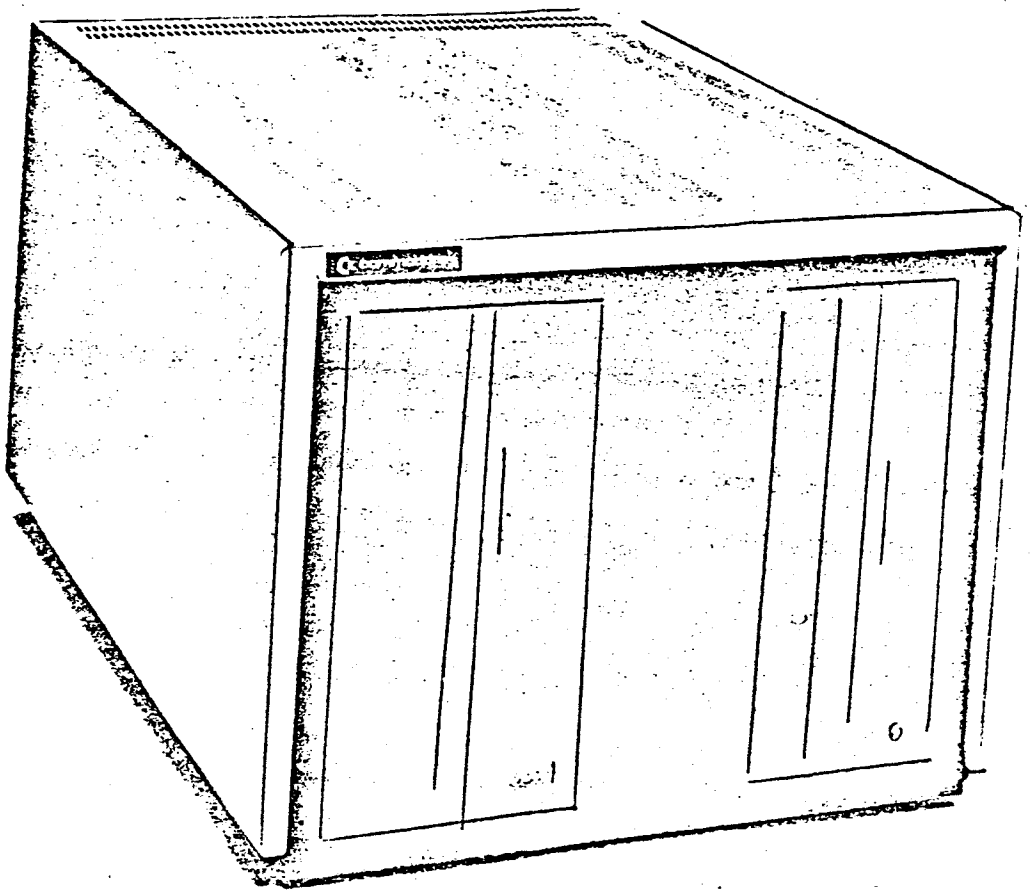
CBM EIGHT-INCH FLOPPY DISK DRIVES

USER'S MANUAL

MODELS 8061 AND 8062



COMMODORE

CBM EIGHT-INCH FLOPPY DISK DRIVES

USER'S MANUAL

MODELS 8061 AND 8062

P/N 324005

- Second Preliminary Draft Copy -

MAY 1981

NOTICE

The information in this manual is not complete, and it has not been through the review cycle. This is therefore preliminary information for reference only.

COMMODORE BUSINESS MACHINES INC.
3330 Scott Blvd.
Santa Clara Ca. 95051

WARRANTY DISCLAIMER

The computer programs supplied on the magnetic diskette with this manual are provided to you, their user, with no warranty of any kind. Although due care has been taken to ensure the correct operation of these programs, no representation is made about their fitness for any particular use or about the accuracy of their results. Commodore Business Machines, Inc., its distributors, retailers, and agents thus assume no responsibility and accept no consequential, incidental, or other liability arising from the use of these programs. Some states do not allow the exclusion or limitation or implied warranties or liability for incidental or consequential damages, so the above limitations may not apply to you.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

