13) is the carriage return necessary for the proper termination of the INPUT# statement.

## Data Transfer from Diskette

After opening one or more read channels, you can transfer data from the diskette to the computer. Use the commands INPUT# or GET#. Both commands follow the rules of the computer's BASIC interpreter.

The syntax of INPUT# is:

**INPUT# *lfn, vname[,vname, . . .vname]***

Where: ***vname*** is the numeric or string variable to be read from the specified file on the diskette.

***vname*** may be either a simple character to specify a numeric input, or a single character followed by a dollar sign ($) to specify alphabetic input. See your User Manual for variable names rules.

The following example shows how INPUT# is used in a BASIC program to read in data:

```
INPUT#1,EN,EM$,ET,ES
```

The syntax of GET# is:

**GET# *lfn, vname[,vname, . . .vname]***

GET# transfers data, one character at a time, to the computer. This command is useful when the data is stored in a form that is not directly acceptable to BASIC. It is also useful when a programmer wishes to have immediate access to a character as it is transferred to the computer's memory.

*See Appendix C for an example of Sequential Reading & Writing under Program Control.*

## REQUESTING ERROR MESSAGES

Whenever the LED device error indicator in the middle of the 2040's front panel is lit, an error has occurred. To find out what the error is, you must execute a short program. The execution of the program displays the error on the computer's screen and resets the device error indicator.

The program you execute should open the error message channel (***sa*** = 15), request one to four variables transferred to the computer and then display those variables. The program below shows one way to request error messages.

**Example:**

```
10 OPEN 1,8,15
20 INPUT#1,A$,B$,C$,D$
30 PRINT A$,B$,C$,D$
```

You may request only field A$, fields A$ and B$, fields A$, B$, and C$, or all fields as shown above.

The error message format is:

**A, B, C, D**

| | | |
|---|---|---|
| Where: | **A** | is the error message number |
| | **B** | is the error message |
| | **C** | is ØØ or the track number |
| | **D** | is ØØ or the sector number |

Table 2 shows the 2040 error messages.

## Recovering DOS Errors

Often the errors described here represent hard errors. This type of error cannot be easily recovered. Other errors may simply require the retyping of a command.

Read errors: 2Ø, 21, 22, 23, 24, 27

2Ø: READ ERROR (block header not found)
The disk controller is unable to locate the header of the requested data block. Caused by an illegal sector number, or the header has been destroyed.

21. READ ERROR (no synch character)
The disk controller is unable to detect a synch mark on the desired track. Caused by misalignment of the read/write head or no diskette is present. Can also indicate a hardware failure.

22. READ ERROR (data block not present)
The disk controller has been requested to read or verify a data block that was not properly written. This error message occurs in conjunction with the BLOCK commands and indicates an illegal track and/or sector request.

23. READ ERROR (checksum error in data block)
This error message indicates that there is an error in one or more of the data bytes. The data has been read into the DOS memory, but the checksum over the data is in error. This message may also indicate grounding problems.

24. READ ERROR (byte decoding error)
The data or header has been read into the DOS memory, but a hardware error has been created due to an invalid bit pattern in the data byte. This message may also indicate grounding problems.

## TABLE 2

## 2040 Error Messages

| Error Number | Error Message | Track | Sector |
|---|---|---|---|
| | **STATUS MESSAGES** | | |
| ØØ | OK | ØØ | ØØ |
| Ø1 | FILES SCRATCHED | # FILES | ØØ |
| | **READ ERRORS** | | |
| 2Ø | READ ERROR (block header not found) | T | S |
| 21 | READ ERROR (no synch character) | T | S |
| 22 | READ ERROR (data block not present) | T | S |
| 23 | READ ERROR (checksum error in data block) | T | S |
| 24 | READ ERROR (byte decoding error) | T | S |
| 27 | READ ERROR (checksum error in header) | T | S |
| | **WRITE ERRORS** | | |
| 25 | WRITE ERROR (write-verify error) | T | S |
| 26 | WRITE PROTECT ON | T | S |
| 28 | WRITE ERROR (long data block) | T | S |
| 29 | DISK ID MISMATCH | T | S |
| | **SYNTAX ERRORS** | | |
| 3Ø | SYNTAX ERROR (general syntax) | ØØ | ØØ |
| 31 | SYNTAX ERROR (invalid command) | ØØ | ØØ |
| 32 | SYNTAX ERROR (long line) | ØØ | ØØ |
| 33 | SYNTAX ERROR (invalid file name) | ØØ | ØØ |
| 34 | SYNTAX ERROR (no file given) | ØØ | ØØ |
| 6Ø | WRITE FILE OPEN | ØØ | ØØ |
| 61 | FILE NOT OPEN | ØØ | ØØ |
| 62 | FILE NOT FOUND | ØØ | ØØ |
| 63 | FILE EXISTS | ØØ | ØØ |
| 64 | FILE TYPE MISMATCH | ØØ | ØØ |
| 65 | NO BLOCK | T | S |
| 7Ø | NO CHANNELS | ØØ | ØØ |
| 71 | DIR ERROR | ØØ | ØØ |
| 72 | DISK FULL | ØØ | ØØ |

27. READ ERROR (checksum error in header)
The controller has detected an error in the header of the requested data block. The block has not been read into the DOS memory. This message may also indicate grounding problems.

<u>Write errors: 25, 26, 28, 29</u>

25: WRITE ERROR (write-verify error)
This message is generated if the controller detects a mismatch between the written data and the data in the DOS memory.

26. WRITE PROTECT ON
This message is generated when the controller has been requested to write a data block while the write protect switch is depressed. Typically, this is caused by using a diskette with a write protect tab over the notch.

28. WRITE ERROR (long data block)
The controller attempts to detect the synch mark of the next header after writing a data block. If the synch mark does not appear within a predetermined time, the error message is generated. The error is caused by a bad diskette format (the data extends into the next block), or by hardware failure.

29. DISK ID MISMATCH
This message is generated when the controller has been requested to access a diskette which has not been initialized. The message can also occur if a diskette has a bad header.

<u>Syntax errors: 3Ø, 31, 32, 33, 34</u>

3Ø: SYNTAX ERROR (general syntax)
The DOS cannot interpret the command sent to the command channel. Typically, this is caused by an illegal number of file names, or patterns are illegally used. For example, two file names may appear on the left side of the COPY command.

31: SYNTAX ERROR (invalid command)
The DOS does not recognize the command. The command must start in the first position.

32: SYNTAX ERROR (long line)
The command sent is longer than 4Ø characters.

33: SYNTAX ERROR (invalid file name)
Pattern matching is invalidly used in the OPEN or SAVE command.

34: SYNTAX ERROR (no file given)
The file name was left out of a command or the DOS does not recognize it as such. Typically, a quotation mark (") or colon (:) has been left out of the command.

File errors: 6Ø, 61, 62, 63, 64, 65

6Ø: WRITE FILE OPEN
This message is generated when a write file that has not been closed is being opened for reading.

61: FILE NOT OPEN
This message is generated when a file is being accessed that has not been opened in the DOS. Sometimes, in this case, a message is not generated; the request is simply ignored.

62: FILE NOT FOUND
The requested file does not exist on the indicated drive.

63: FILE EXISTS
The file name of the file being created already exists on the diskette.

64: FILE TYPE MISMATCH
The file type does not match the file type in the directory entry for the requested file.

65: NO BLOCK
This message occurs in conjunction with the B-A command. It indicates that the block to be allocated has been previously allocated. The parameters indicate the next higher in number available track and sector. If the parameters are zero (Ø), then all blocks higher in number are in use.

System errors: 7Ø, 71, 72

7Ø: NO CHANNEL (available)
The requested channel is not available, or all channels are in use. A maximum of five sequential files may be opened at one time to the DOS. Direct access channels may have six opened files.

71: DIR(ectory) ERROR
The BAM does not match the internal count. There is a problem in the BAM allocation or the BAM has been overwritten in DOS memory. To correct this problem, reinitialize the diskette to restore the BAM in memory. Some active files may be terminated by the corrective action.

72. DISK FULL
Either the blocks on the diskette are used or the directory is at its limit (152 entries).

## SIMPLIFYING THE USE OF 2040-RELATED COMMANDS

It has been explained that all disk commands must be preceded with the BASIC **PRINT#** command and enclosed in quotation marks. This is true, but you can program your computer to perform these tasks for you. You also know how to load and run programs stored on diskette. These tasks can be simplified, too.

## Loading the DOS SUPPORT Program

The first file on the TEST/DEMO diskette that comes with your 2040 contains a program called DOS SUPPORT. This program, when loaded into your computer's memory, takes care of the tasks mentioned above. You can use a special LOAD command to load this program into your computer's memory. This command is:

**LOAD "✱",8**

Give this command as your first command immediately after power up. It not only loads DOS SUPPORT, but it initializes Drive Ø and opens the file. This means that you must have the diskette containing DOS SUPPORT in Drive Ø. If you use this command later in the session, it loads the last program successfully accessed with LOAD or SAVE regardless of the diskette involved.

When DOS SUPPORT is loaded into your computer's memory, you should give the RUN command to implement it. Unlike other programs that you call from diskette, it is automatically relocated into the highest available RAM memory location. In this way, it is not lost when you remove it from the computer's memory.

## Using the DOS SUPPORT Symbols: > and @

Once DOS SUPPORT is implemented you no longer have to precede disk commands with PRINT#lfn or enclose them in quotation marks. You need only precede the disk command with either the greater than symbol (>) or the *at* sign (@). The examples in this manual use the > symbol.

**Examples:**

| | | |
|---|---|---|
| >IØ | is the same as | PRINT#1,"IØ" |
| >SØ:FILE1 | is the same as | PRINT#15,"SØ;FILE1" |

No OPEN statement is required before a statement.

The greater than symbol can also be used to load a diskette directory. Usually you load a directory with LOAD"$*dr*",8. This command destroys any program you might have in memory. When you use the > symbol, the directory is printed directly to the screen, thus preserving the data in the computer's memory.

**Examples:**

| | |
|---|---|
| >$Ø | means display the entire directory of Drive Ø. |
| >$1:Q* | means to display all the files on Drive 1 that begin with a Q. |

NOTE: To avoid scrolling of the directory, depress the space bar to stop the listing. To continue the listing of the directory, press any key on the keyboard.

To stop a directory listing and return to BASIC, press $\boxed{\text{RUN STOP}}$

The third use of > is the request of error messages.

**Example:**

```
>          is equivalent to:

           1Ø OPEN 2,8,15
           2Ø INPUT#2,A$,B$,C$,D$
           3Ø PRINTA$,B$,C$,D$
```

## Loading a Program with the /

Use the slash (/) to load a program from diskette. Both diskettes are searched if the drive number is not specified.

**Example:**

/ACCT loads the program ACCT into the computer's memory.

## Loading and Running a Program with ↑

The up arrow (↑) loads a program from a diskette and runs it. Both diskettes are searched if necessary.

**Example:**

↑JDATA loads and runs the program JDATA.

## Special DOS SUPPORT Information

The DOS SUPPORT program has certain limitations. These are:

1. The program must be reaccessed from the disk whenever you reset the computer.
2. You may only give DOS SUPPORT commands when you are communicating with the 2040 in the direct mode. In other words, you cannot use DOS SUPPORT commands in a program.
3. You can print the disk directory on the printer by giving these commands:

| | |
|---|---|
| OPEN 4,4: CMD 4 | Opens device 4 and changes the primary output device to 4. |
| >$ Ø | Prints the directory. |
| PRINT#4: CLOSE 4 | Returns output to the screen and closes the file. |

## Section 4

# ADVANCED DISK PROGRAMMING

This section provides the advanced user with information about DOS structure and disk utility commands. These utility commands provide the programmer with low level functions that may be used for special applications such as special disk handling routines and random access techniques.

## DISK OPERATING SYSTEM (DOS)

The DOS file interface controller is responsible for managing all information between the disk controller and the IEEE-488 bus. Most disk I/O is performed on a pipelined basis, resulting in a faster response to a requested operation.

The file system is organized by channels. You open each channel through the BASIC OPEN statement. When you execute the OPEN statement, the DOS assigns a workspace to the channel and allocates either one or two Disk I/O buffer areas. If either the workspace or the buffer is not available, a NO CHANNEL error is generated. The DOS also uses the channel structure to search the directory, delete files, and copy files.

The common memory between the disk controller and the file interface controller is used as the 256-byte buffer areas. Three of the sixteen buffers are used by the DOS for the Block Availability Maps (BAM), variable space, command channel I/O, and the disk controller's job queue.

The job queue is the vital link between the two controllers. Jobs are initiated by the file side by providing the disk controller with sector header and type of operation information. The disk controller seeks the optimum job and attempts the execution. An error condition is then returned in place of the job command. If the job is not successful, the file side re-enters the job a set number of times, depending on the operation, before generating an error message for the computer.

The secondary address given in the OPEN statement is used by DOS as the channel number. The number you give to a channel is only a reference number that is used to access the work areas, and is in no way significant to the ordering of channels. The LOAD and SAVE statements transmit secondary addresses of Ø and 1, respectively. The DOS automatically interprets these secondary addresses as LOAD and SAVE functions. Unless you want these functions when opening files, you should avoid secondary addresses of Ø and 1. The remaining numbers, 2 through 14, can be used as secondary addresses to open up to five channels for data.

## Special OPEN and CLOSE Statements for Direct Access

The BASIC statement:

```
OPEN 2,8,4,"#"
```

or

```
OPEN 2,8,4,"#12"
```

opens a channel to one buffer, to be used with the block commands. The first available buffer is allocated to channel 4 in the first example. The second example is an attempt to allocate buffer 12 to the channel. If the buffers are not available, a NO CHANNELS error condition is generated. The explicit buffer allocation can be used to reserve a buffer for position dependent code as in the case of an execute command.