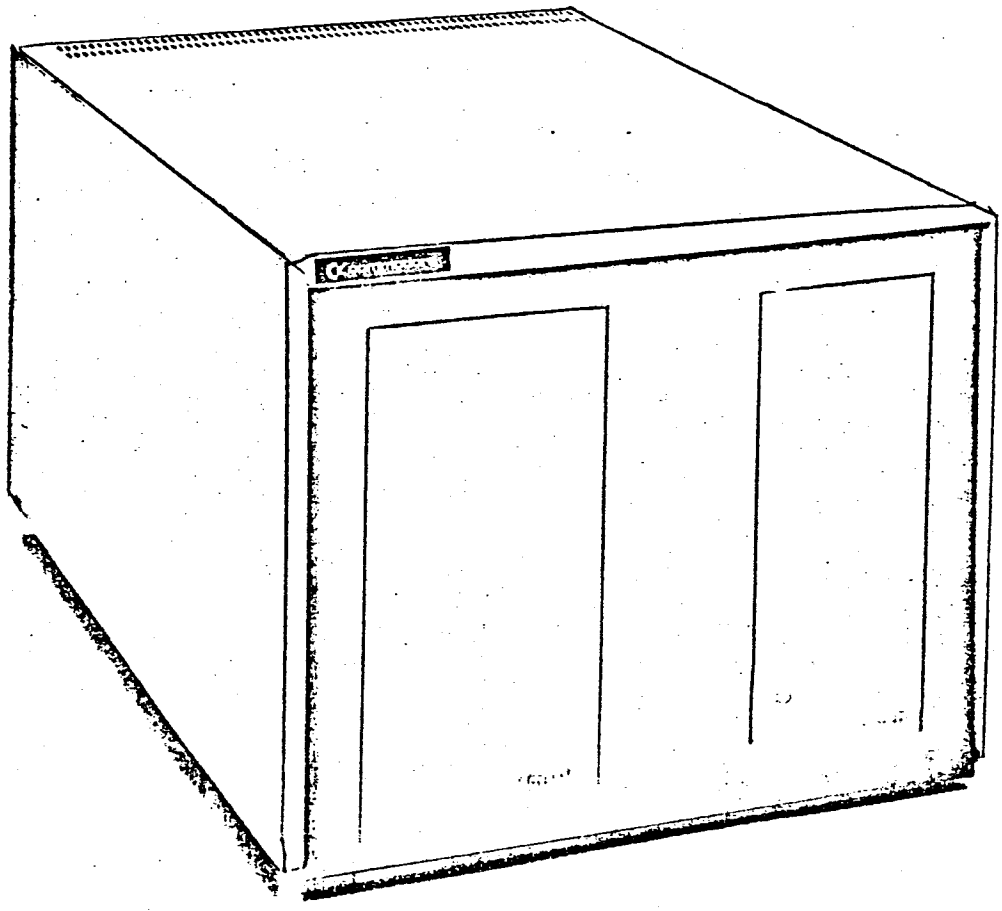


Figure 1-1. 8061/8062 Eight-Inch Floppy Disk Drive

Chapter 1

GENERAL DESCRIPTION

INTRODUCTION

The 8061 and 8062 are eight-inch, dual floppy disk drives. The 8061 contains two single-sided disk drives; the 8062 contains two double-sided disk drives, but has both double-sided and single-sided disk capabilities.

Wide spread use and reliabilty have made the eight-inch floppy drive the standard for data interchange between different systems. These advantages, coupled with the large storage capacity, will indeed enhance your system.

THE MANUAL

This manual contains the installation, operation, and basic programming instructions for the 8061/8062 Eight-Inch Floppy Disk Drives. See Figure 1. The manual is organized as follows:

Chapter 1 provides a general description of the manual and the 8061/8062 Eight-Inch Floppy Disk Drive including a table of specifications.

Chapter 2 presents instructions to unpack, install, cable, and power up the 8061/8062 Disk Drives.

Chapter 3 discusses general operation of the 8061/8062 Disk Drive including handling, formatting, and validating disks.

Chapter 4 describes space allocation on the disk, and BASIC commands for manipulating the directories and files on the disk.

Chapter 5 discusses BASIC commands for handling data and includes a few short programs to handle data.

Chapter 6 describes DOS memory access commands.

Chapter 7 describes advanced file handling techniques for relative and sequential files.

Chapter 8 describes specific commands from the Disk Operating System (DOS) support program.

Chapter 9 contains a list of error messages, and a programmer's quick reference guide.

8061/8062 GENERAL FEATURES

The 8061/8062 operates in the 'native format' mode reading and writing disks formatted by the "formatter" program. The native format is the drive's normal format mode. The formatter program is described in Ghapter 3.

Native-format disks allow full use of the commands described in this manual. Disks written in the data-exchange format, however, must be translated into the native format by the "ibmcopy" program. The data-exchange format is described under Pre-formatted Disks in Chapter 3.

MECHANICAL DESCRIPTION

The drives and the electronic circuitry are mounted on a chassis enclosed by a 10.5" x 13" x 19.5" cover. The cover protects the equipment from accidental damage and the operator from high voltage. The electronics control the drive mechanism and, under control of DOS, communicates with the computer.

CONTROLS AND INDICATORS

The only manual control for the 8061/8062 drive is the power on and off switch located on the back panel of the unit. See Figure 2-1. Power is applied to the drive when the white dot on the power on/off switch shows.

The two LED indicators on the front of the drive display the operating status of the drive. Blinking or continuously lit LEDs indicate a malfunction in the 8061/8062.

UTILITY DISK

The Utility Disk supplied with the 8061/8062 contains the following programs:

1. The "formatter" program that formats a new disk in the native-format mode.

2. The "ibmcopy" program that copies data-exchange disks and translates them into the native format, which essentially allows the user to read from or write to an IBM disk.

3. The "validate" program that executes the validate command and lists those files that have unreadable sectors.

RELATED DOCUMENTS

The following documents contain information relative to the operation of the 8061/8062 disk drives:

1. Commodore Business Computer User's Guide, part number 320894.

2. User's Reference Manual Commodore BASIC Version 4.0, part number 321604.

3. IBM* Diskette General Information Manual, IBM part number GA21-9182-2.

*IBM is a registered trade mark of International Business Machines.