You can find the number of the allocated buffer by executing a GET# statement. The byte transmitted is the buffer number. The only time you can get a buffer number is before any write or read operation to that buffer.

The CLOSE statement clears the opened channel and writes the BAM to the diskette that was last used by that channel. It is recommended that to avoid confusion, you limit yourself to accessing one drive with any direct access channel.

## DISK UTILITY COMMAND SET

The 2040 diskette utility command set consists of the following commands:

| Commands | Abbreviations | General Syntax |
|---|---|---|
| **BLOCK-READ** | **B-R** | **"B-R:*ch,dr,t,s*"** |
| **BLOCK-WRITE** | **B-W** | **"B-W:*ch,dr,t,s*"** |
| **BLOCK-EXECUTIVE** | **B-E** | **"B-E:*ch,dr,t,s*"** |
| **BUFFER-POINTER** | **B-P** | **"B-P:*ch,p*"** |
| **BLOCK-ALLOCATE** | **B-A** | **"B-A:*dr,t,s*"** |
| **BLOCK-FREE** | **B-F** | **"B-F:*dr,t,s*"** |
| **memory-write** | **M-W** | **"M-W"*adl/adh/nc/data*** |
| **memory-read** | **M-R** | **"M-R"*adl/adh*** |
| **memory-execute** | **M-E** | **"M-E"*adl/adh*** |
| **USER** | **U** | **"U*i[:parms]*"** |

| | | |
|---|---|---|
| Where: | ***ch*** | is the channel number in DOS: identical to the secondary address in the associated **OPEN** statement. |
| | ***dr*** | is the drive number: Ø or 1 |
| | ***t*** | is the track number: 1 through 35 |
| | ***s*** | is the sector number: Ø through 2Ø. For each track number, the sector range is: |

| Track number | Block or Sector Range | Total |
|---|---|---|
| 1 to 17 | Ø to 2Ø | 21 |
| 18 to 24 | Ø to 19 | 2Ø |
| 25 to 3Ø | Ø to 17 | 18 |
| 31 to 35 | Ø to 16 | 17 |

| | |
|---|---|
| ***p*** | is the pointer position in the buffer |
| ***adl*** | is the low byte of the address* |
| ***adh*** | is the high byte of the address* |
| ***nc*** | is the number of characters: 1 through 34* |
| ***data*** | is the number of bytes of data: 34 maximum |
| ***i*** | is the index to the User Table |
| ***parms*** | (optional) is the parameters associated with the U command. |

*Since these values exist in the 2040 as single bytes, you must request the values with CHR$*(n)*, where *n* is the decimal value of the byte.

As implied in the general syntaxes shown above, these commands may be abbreviated to the first character of each of the key words. Abbreviations *only* are accepted for those commands shown in lower case. The parameters associated with each command are searched for starting at a colon (:), or in the fourth character position if a colon is not present. The example below shows four ways the same Block-Read command may be given.

**Examples:**

```
"BLOCK-READ:2,1,4,Ø"

"B-R2,1,4,Ø"

"B-R"2;1;4;Ø

"B-READ:"2;1;4;Ø
```

Parameters following the key words within quotation marks may be separated by any combination of the following characters:

| Character Name | Keyboard Representation |
|---|---|
| Skip | ⇨ |
| Space | Space bar |
| Comma | , |

The use of these characters allows you to send both ASCII strings and integers.

Parameters not within the confines of quotation marks should be speparated by semicolons (;).

In the following sections, a **PRINT#** is assumed in all examples.

## The BLOCK-READ Command

This diskette utility command provides you with direct access to any block on a diskette in either disk drive. Used with other block commands, a random access file system may be created through BASIC. This command finds the character pointer in the Ø position of the block. When this character is accessed with GET# or INPUT#, and End-or-Identify (EOI) is sent. This terminates an INPUT# and sets the Status Word (ST) to 64 in the computer.

**Example:**

This command reads the block from Drive 1, Track 18, Sector Ø into channel 5's buffer area.

```
"B-R";5;1;18;Ø
```

Read about the USER commands associated with BLOCK-READ on page 44.

## The BLOCK-WRITE Command

When this command is initiated, the current buffer pointer is used as the last character pointer and is placed in the Ø position of the block. The block is written to diskette and the buffer pointer is left in position 1.

**Example:**

This command writes Channel 7's buffer to the block on Drive Ø, Track 35, Sector 1Ø:

```
"B-W";7;Ø;35;1Ø
```

Read about the USER commands associated with BLOCK-WRITE on page 44.

## The BLOCK-EXECUTE Command

This command allows part of the operation system to reside on the diskette. B-E is really a B-R with an addition. The File Interface Controller begins execution of the contents after the block is read into a buffer. Execution must be terminated with a return from the subroutine (RTS) instruction. Future system extensions may be implemented using this technique. User-created functions may also implement this command.

**Example:**

This command reads a block from Drive 1, Track 1, Sector 1Ø into Channel 6's buffer and executes its contents beginning at position Ø in the buffer:

```
"B-E":6;1;1;1Ø
```

## The BUFFER-POINTER Command

This command updates to a new value the pointer associated with the given channel. This is useful when accessing particular fields of a record in a block or, if the block is divided into records, individual records may be set for printing or receiving data.

**Example:**

The command shown here sets Channel 2's pointer to the beginning of the data area in the direct access buffer:

```
"B-P";2;1
```

## The BLOCK-ALLOCATE Command

The appropriate BAM is updated in the DOS memory to reflect the indicated block as used. In future sequential operations, the system does not use the allocated block in sequential operations. The updated BAM is written to diskette upon the closure of a write file or the closure of a direct access channel.

If the block requested has been previously allocated, the error channel indicates the next available block (increasing track and sector numbers) with a NO BLOCK error. If there are no blocks available that are greater in number than the one requested, zeroes are returned as track and sector parameters.

## The MEMORY-WRITE Command

All three MEMORY commands are byte oriented. They are intended for use in machine langauge programs in your computer. BASIC statements may be used to access information through the MEMORY commands by using the **CHR$** function. The system accepts only M-R, M-W, and M-E: no verbose spelling is allowed nor is the use of the colon (:).

The MEMORY-WRITE command provides direct access to the DOS memory. Special routines may be downloaded to the 2040 for execution through this command. Up to 34 bytes may be deposited with each use of the command. The low byte of the address must precede the high byte of the address.

**Example:**

This command writes four bytes to Buffer 2 ($12ØØ or 46Ø8):

```
"M-W"CHR$(ØØ)CHR$(18)CHR$(4)CHR$(32)CHR$(Ø)CHR$(17)CHR$(96)
```

## The MEMORY-READ Command

This command sets up the byte pointed to by the address in the command string. Variables from the DOS or the contents of the buffers may be read with this command. The command changes the contents of the error channel since it is used for transmitting the information to the computer. The next GET# from the error channel (15) transmits the byte. An INPUT# should not be executed after a MEMORY-READ command until a DOS command other than one of the MEMORY commands is executed.