SPECIFICATIONS

Table 1-1 summarizes the specifications for the 8061/8062 disk drives. The 8061 and 8062 floppy drives are used world wide and the power requirements differ in various locations. Power requirements for each drive is specified on the back of the drive. Table 1-1 lists the power requirements for both the 8061 and 8062 disk drives.

Table 1-1. 8061 and 8062 Disk Drive Specifications

Dimensions:

Height	10.5"	(26.67 cm)
Width	13.0"	(33.02 cm)
Depth	19.5"	(49.53 cm)
Weight	53.0 lbs	(24.03 kg)

Media:

8061/8062	8" Single Sided, Double Density, Soft Sectored
8062	8" Double Sided, Double Density, Soft Sectored

Storage:

Single Sided	1.6 Megabytes/Two Diskettes 3106 Blocks/One Diskette
Double Sided	3.2 Megabytes/Two Diskettes 6215 Blocks/One Diskette

Power Requirements:

Unit	Volts	Amps	Freq.	Comments	
8061	117V	1.2A	60Hz	UL	(USA)
	117V	1.2A	60Hz	CSA	(CANADA)
	100V	1.3A	50Hz	JIS	(JAPAN)
	100V	1.3A	60Hz	JIS	(JAPAN)
	220V	0.75A	50Hz	VDE	(GERMANY)
	240V	0.75A	50Hz	BSI	(UK)
	220V	0.75A	50Hz	SEV	(SWITZERLAND)
	220V	0.75A	50Hz	SEMKO	(SWEDEN)
8062	117V	1.2A	60Hz	UL	(USA)
	117V	1.2A	60Hz	CSA	(CANADA)
	100V	1.2A	50Hz	JIS	(JAPAN)
	100V	1.2A	60Hz	JIS	(JAPAN)
	220V	0.75A	50Hz	VDE	(GERMANY)
	240V	0.75A	50Hz	BSI	(UK)
	220V	0.75A	50Hz	SEV	(SWITZERLAND)
	220V	0.75A	50Hz	SEMKO	(SWEDEN)

Chapter 2

INSPECTION AND INSTALLATION

INTRODUCTION

This chapter contains instructions to inspect, unpack, cable and apply power to the 8061/8062.

INSPECTION/UNPACKING/PACKING INSTRUCTIONS

Before unpacking the 8061/8062 drive, inspect the carton for signs of rough handling during shipment. If the carton shows signs of rough handling, special care should be taken to inspect each enclosed item as it is removed. Save the packing material, for reshipment, as you unpack the unit. Remove contents and check against the following list to make certain you have removed all of the contents:

1. 8061/8062 Eight-Inch Disk Drive, part number 8062001.
2. 8061/8062 Eight-Inch Disk Drive User's Manual, part number 324005.
3. Utility Diskette
4. Power Cable, part number 8062007-01.

If any item is damaged or missing, please notify your Commodore dealer immediately.

INSTALLATION

The following procedures check the initial power turn-on with the 8061/8062 out of the system. After successful power turn-on, the drive is cabled to the system and the diagnostic program is performed.

INITIAL POWER TURN ON

The 8061/8062 drive is factory tested and ready to use. Perform the following turn-on procedures to ensure that the drive operates correctly before connecting it to the system.

1. Turn power OFF (white dot not showing) on the 8061/8062 and the computer.
2. Connect the power cord to the back of the 8061/8062 and to an appropriate ac power source; 117Vac, 60Hz.

3. Turn the computer power ON and verify that it is working properly.

4. Turn the 8061/8062 power ON (white dot showing).

Both LEDS on the front panel will light and then go out. If the LEDs remain on or flash continuously, turn power OFF. Wait one minute and try again. Diagnostics are performed inside the drive on power up. Flashing LEDs mean that the diagnostics detected a problem. Continously lit LEDs mean that the diagnostic routine was not completed. If an LED remains lit, or all LEDs flash continuously, contact your Commodore dealer.

CABLING THE 8061/8062 TO THE COMPUTER

One of the following two cables are required to interface the 8061/8062 to the computer. These cables can be obtained from your Commodore dealer.

1. Use the CBM-to-IEEE cable (part number 320101) if the disk drive is the only peripheral device connected to your computer.

2. Use the IEEE-to-IEEE cable (part number 905080) if other peripheral units are connected along with the disk drive.

Follow these steps to cable the 8061/8062 to your computer:

1. Turn power OFF at the computer and the 8061/8062.

2. Place the 8061/8062 in a convenient location close to the computer. See Figure 2-1.

3. Connect the PET-to-IEEE cable between the IEEE-488 interface connector on the computer and the connector on the back of 8061/8062. If additional IEEE devices are to be connected, the IEEE-to-IEEE cable(s) must be used.

4. Turn computer power ON; white dot is showing.

5. Turn the power to all other peripherals ON in the system.

6. Turn the 8061/8062 power ON; white dot is showing.

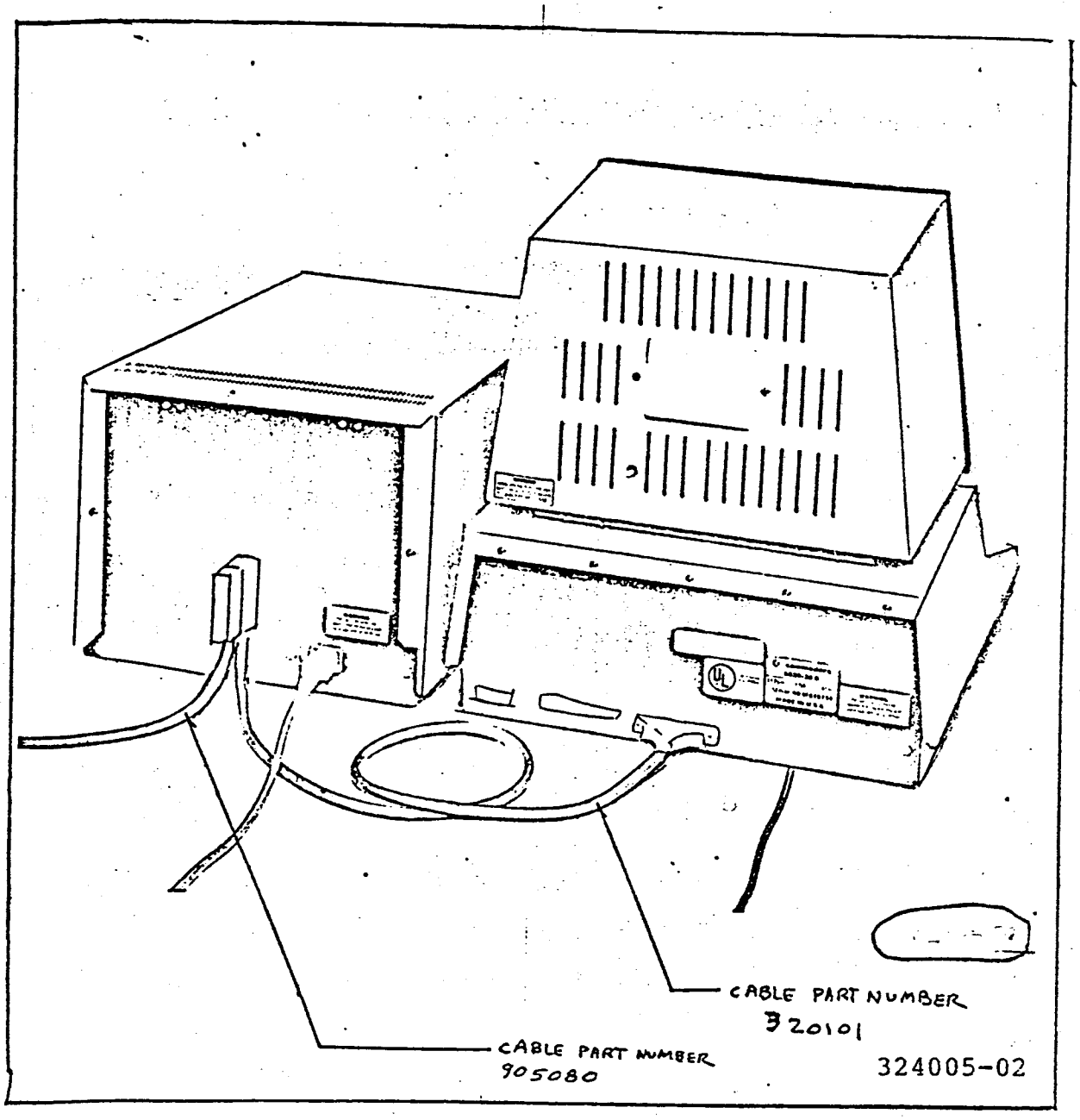


Figure 2-1. System Interconnection Diagram.

NOTE

Both LEDS on the front panel will light and then go out. If the LEDs remain on or flash continuously, turn power OFF. Wait one minute and try again. Diagnostics are performed inside the drive on power up. Flashing LEDs mean that the diagnostics detected a problem. Continously lit LEDs mean that the diagnostic routine was not completed. If an LED remains lit, or all LEDs flash continuously, contact your Commodore dealer.

If the problem persists, disconnect the other devices attached to the IEEE bus. This will assure that a problem related to another device does not affect the disk drive.

Chapter 3

GENERAL DISK DRIVE OPERATION

INTRODUCTION

The following paragraphs discuss general operation of the 8061/8062 disk drive including handling of the disks, initializing, formatting and validating the disks.

DISK HANDLING

The disk stores valuable data and care should be taken when handling it. The following rules will prevent loss of data:

1. Do not touch the surface of the disk.
2. Store the disk in its protective jacket away from magnetic fields and heat. Do not lay the disk on top of the 8061/8062 disk drive.
3. Do not write on the disk label or protective jacket with a hard tipped-pen; use a felt-tipped pen.
4. Do not clean the disk surface.
5. Do not use head cleaning diskettes in the 8061/8062.

Disk Insertion

When the unit has been cabled into the system successfully and the computer is ready, a disk can be inserted into either of the drives. To insert a disk, perform the following procedure:

NOTE

To write on the 8061/8062 disk, the write-protect notch must be covered.

1. Hold the disk so that the label side is to your left and the write protect notch is away from you. See Figure 1.
2. Insert the disk into either drive until a click is heard.
3. Observe the LED's as you close the door. The LED's will flash as follows:
   a. A momentary flash for a correctly formatted disk.
   b. LED remains on for a short period for an unformatted or a pre-formatted IBM disk.