**Example:**

```
"M-R"CHR$(128);CHR$(Ø)
```

## The MEMORY-EXECUTE Command

You can execute subroutines with this command. To return to the DOS, terminate the subroutine with RTS, $6Ø. The following example shows a request for the execution of code located at $318Ø.

**Example:**

```
"M-E";CHR$(128);CHR$(49)
```

## The USER Command

The USER command provides a link to any machine code according to a jump table pointed at by the special user pointer. The second character in the command is used as an index to the table. The ASCII character Ø through 9 or A through Ø may be used. Zero sets the user pointer to a standard jump table that contains links to special routines.

The special USER commands U1 (or UA) and U2 (or UB) can be used to replace the BLOCK-READ and the BLOCK-WRITE commands.

The syntax of U1 is:

**U1:*ch,dr,t,s***

U1 reads a block into a buffer and sets the character count to 255. This allows complete access to all bytes in positions 1 through 255.

The syntax of U2 is:

**U2:*ch,dr,t,s***

U2 writes a buffer to a block on the disk without changing the contents of Position Ø as B-W does. This is useful when a block is to be read in (with B-R) and updated (B-P to the field and PRINT#), then written back to diskette with U2.

Table 3 shows the standard jump table in ROM at the time of power-up.

**TABLE 3**

**Standard Jump Table**

| **User Designation** | **Alternate User Designation** | **Function** |
|---|---|---|
| U1 | UA | BLOCK-READ replacement |
| U2 | UB | BLOCK-WRITE replacement |
| U3 | UC | jump to $1300 |
| U4 | UD | jump to $1303 |
| U5 | UE | jump to $1306 |
| U6 | UF | jump to $D008 |
| U7 | UG | jump to $D00B |
| U8 | UH | jump to $D00E |
| U9 | UI | jump to $D0D5 |
| U: | UJ | power up $E18E |

## RANDOM ACCESS EXAMPLE

Since the BLOCK-ALLOCATE command returns the next available diskette block through the error channel, it can be used in the allocation of records. This feature allows you to create a random file without being concerned with the actual physical structure of the diskette. The allocated blocks must be recorded for reference in the BASIC program, however.

The following random file example demonstrates the block access commands. Notice that the U1 and U2 commands are used. These commands are used since more than one record is stored in a block and it is necessary to manage end-of-record pointers in BASIC. A smaller application might take advantage of the B-R and B-W commands.

This example is built upon a relative record scheme and provides single record access through BASIC programming. Most of the programming below line 2000 is relative record access. The field accessing routines left-justify binary and alpha fields, and right-justify numeric fields.

In a real situation, the program should generate error messages to the operator, or automatically take corrective action such as rounding numbers to fit a field. It would also be possible to add data sorts and searches as well as key fields to the program. Record size, including field markers, must be less than 254 characters. Field size is restricted to 80 characters because of the restrictions of the BASIC INPUT# statement.

Two sequential files are used to support the random access file in this example. Each file bears the name of the file name given in the CREATE file code (1100 to 1180) plus a six-character extension. Since primary file names are ten or less characters, the file names are padded with spaces. The two files are named FILENAME   .DESCR and FILENAME   .KEY01.

The descriptor (   .DESC) contains information on record structure and location. The primary key file (   .KEY01) contains the first field of each record and the relative record number. The example allows the random records to reside on a separate diskette from the sequential support files. This provides added room for random data. The OPEN code (1200 to 1275) requires the disk ID of the random file disk for comparison.

## EXAMPLE

**TO CREATE A FILE:**

1. Load and run RAND 1.0
   - Insert a disk in Drive 0.
   - Open the command channel and initialize the diskette by typing:

     OPEN 15,8,15,"I0" RETURN

   - Load RAND 1.0 by typing:

     LOAD"0:RAND 1.0",8 RETURN

   - Insert a blank diskette into Drive 1, and type:

     PRINT#15,"N1:MAILING LIST RETURN

   - Type: RUN RETURN

2. The computer asks: "DO YOU WISH TO CREATE A FILE"?

   Type: Y RETURN

3. The computer displays: "RANDOM FILE NAME"?

   Enter the file name, i.e.: PHONE LIST RETURN

4. The computer displays: "KEY FILE DRIVE NUMBER"?

   Enter: 1 RETURN

5. The computer displays: "RANDOM FILE NUMBER"?

   Enter: 1 RETURN

6. The computer asks: "ENTER ID OF RANDOM DISK"?

   Type: CS RETURN

7. The computer displays: "NUMBER OF RECORDS"?

   This is the MAXIMUM number of records the file can contain. For this example, enter: 1Ø RETURN

8. The computer asks: "NUMBER OF FIELDS PER RECORD"?

   This is the number of 'items' each field contains. For this example, enter: 4 RETURN

9. The computer displays: "INPUT FIELD NAME, FIELD SIZE, FIELD TYPE"

   TYPES: Ø=BINARY, 1=NUMERIC, 2=ALPHA

   FIELD 1?" enter: NAME,2Ø,2 RETURN

   FIELD 2?" enter: PHONE,15,2 RETURN

   FIELD 3?" enter: ADDRESS,4Ø,2 RETURN

   FIELD 4?" enter: COMMENTS,4Ø,2 RETURN

**TO ADD A RECORD:**

10. The computer displays: "WHOSE RECORD DO YOU WISH TO SEE"?

    Press RETURN

11. The computer displays: "**** ADD RECORD****"

    NAME"? enter the desired name, for example: COMMODORE RETURN

PHONE"? enter the phone number: 408-727-1130 RETURN

ADDRESS"? enter the address: 3330 SCOTT BLVD SANTA CLARA CA. 95050 RETURN

COMMENTS"? enter a comment, for example: MANUFACTURES MICROCOMPUTERS RETURN

12. The computer displays: "WHOSE RECORD DO YOU WISH TO SEE"?

    Press RETURN

13. The computer displays: "***** ADD RECORD*****"

    NAME"? enter the desired name, for example: SMITH RETURN

    PHONE"? enter the phone number: 999-356-1012 RETURN

    ADDRESS"? enter the address: 247 MASSOL DR LOS GATOS CA. 95030 RETURN

    COMMENTS"? enter a comment, for example: MANUFACTURES PERIPHERALS RETURN

14. The computer displays: "WHOSE RECORD DO YOU WISH TO SEE"?

    Press RETURN

15. The computer displays: "***** ADD RECORD*****"

    NAME"? enter the desired name, for example: JONES RETURN

    PHONE"? enter the phone number: 999-268-1795 RETURN

    ADDRESS"? enter the address: 4086 AMBER WAY SAN JOSE CA. 95028 RETURN

    COMMENTS"? enter a comment, for example: MANUFACTURES COMPUTERS RETURN

**TO SEE A RECORD:**

16. The computer displays: "WHOSE RECORD DO YOU WISH TO SEE"?

    Enter: COMMODORE RETURN

**TO CHANGE A RECORD:**

17. After the computer has displayed the record, it asks: "ANY MODS"?

    Enter: YES RETURN

18. The computer asks: "WHICH FIELD"?

Enter the number of the field you wish to change, for example: 4 RETURN

19. The computer displays that field, for example: "MANUFACTURES MICROCOMPUTERS"

? Enter the correct contents of the field, for example: US HEADQUARTERS RETURN

20. The computer again asks if there are: "ANY MODS".

If the record is correct, type: NO RETURN

**GETTING THE DIRECTORY OF LISTINGS:**

21. The computer displays: "WHOSE RECORD DO YOU WISH TO SEE"?

Type: /DIR RETURN

The computer displays the directory.