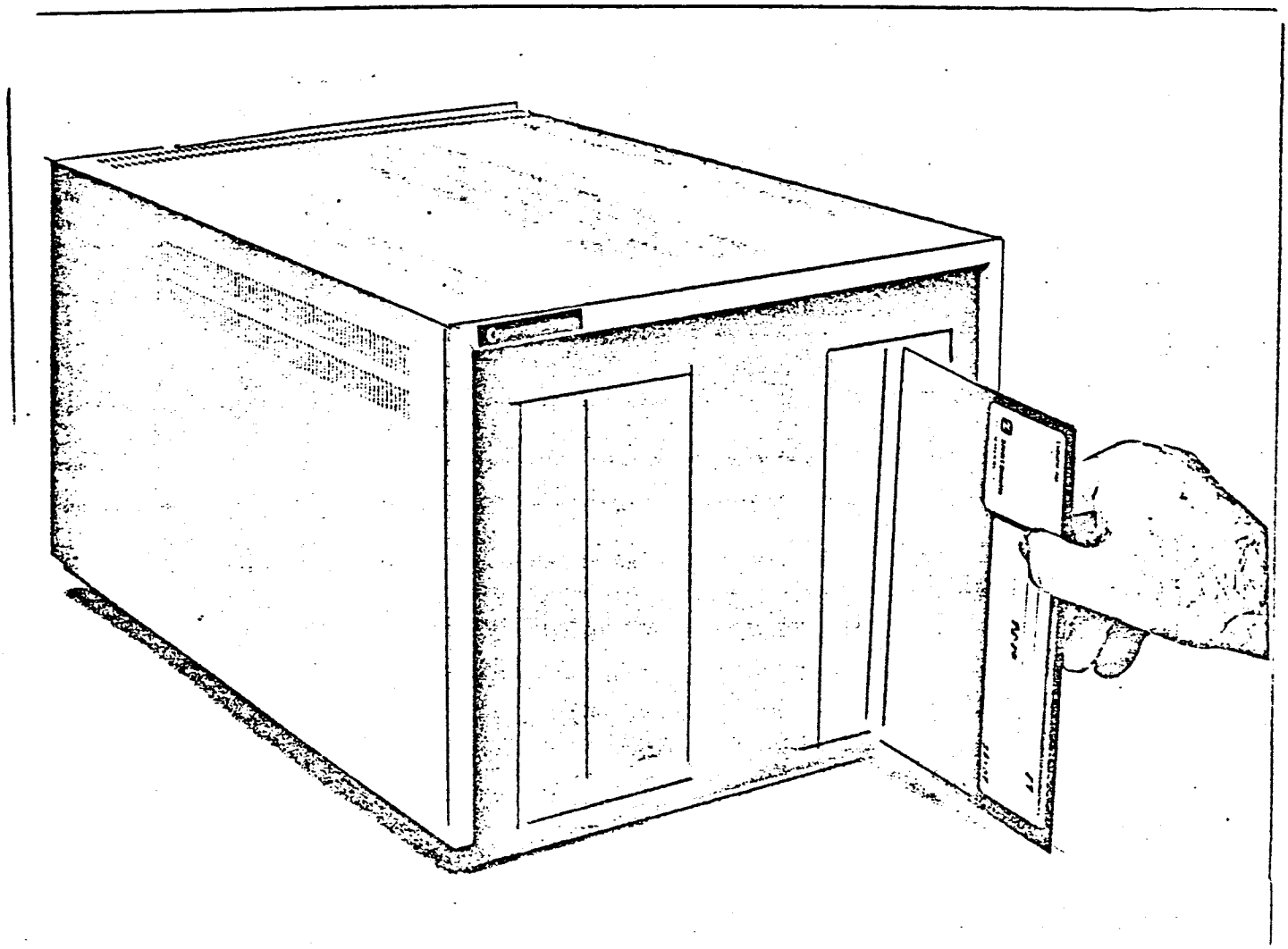


Figure 3-1. Disk Insertion

Disk Removal

A disk is removed from the drive by pressing on the latch release mechanism bar located at the front of each disk drive. The disk will be partially ejected. Remove the disk and store it in its protective jacket.

NOTE

Opening a door on the drive while a file is open, will result in an open file that can neither be read nor closed. Sequential files that have been left open will be flagged by an asterisk in the directory.

INITIALIZING THE DISK

Initialization of the disk is performed automatically by DOS when a 'new' disk is inserted into the drive and the door is closed. If initialization fails, the error message Drive Not Ready is displayed on the computer.

FORMATTING A DISK

To format a disk in the 'native format', the "formatter" program on the utility disk, must be loaded into the computer. The same "formatter" program is used to format both single sided and double sided disks. With this program a double sided disk can be formatted to be used in a single sided disk drive by typing "single" when the program asks for 'single' or 'double'. Do not type 'double' when formatting a single sided disk.

NOTE

Formatted, double-sided disks cannot be used in an 8061 drive. The other formats are, however, interchangeable among the 8061 and 8062 drives.

Table 3-1 summarizes the possible formats for the 8061/8062 drives.