**ENDING THE PROGRAM:**

22. The computer displays: "WHOSE RECORD DO YOU WISH TO SEE"?

Type: / / RETURN and the program ends.

**RANDOM 1.00**

```
 1 REM RANDOM 1.0
 2 REM SUBROUTINES TO MANAGE RANDOM ACCESS FILES
 3 REM  VARIABLES ARE SET FROM DATA OF DESCRIPTOR FILE & KEY LIST FILES...
 4 REM ...DEFINED BY USER PROGRAM
 5 REM VARIABLES SHOULD REFLECT DESIRED FILE STRUCTURE
 6 REM ALL FUNCTIONS ACT UPON THE VARIABLES DEFINED BELOW
 10 REM
 11 REM **************************
 12 REM
 15 M$=CHR$(13):REM FIELD MARKER
 16 SP$="                                                        "+"":REM SPACE FOR PADDING
 20 C0=2:          REM DIRECT CHANNEL
 21 C1=3:          REM SEQUENTIAL CHANNEL
 25 CC=15:         REM COMMAND CHANNEL
 30 D=0:           REM CURRENT DRIVE #
 31 T=0:           REM CURRENT TRACK #
 32 S=0:           REM CURRENT SECTOR #
 35 DD=0:          REM DESCRIPTOR DRIVE #
 36 RD=0:          REM RANDOM DRIVE #
 40 ID$="":        REM RANDOM DISK ID
 45 NR=0:          REM # RECORDS IN R-FILE
 46 CR=0:          REM CURRENT RECORD #
 47 FR=0:          REM 1ST FREE RECORD UNUSED
 50 NF=0:          REM # FIELDS IN RECORD
 51 CF=0:          REM CURRENT FIELD #
 55 RB=0:          REM # RECORDS PER BLOCK
```

```
56 RS=0:        REM RECORD SIZE IN BYTES
60 NB=0:        REM # BLOCKS IN R-FILE
65 E=0:         REM ERROR FLAG, OK =0
66 REM EN$,EM$,ET$,ES$,ET,ES  ERROR CHANNEL VARIABLES
70 EP=.5/256:   REM INTEGER CORRECTION
75 AS=0:        REM INDEX ARRAY ADDRESSING STRATEGY
76 REM AS=0: USE ARRAY INDEX, AS=1: T&S ARE SET, CR= RECORD OFFSET IN BLOCK
90 REM "A" VARIABLES ARE TEMPORARY
95 DN=8:OPENCC,DN,CC:  REM DN= DEVICE NUMBER
98 GOTO2000:    REM START OF USER PROGRAM
99 REM
100 REM ************************
101 REM RANDOM FILE DIMENSION ROUTINE
102 REM  1ST SET NR, NF & NB
103 REM
105 GOSUB150
110 IFFP%=-1THENRETURN
111 FP%=-1
115 DIM FS%(NF) :REM FIELD SIZE
120 DIM FP%(NF) :REM FIELD POSITION
125 REM          FP%(I)= SUM [FS%(I-1)]
130 DIM FT%(NF) :REM FIELD TYPE: 0:BINARY, 1:NUMERIC, 2:ALPHA
135 DIM FH$(NF) :REM FIELD HEADING
140 DIM F$(NF)  :REM FIELD ARGS-ALPHA,BINARY
145 DIM F(NF)   :REM FIELD ARGS-NUMERIC
146 RETURN
150 IFIT%=-1THENRETURN
151 IT%=-1
155 DIM IT%(NB) :REM TRACK INDEX ARRAY
160 DIM IS%(NB) :REM SECTOR INDEX ARRAY
165 DIM K1$(NR) :REM PRIMARY KEY VALUE
170 DIM RR%(NR) :REM RELATIVE RECORD LIST PER KEY
175 RETURN
200 REM ************************
201 REM UPDATE RECORD, CR
202 REM
205 GOSUB900
210 PRINT#CC,"U1:"C0;D;T;S
215 PRINT#CC,"B-P:"C0;RP
220 FORCF=1TONF
225 GOSUB500
230 NEXTCF
235 PRINT#CC,"U2:"C0;D;T;S
240 GOSUB1000:IFETHEN1900
245 RETURN
300 REM ************************
301 REM READ RECORD, CR
302 REM
305 GOSUB900
310 PRINT#CC,"U1:"C0;D;T;S
315 PRINT#CC,"B-P:"C0;RP
320 GOSUB1000:IFETHEN1900
325 FORCF=1TONF
330 GOSUB600
335 NEXTCF
340 RETURN
400 REM ************************
401 REM UPDATE FIELD(CF) OF RECORD CR, SINGLE FIELD UPDATE
402 REM
405 GOSUB900
410 PRINT#CC,"U1:"C0;D;T;S
415 GOSUB1000:IFETHEN1900
420 PRINT#CC,"B-P:"C0;FP%(CF)+RP
425 GOSUB500 :REM UPDATE FIELD
```

```
430 PRINT#CC,"U2:"C0;D;T;S
435 GOSUB1000:IFETHEN1900
440 RETURN
450 REM ***************************
451 REM READ FIELD(CF) OF RECORD CR, SINGLE FIELD READ
452 REM
455 GOSUB900
460 PRINT#CC,"U1:"C0;D;T;S
465 GOSUB1000:IFETHEN1900
470 PRINT#CC,"B-P:"C0;FP%(CF)+RP
475 GOSUB600 :REM READ FIELD
480 RETURN
500 REM ***************************
501 REM UPDATE FIELD(CF), B-P IS SET
502 REM
510 IFFT%(CF)<>1THEN520