Table 3-1. 8061/8062 Formats

Unit	Media	Format Possibilities
8061	Single Sided	Format as single sided sided
8061	Double Sided	Cannot be formatted
8062	Single Sided	Format as single sided
8062	Double Sided	Format as single or double sided

Perform the following steps to format a new disk:

1. Insert the utility disk into either drive and close the drive door.

2. Type: LOAD"FORMATTER",8 or DLOAD"FORMATTER",<DN> and press RETURN.

3. When the program is loaded and READY is displayed, remove the utility disk.
4. Insert a new, unformatted disk into either drive.

5. Type: RUN and press RETURN.

6. Type: 'CONTINUE' over the word STOP when asked by the program, and press RETURN.

7. Enter data asked for, and press RETURN. The program continues.

FORMAT PROGRAM SUMMARY

The format program accomplishes the following tasks:

1. Writes sectors and sector headers on the disk.

2. Writes a volume label on sector 1 of track 0 and verifies that it is readable. If it is not readable, the media is declared bad and the program halts.

3. If the volume label is read correctly, then each sector on the disk is verified . Those sectors that are unreadable are listed in the bad sector list (BSL). The BSL is stored on disk.

4. BAMs are assembled and written to disk. The BAMs are verified for a match to the actual sectors available. Sectors listed in the BSL are not available.

5. The directory is assembled and written to disk.

PREFORMATTED DISKS

Preformatted disks must meet the following data-exchange requirements:

Single Sided Media	Single Density Recording (FM)
77 Tracks/Diskette	128 Bytes/Sector
26 Sectors/Track	

IBMCOPY PROGRAM

The "ibmcopy" program copies a data-exchange formatted disk to a native-format disk or copies a native-format disk to a data-exchange disk. Load the "ibmcopy" program from the utility disk before using the data-exchange disk.

To load the "ibmcopy" program, perform the following steps:

1. Insert the utility disk into either drive.
2. Type, LOAD"IBMCOPY",8 or DLOAD"IBMCOPY", <DN and press RETURN.
3. When the program is loaded and READY is displayed, remove the utility disk.
4. Insert a data-exchange preformatted disk into drive 1 and a native-format disk into drive 0.

The "ibmcopy" program does not perform code conversion. During read, the "ibmcopy" program either creates or replaces some of the relative file on the native-formatted disk. Commodore recommends that you scratch a relative file before copying IBM formatted data to it. This way the file will only contain records copied and not records from a previous operation. Each record of a relative file is composed of one sector from an IBM* formatted disk. A record is 131 bytes long. The bytes are defined as follows:

1. The first byte indicates the sector read correctly. A correct read is indicated by Hex'00'; an unreadable sector is indicated by a non-zero character.
2. The second byte is the IBM* 'deleted data' mark. It indicates that a sector should be either ignored or was not deleted. If it is not deleted it will be a Hex'6F'. If it is to be ignored, it will be a Hex'6A'.
3. The next 128 bytes are exactly as read from the IBM* sector.
4. The last byte is always a Hex'FF'.

*IBM is a registered trade mark of International Business Machines

VALIDATE PROGRAM

The "validate" program lists the names of files that have unreadable sectors. As the program executes, DOS reads each sector. If the sector is ok, it is marked as allocated in the BAM.

If the sector is not ok, the file is flagged as containing bad sectors. This results in the disk being left in a state of 'validate lock'. You cannot write to the disk until "validate" has been run successfully. The program is destructive in that it restructures the BAMs as it executes.

To load the "validate" program from the utility disk, perform the following steps:

1. Insert the utility disk into either drive.
2. Load the "validate" program by typing:
   LOAD"VALIDATE",8 or DLOAD"VALIDATE",<DN>.
   Press RETURN.
3. When the program is loaded, remove the utility disk.
4. Insert the disk to be validated into the drive.
5. Type: RUN and press RETURN.
6. Enter the drive number when asked by the computer.

If the program has been completed successfully, the CRT displays:

'00,OK,00,00'

If an unrecoverable read error occurs, the CRT displays:

'XX,READ ERROR,00,00'

Where: XX = error code

If an unrecoverable error occurs, rerun the "validate" program on another drive as it could be a mechanical failure of the drive being used. If the drive that wrote the data on the disk is available, rerun "validate" on it. Finally, scratch those files in error and rerun "validate".

Chapter 4

DISK SPACE ALLOCATION AND FILE MAINTENANCE COMMANDS

INTRODUCTION

The information presented in this chapter describes space allocation on the 8061/8062 disk and the BASIC commands to manipulate the directory and files.

DISK SPACE ALLOCATION

Space on the disk is allocated by the disk operating system (DOS) which resides in ROM of the 8061/8062 drives. Disk space is divided into tracks and sectors (blocks of data). The single sided disk has 3106 sectors (blocks) available; the double sided, 6215. There are 256 bytes per sector (block).

Seven items are used by DOS to manage files on the disk: (1) volume label, (2) bad sector list, (3) block availablity map, (4) directory, (5) super side sector, (6) side sector, and (7) data sectors.

Volume Label

The volume label points to the BSLs, BAMs, and DIRs. The volume label is the only data on the disk fixed in position. It is located in sector 1 of track 00. All other items such as BAMs, BSL, directories, super side sectors, side sectors, and data sectors, can be located anywhere on the disk.

Bad Sector List

As the disk is formatted, DOS verfies each sector. Unreadable sectors are listed in the bad sector list (BSL) and marked as unavailable in the block availabilitly map.