515 A$=RIGHT$(SP$+STR$(F(CF)),FS%(CF)):GOTO530
520 A$=LEFT$(F$(CF)+SP$,FS%(CF))
530 PRINT#C0,A$;M$;
535 RETURN
600 REM ***************************
601 REM READ FIELD(CF), B-P IS SET
602 REM
610 IF FT%(CF) THEN645
615 A1$=""
620 FORJ=1TOFS%(CF)
625 GET#C0,A$:IFA$=""THENA$=CHR$(0)
630 A1$=A1$+A$
635 NEXT:F$(CF)=A1$
640 GET#C0,A$:RETURN
645 INPUT#C0,F$(CF)
650 IFFT%(CF)<>1THEN RETURN
655 F(CF)=VAL(F$(CF)):RETURN
700 REM ***************************
701 REM ALLOCATE ONE BLOCK, T & S =REQUESTED TRACK & SECTOR
702 REM  RETURNED T & S ARE ALLOCATED VALUES  (T=18 IS SKIPPED)
703 REM
710 GOSUB800:IFETHEN1900: REM CHECK T & S
715 PRINT#CC,"B-A:"D;T;S
720 INPUT#CC,EN,EM$,ET,ES
725 IFEN=0THENRETURN
730 IFEN<>65THEN1900
735 IFET=18THENT=19:S=0:GOTO715
736 T=ET:S=ES
740 GOTO715
750 REM ***************************
751 REM FREE ONE BLOCK, T & S = TRACK & SECTOR
752 REM
760 GOSUB800:IFETHEN1900: REM CHECK T & S
770 PRINT#CC,"B-F:"D;T;S
780 INPUT#CC,EN,EM$,ET,ES
785 IFEN=0THENRETURN
790 GOTO1900
800 REM ***************************
801 REM CHECK MAX SECTOR
802 REM
810 IFT>35THEN1900
820 E=0:IFT=0THEN=40:GOTO1900
840 A3=16:IFT>30THEN880
850 A3=17:IFT>24THEN880
860 A3=19:IFT>17THEN880
870 A3=20
880 IFS>A3THEN1900
```

```
890 RETURN
900 REM ****************************
901 REM SET RECORD'S TRACK, SECTOR & RECORD POINTER FROM INDEX ARRAYS
902 REM
905 D=RD
910 E=0
915 IFAS=-1THENRP=CR*RS+1:GOTO950
920 RP=INT((CR-1)/RB+EP):IFRP>NB OR RP<0THENEN=41:GOTO1900
930 T=IT%(RP):S=IS%(RP)
940 RP=INT(((CR-1)/RB-RP+EP)*RS*RB)+1
950 IFRP>254THEN EN=41:GOTO1900
960 RETURN
1000 REM ****************************
1001 REM INPUT 2040 ERROR STATUS
1002 REM
1005 INPUT#CC,EN$,EM$,ET,ES
1010 EN=VAL(EN$):E=0
1015 IF EN$="00" THEN RETURN
1017 ET$=STR$(ET):ES$=STR$(ES)
1020 IFEN$<>RIGHT$("0"+MID$(STR$(EN),2),2)THEN1070
1030 IF EN=1 THEN EM$= ET$+" "+EM$: RETURN
1035 E=E+1
1040 EM$="▒"+EN$+"■ "+EM$
1050 IF EN<30 OR EN=65 THEN EM$=EM$+" ON "+ET$+", "+ES$
1060 RETURN
1070 EM$="▒SYSTEM NOT RESPONDING PROPERLY"
1080 EM$=EM$+EN$+EM$+ET$+ES$
1085 E=E+1
1090 RETURN
1100 REM ****************************
1101 REM CREATE DESCRIPTOR FILE
1102 REM INPUT: F$= FILENAME
1103 REM        ID$,NR,NF,FS%(),FT%(),FH$()
1104 REM        DD= DESCRIPTOR FILE DRIVE #
1105 REM        RD= RANDOM DISK DRIVE #
1106 REM DRIVES MUST BE INITIALIZED
1109 REM
1110 RS=1:D=RD
1115 FORA0=1TONF:FP%(A0)=RS:RS=FS%(A0)+RS+1:NEXT:RS=RS-1
1116 RB=INT(254/RS+EP)
1120 OPENC0,DN,C0,"#":GOSUB1000:IFETHEN1900
1121 GOSUB1280
1122 PRINT#CC,"B-P:"C0;1
1123 FORA0=1TORB:FORA1=1TONF
1124 PRINT#C0,LEFT$(SP$,FS%(A1));M$;
1126 NEXTA1,A0
1130 NB=INT(NR/RB+EP):IF(NR/RB-NB)*RB>=1THENNB=NB+1
1135 T=1:S=0:GOSUB150
1140 FORA0=0TONB-1:GOSUB710:IFETHEN1900
1145 IT%(A0)=T:IS%(A0)=S:GOSUB430:NEXT
1150 GOSUB710
1152 PRINT#CC,"B-P:"C0;1
1155 PRINT#C0,NR;M$;1;M$;NB;M$;RS;M$;RB;M$;NF;M$;
1160 PRINT#CC,"B-W:"C0;D;T;S
1165 A$=STR$(DD)+":"+LEFT$(F$+SP$,10)+".DESCR,U,W"
1166 OPENC1,DN,C1,A$
1167 GOSUB1000:IFETHEN1900
1168 PRINT#C1,ID$;M$;T;M$;S;M$;
1170 FORA0=1TONF:PRINT#C1,CHR$(FS%(A0));CHR$(FT%(A0));FH$(A0);M$;:NEXT
1175 FORA0=0TONB-1:PRINT#C1,CHR$(IT%(A0));CHR$(IS%(A0));:NEXT
1180 CLOSEC1:CLOSEC0:RETURN
1200 REM ****************************
1201 REM OPEN RELATIVE FILE
1202 REM INPUT: F$= FILENAME
```

```
1203 REM        DD= DESCRIPTOR FILE DRIVE #
1204 REM        RD= RANDOM DISK DRIVE #
1205 REM DRIVES MUST BE INITIALIZED
1209 REM
1210 A$=STR$(DD)+":"+LEFT$(F$+SP$,10)+".DESCR,U,R"
1215 OPENC1,DN,C1,A$:GOSUB1000:IFETHEN1900
1220 INPUT#C1,ID$,T,S
1225 OPENC0,DN,C0,"#":GOSUB1000:IFETHEN1900
1226 GOSUB1280
1227 PRINT#CC,"B-R:";C0;RD;T;S:GOSUB1000:IFETHEN1900
1230 INPUT#C0,NR,FR,NB,RS,RB,NF
1235 GOSUB100:FT%(0)=T:FS%(0)=S
1240 FORA0=1TONF:GOSUB1298:FS%(A0)=ASC(A$)
1245 GOSUB1298:FT%(A0)=ASC(A$)
1250 INPUT#C1,FH$(A0):NEXT
1255 FORA0=0TONB-1:GOSUB1298:IT%(A0)=ASC(A$)
1260 GOSUB1298:IS%(A0)=ASC(A$):NEXT
1265 GOSUB1000:IFETHEN1900
1270 CLOSEC1
1275 RETURN
1280 PRINT#CC,"U1:";C0;RD;",18,0":GOSUB1000:IFETHEN1900
1285 PRINT#CC,"B-P:";C0;162
1286 GET#C0,A$,A1$:A$=A$+A1$:IFID$<>A$THENEN=43:EM$="WRONG RAND DISK":GOTO1900
1290 RETURN
1298 GET#C1,A$:IFA$=""THENA$=CHR$(0)
1299 RETURN
1400 REM **************************
1401 REM CLOSE RELATIVE FILE
1402 REM INPUT: VARIABLES FROM OPEN SHOULD BE VALID
1409 REM
1410 PRINT#CC,"B-P:"C0;1
1420 PRINT#C0,NR;M$;FR;M$;NB;M$;RS;M$;RB;M$;NF;M$;
1430 PRINT#CC,"B-W:"C0;D;FT%(0);FS%(0)
1440 CLOSEC0
1490 RETURN
1900 E=E+1:RETURN
2000 INPUT"■■■DO YOU WISH TO CREATE A FILE  N■■■";A$:IFLEFT$(A$,1)<>"Y"THEN2100
2001 INPUT"■RANDOM FILE NAME";F$
2002 INPUT"KEY FILE DRIVE NUMBER";DD
2003 INPUT"RANDOM FILE DRIVE NUMBER";RD
2005 INPUT"ENTER ID OF RANDOM DISK  __■■■■";ID$:ID$=LEFT$(ID$,2)
2006 INPUT"NUMBER OF RECORDS";NR
2007 INPUT"NUMBER OF FIELDS PER RECORD";NF
2010 GOSUB110
2015 PRINT"■ INPUT FIELD NAME,FIELD SIZE,FIELD TYPE"
2016 PRINT"    TYPES: 0=BINARY, 1=NUMERIC, 2=ALPHA■"
2019 RS=0
2020 FORI=1TONF:PRINT"FIELD";I,:INPUTFH$(I),FS%(I),FT%(I):RS=FS%(I)+RS+1:NEXT
2025 A$="I":IFDD=RDTHENA$="I"+STR$(DD)
2030 PRINT#CC,A$
2040 GOSUB1100:IFETHEN3900
2050 OPEN4,8,4,STR$(DD)+":"+LEFT$(F$+SP$,10)+".KEY01,U,W"
2055 PRINT#4,0;M$;:CLOSE4
2090 GOTO2120
2100 REM OPEN RANDOM FILE FOR ACCESS
2103 INPUT"■RANDOM FILE NAME";F$
2105 INPUT"KEY FILE DRIVE NUMBER";DD
2110 INPUT"RANDOM FILE DRIVE NUMBER";RD
2120 GOSUB1200:IFETHEN3900
2140 OPEN4,8,4,STR$(DD)+":"+LEFT$(F$+SP$,10)+".KEY01,U"
2142 INPUT#4,RR:IFRR=0THEN2147
2145 FORI=1TORR:INPUT#4,K1$(I),RR%(I):NEXT
2147 CLOSE4
```

```
 2150 PRINT"]QQ▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶SAMPLE RANDOM ACCESSQ"
 2155 PRINT"TYPE // TO QUITQ"
 2156 PRINT"(HIT RETURN TO ADD RECORD)"
 2160 PRINT"QQWHOSE RECORD DO YOU"
 2161 INPUT"WISH TO SEE    ■■■";RR$
 2165 IFRR$=" "THEN2310
 2167 IFRR$="//"THEN2400
 2168 IFRR$="/DIR"THENGOSUB4000:GOTO2160
 2170 FORII=1TORR:IFK1$(II)<>RR$THENNEXT:GOTO2300
 2175 CR=RR%(II):GOSUB300
 2180 FORI=1TONF:PRINTI;")"FH$(I)":",F$(I):NEXT:PRINT
 2185 FF=0
 2190 INPUT"ANY MODS    ■■■";A$:IFLEFT$(A$,1)<>"Y"THEN2220
 2195 INPUT"WHICH FIELD";A
 2200 PRINT"   "F$(A):PRINT"]";:INPUTF$(A):F(A)=VAL(F$(A))
 2210 FF=1:GOTO2190
 2220 IFFF=0THEN2160
 2222 IFA=1THENK1$(II)=F$(A)
 2225 GOSUB200
 2230 GOTO2160
 2300 PRINT"QRECORD NOT PRESENT"
 2305 INPUT"DO YOU WISH TO ADD";A$:IFLEFT$(A$,1)<>"Y"THEN2160
 2310 PRINT"Q**** ADD RECORD ****QQ"
 2312 IFFR>NRTHEN2500
 2315 CR=FR:FR=FR+1:RR=RR+1
 2320 FORI=1TONF:PRINTFH$(I);:INPUTF$(I):F(I)=VAL(F$(I)):NEXT
 2330 GOSUB200
 2340 K1$(RR)=F$(1):RR%(RR)=CR
 2350 GOTO2160
 2400 REM CLOSE RAND FILE
 2405 GOSUB1400
 2410 OPEN4,8,4,"@"+STR$(DD)+":"+LEFT$(F$+SP$,10)+".KEY01,U,W"
 2420 GOSUB1000:IFETHEN3900
 2430 PRINT#4,RR;M$;
 2440 FORI=1TORR:PRINT#4,K1$(I);M$;RR%(I);M$;:NEXT
 2445 GOSUB1000:IFETHEN3900
 2450 CLOSE4
 2455 GOSUB1000:IFETHEN3900
 2500 PRINT"THE FILE IS FULL, NO ADDITIONAL RECORDS MAY BE ADDED"
 2510 GOTO2160
 3900 PRINTE,EM$:STOP
 4000 FORDI=0TONR:PRINTK1$(DI):NEXT:RETURN
READY.
```