Block Availability Map

The block availability map (BAM) keeps track of which blocks have been allocated to files. The single sided disk has three BAMs; the double sided disk has six.

Directory

The directories for the 8061 and 8062 drives differ from the 2040 or 8050 drives in at least three ways as follows:

1. The directory is not in a fixed location, it may occupy any group of sectors. The volume label contains the address of the first sector of the directory.

2. The directory is treated as a sequential file in that the maximum size is limited only by the number of sectors available.

3. The directory increases as files are created. Each directory sector contains descriptors pointing to eight files. The physical size of the directory stays the same as files are scratched. The DOS will, however, shrink and rewrite the directory to a new series of sectors when the validate command or validate program is run.

Super Side Sector

The super side sector contains the address of the first sector within each 'group' of the file. See Figure 4-1. A group consists of six side sectors as described in the next paragraph. The maximum number of groups is 91 in the largest relative file. There is one super side sector for each relative file.

Side Sector

The side sector is used by DOS to control relative files. The side sector contains the addresses of up to 120 data sectors. Side sectors also contain a table of pointers to the other five side sectors within that group.

An algorithm computes where the pointer is set when a record number is given by a record command. The information in the referenced side sector locates the data block pointer and reads the data block containing the record desired.

Data Sector

The data sectors, containing file data, are linked as follows:

1. The first byte is the sector address (actually the head and sector combined). It will be a Hex 'FF' if no more data sectors exist. FF is referred to as the end of list (EOL).

2. The second byte is the track address of the next sector or it can be the pointer to the last byte in the sector if EOL. This definition is the same as a data sector in a sequential file.

Figure 4-1.

DISK MAINTENANCE COMMANDS

The disk maintenance commands described in following paragraphs will enable you to manipulate files and perform disk maintenance functions.

Commands and statements in this manual are in lower case. If upper case characters are desired:

```
POKE 59468,12    Press RETURN.
```

To return to lower case:

```
POKE 59468,14    Press RETURN.
```

NOTE

The statements in this manual apply to floppy drives 8061 and 8062. Some of the commands differ in format or produce different results when used with other computers and peripherals. For exact usage of the commands and statements refer to the appropriate manual.

Format Notation

Wherever the format for a statement or command is given, the following rules apply:

1. Items in capital letters must be input as shown.
2. Items in lower case letters enclosed in angle brackets (<>) are to be supplied by the user.
3. Items in square brackets ([]) are optional.
4. All punctuation except angle brackets and square brackets (i.e., commas, parentheses, semicolons, hyphens, and equal signs) must be included where shown
5. Items followed by an ellipsis (...) may be repeated any number of times (up to the length of the line).
6. Items separated by a vertical bar ( ) are mutually exclusive; choose one.

Command Channels

The BASIC and DOS commands described in this manual and those preceded by an "open" command contain secondary addresses numbered 0 through 15. Secondary addresses 0 and 1 are reserved for load and save channels respectively. These files are flagged as program files.

Secondary address 15 is for either the command channel (when printed to) or error channel (when input from). Secondary addresses 2 through 14 are user file channels (sequential/relative files).

Commands are made into statements by commas, colons, and quotation marks. Command bytes are separated by commas as in the following examples:

1. OPEN 1,8,15

Where: OPEN 1 is the logical file number 1

8 is the primary address of the 8061/8062 drive.

15 is the secondary address of the command channel

2. PRINT#1,"CMD"

Where: PRINT#1 is the BASIC command

"CMD" is the request to DOS to execute that enclosed between the quotation marks.

BASIC FILE MANIPULATION COMMANDS

The following paragraphs describe BASIC commands to maintain files on the disk by displaying the directory, printing the directory, copying files, concatenating files, renaming files, scratching files, or collecting files.

Display Directory

To display a directory in BASIC 2.0 through 4.0 use the following commands:

To display either directory	1. Type: LOAD"$<drive #>",8	Press RETURN.
	2. Type: LIST	Press RETURN.
To display both directories	1. Type: LOAD"$",8	Press RETURN.

Print Directory

To print the directory, perform the following steps:

1. Insert the disk into drive 0.
2. Open device #4 by typing:

   OPEN4,4:CMD4 Press RETURN

3. Print the directory by typing:

   DIRECTORY D0 Press RETURN.

4. Close file and return control to the computer by typing:

   PRINT#4:CLOSE4 Press RETURN.

COPY Command

The COPY command, copies one or multiple files. Files may be copied from one disk to another or to the same disk. The copy command does not allow pattern matching as described under Scratching Files in this chapter.

The DUPLICATE or BACKUP command is not supported on the 8061/8062; use the all file COPY command.

Single File Copy

To copy a single file in BASIC 2.0 through BASIC 4.0, use one of the following copy commands:

(First) OPEN <LFN>,8,15

1. PRINT#(LFN),"COPYDDR:<DFN=SDR:SFN>"

2. PRINT#(LFN),"CDDR:<DFN=SDR:SFN>"

For example, to copy a file from 0 to 1:

```
PRINT#1,"C1:ACCT1=0:ACCT"
```

For BASIC 4.0 only use the following:

3. COPY DSDR,"<SFN>" to DDR,"<DFN>"

Where: SDR = source drive number
DDR = destination drive number
SFN = source file name
DFN = destination file name

All File Copy

To copy all files in BASIC 2.0 through BASIC 4.0, use the following copy commands:

1. PRINT#LFN,"COPYDDR=SDR"

2. PRINT#LFN,"CDDR=SDR"

For BASIC 4.0 only use the following copy command:

3. COPY DSDR TO DDR

Where DDR = the destination drive, it may be the same drive or another drive.