# APPENDIX A

## SUGGESTED READING
(USA produced)

**Hands-On Basic with a Pet.** H. D. Peckham. McGraw Hill, 1979

**Entering BASIC.** J. Sack and J. Meadows. Science Research Associates, 1973

**BASIC: A Computer Programming Language.** C. Pegels, Holden-Day, Inc., 1973

**BASIC Programming.** J. Kemeny and T. Kurtz, Peoples Computer Co., 1010 Doyle (P. O. Box 3100), Menlo Park, CA 94025, 1967

**BASIC FOR HOME COMPUTERS.** Albrecht, Finkle and Brown. Peoples Computer Co., 1010 Doyle (P. O. Box 3100), Menlo Park, CA 94025, 1973

**A Guided Tour of Computer Programming in BASIC.** T. Dwyer, Houghton Mifflin Co., 1973

**Programming Time Shared Computer in BASIC.** Eugene H. Barnett. Wiley-Interscience L/C 72-175789 ($12.00)

**Programming Language #2.** Digital Equipment Corp., Maynard, MA 01754

**101 BASIC Computer Games.** Software Distribution Center. Digital Equipment Corp., Maynard, MA 01754 ($7.50)

**What to Do After You Hit Return.** Peoples Computer Co., 1010 Doyle (P. O. Box 310), Menlo Park, CA 94025 ($6.95)

**Basic BASIC.** James S. Coan, Hyden Book Co., Rochelle Park, NJ

**WORKBOOKS 1-5.** T. I. S., P. O. Box 921, Los Almos, NM 87544

**Programming the 6502.** R. Zaks Sybex, 1978

**24 Tested, Ready-to-Run Game Programs in Basic.** K. Tracton, Tab Books, 1978

**Some Basic Programs.** M. Borchers and R. Poole, Osborne & Assoc. Inc., 1978

**Basic Programming for Business.** I. H. Forkner, Prentice-Hall, 1977

**The Channel Data Book.** B. Lewis, 5960 Mandarin Ave., Goleta, CA 93017, 1978

# APPENDIX B

## VARIABLE ABBREVIATIONS USED IN COMMAND SYNTAXES

| ABBREVIATION | MEANING |
|---|---|
| ***commandstring*** | Diskette handling or file handling command |
| ***datastring*** | Data to be stored on diskette |
| ***ddr*** | Destination drive: Ø or 1 |
| ***dn*** | Device number: must be 8 for the 2040 unless hardwiring is altered |
| ***dr*** | Drive number: Ø or 1 |
| ***dskname*** | Disk name: up to 16 characters |
| ***fn*** | File name: up to 16 characters |
| ***ft*** | File type: SEQ, PRG, or USR only |
| ***id*** | Disk identifier: two-characters |
| ***lfn*** | Logical file number |
| ***mode*** | Transmission mode: READ or WRITE |
| ***sa*** | Secondary address. Used to transmit channel number: 1 through 15 |
| ***sdr*** | Source drive: Ø or 1 |
| ***vname*** | variable designations — may be one, or several designations separated by commas or semicolons |

# APPENDIX C

## EXAMPLE PROGRAM — READING & WRITING SEQUENTIAL FILES

```
SEQUENTIAL 1.00

 1 REM ************************
 2 REM *       EXAMPLE        *
 3 REM *  READ AND WRITE A    *
 4 REM *  SEQUENTIAL DATA     *
 5 REM *  FILE USING DRIVE 0 *
 9 REM ************************
 10 PRINT"██INITIALIZE DISK"
 20 DIMA$(25):REM                         SET A$ ARRAY
 30 DIMB(25):REM                          SET B ARRAY
 40 OPEN15,8,15:REM                       OPEN THE COMMAND CHANNEL
 50 PRINT#15,"I0":REM                     INITIALIZE DRIVE ZERO
 60 GOSUB 1000:REM                        READ THE ERROR CHANNEL
 70 CR$=CHR$(13):REM                      SET STRING CR$ TO A CARRIAGE RETURN
 90 PRINT"█WRITE TEST FILE"
 100 REM ************************
 101 REM *                      *
 102 REM *   WRITE TEST FILE    *
 103 REM *                      *
 105 REM ************************
 110 OPEN2,8,2,"@0: TEST FILE ,S,W":REM OPEN LOGICAL FILE 2 ON DISK 8 TO
 111 REM                                  CHALLEL 2 REPLACE DATA FILE NAMED
 112 REM                                  TEST FILE WITH SEQUENTIAL WRITE
 115 GOSUB 1000:REM                       READ THE ERROR CHANNEL
 120 INPUT"A$,B";A$,B:REM                 INPUT NAME, NUMBER INTO A$ AND B
 130 IFA$="END"THEN 160:REM               STOP THE DATA INPUT
 140 PRINT#2,A$","STR$(B)CR$;:REM         PRINT TO THE DISK
 145 GOSUB 1000:REM                       READ THE ERROR CHANNEL
 150 GOTO 120
 160 CLOSE 2:REM                          CLOSE TEST FILE
 200 REM ************************
 201 REM *                      *
 202 REM *   READ TEST FILE     *
 203 REM *                      *
 205 REM ************************
 206 PRINT"█READ TEST FILE"
 210 OPEN2,8,2,"0: TEST FILE ,S,R":REM  OPEN LOGICAL FILE 2 ON DISK 8 TO
 211 REM                                  CHANNEL 2 NAMED TEST FILE WITH
 212 REM                                  SEQUENTIAL READ
 215 GOSUB 1000:REM                       READ THE ERROR CHANNEL
 220 INPUT#2,A$(I),B(I):REM               READ STRING INTO STRING ARRAY A$
 221 REM                                  AND NUMBER INTO ARRAY B
 224 RS=ST:REM                            STORE THE DISK STATUS
 225 GOSUB 1000:REM                       READ THE ERROR CHANNEL
 230 PRINTA$(I),B(I):REM                  PRINT WHAT WAS READ
 240 IFR S=64 THEN 300:REM                CHECK FOR END OF FILE STATUS
 250 IF RS<>0 THEN 400:REM                CHECK FOR ERROR IN FILE STATUS
 260 I=I+1:REM                            ADD 1 TO ARRAY POINTER
 270 GOTO 220
 300 CLOSE 2:REM                          CLOSE TEST FILE
 310 END:REM                              END THE PROGRAM EXECUTION
 400 PRINT"█BAD█DISK█STATUS█IS█"RS
 410 CLOSE 2:REM                          CLOSE TEST FILE
 420 END:REM                              END THE PROGRAM EXECUTION
 1000 REM ************************
 1001 REM *   READ THE ERROR     *
 1002 REM *       CHANNEL        *
 1005 REM ************************
 1010 INPUT#15,EN$,EM$,ET$,ES$:REM        READ ERROR
 1011 REM                                 EN$ IS THE ERROR NUMBER
 1012 REM                                 EM$ IS THE ERROR MESSAGE
 1013 REM                                 ES$ IS THE ERROR SECTOR
 1020 IF EN$="00" THEN RETURN:REM         RETURN TO MAIN LOGIC IF NO ERRORS
 1030 PRINT"█ERROR ON DISK":REM           PRINT THE ERROR
 1040 PRINTEM$,EN$,ET$ES$:REM             PRINT THE ERROR
 1050 CLOSE 2:REM                         CLOSE TEST FILE
 1060 END:REM                             END THE PROGRAM EXECUTION
READY.
```

**Commodore Business Machines, Inc.**
3330 Scott Blvd.
Santa Clara, CA 95050

**Commodore Business Machines Limited**
3370 Pharmacy Avenue
Agincourt, Ontario, Canada M1W2K4

**Commodore Business Machines (UK) Limited**
360 Euston Road
London NW1 3BL, England

**Commodore Buromaschinen GmbH**
Frankfurter Strasse 171-175
6078 Neu Isenburg
West Germany

**Commodore Japan Limited**
Taisei-Denshi Building
8-14 Ikue 1-ChomeAsahi-Ku, Osaka 535, Japan

**Commodore Electronics (Hong Kong) Ltd.**
Watsons Estates
Block C, 11th floor
Hong Kong, Hong Kong

**P/N 320830-4**