DFN = the destination file name, the file name may be the same if it is written to another disk or it must be different if it is written to the same disk.

All files are copied from drive 0 to drive 1.

```
PRINT#1,"C1=0"
```

Concatenation

To concatenate files in BASIC 2.0 through BASIC 4.0, use either of the following commands:

1. PRINT#LFN,"COPYDDR:<DFN=SDR:SFN1,SDR:SFN...>"

2. PRINT#LFN,"CDDR:<DFN=SFN1,SDR:SFN...>"

   Where: The DDR:DFN and SDR:SFN1 are the same.

For BASIC 4.0, the following is supported.

CONCAT DSDR,"<SFN>"to DDR,"<DFN>"

Where: The file named DFN on drive DDR will contain the contents of both DFN and SFN after concatenation.

CONCAT D0,"yourfile" TO D1,"myfile" will result in myfile on drive 1 containing data from the old myfile and from yourfile. Yourfile will remain the same.

For example:

Files from both drives are concatenated into a file on drive 1. The file name should be short as the maximum length of a disk string is 50 characters.

PRINT#1,"C1:JDATA=1:ACCT1,0:ADATA,0:BDATA"

RENAME

To rename an existing file, a file with the same name should not exist or a file exists error message will be generated. The letter R is an abbreviation to rename a file. The RENAME command does not allow pattern matching as described under scratching files. To rename a file in BASIC 2.0 through BASIC 4.0 use the following command:

PRINT#1,"RENAME DR:<NFN=OFN>"

For BASIC 4.0 the following is supported.

NOTE

Close any open files before using the RENAME command. The command cannot be executed on an open file.

RENAME DDR,"<OFN>" TO "<NFN>"

Where: DR = drive in which the file is located
NFN = new filename
OFN = old filename
LFN = logical file number between 1 and 255

SCRATCH

The SCRATCH command erases one or more files from a disk. The files are not physically erased, but their sectors are marked as available in the BAM. Refer to BASIC 4.0 Reference Manual for complete information on the scratch command.

Scratching One File - To scratch a specific file on a disk in BASIC 2.0 through BASIC 4.0 use the following command:

PRINT#1FN,"SDR:FN"

Where: DR = the drive to be searched
:(alone) = (last drive acessed)
:(with DR) = specified drive
:(omitted) = (both drives)
FN = filename to be erased

For BASIC 4.0 only, use the following command:

SCRATCH DDR,"<FN>"

Where: DR = drive number
FN = file name

Scratching Multiple Files - To scratch more than one file at a time, enter the name of each file as in the following example:

1. Several files (unrelated)

PRINT#1,"S0:ACCT,0:CUSTOMER,0:INV,0:BILLING"

2. All files:

PRINT#1,"S0:*"

Pattern Matching - The following discussion on pattern matching applies, not only to scratching files, but also to DOPEN, OPEN, LOAD and DLOAD.

Scratching Files By Pattern Matching - Pattern matching is available on the 8061/8062 drives. The asterisk (*) and the question mark (?) are used to define filename patterns for multiple files. The asterisk appears at the end of a string and indicates that the rest of the string is not significant. For example:

FIL* refers to all other files starting with "FIL"

such as:

FIL
FILE1
FILEDATA
FILLER

For example:

PRINT#1,"S0:FIL*" scratches all files that begin with "FIL" on drive 0.

A question mark may be used within a filename to indicate that the character in that position should be ignored. For example:

?????.SCR refers to files named

TSTER.SCR
DIAGN.SCR
PROGR.SCR

The characters .SCR must appear after the question marks or the filename will not be recognized. Both the characters and their position are significant.

The asterisk and the question mark may be combined in a string, but the asterisk must appear at the end of that string or the filename will not be recognized. For example:

P???FIL* will access files named:

PET FILE
PRG FILE-32
POKEFILES$$

SCRATCH should be used very carefully with pattern matching as any file that matches the string will be scratched.

COLLECT

This command scratches files that have not been closed. The COLLECT command in BASIC 4.0 DOS systems is the same as the VALIDATE command used by the Validate Utility program, in that it reconstructs the existing BAM according to the valid data on the disk. The format for the COLLECT command follows:

```
COLLECT DX   (COLLECT alone defaults to drive 0)
Where:  X = the drive number.
```

When pattern matching, the OPEN and DOPEN commands will open the first file encountered that fits the description. The commands OPEN and DOPEN are not used to open a new file when doing pattern matching.

The LOAD and DLOAD commands will load the first file which fits the pattern matching description.

The commands RENAME, SAVE, DSAVE, and COPY do not allow pattern matching. An error condition will result if attempted.

Chapter 5

DATA HANDLING

INTRODUCTION

Data is transferred to and from the 8061/8062 by issuing BASIC commands from the computer. The commands discussed here are for systems having BASIC 2.0 through BASIC 4.0. Systems with BASIC 4.0 may use all the commands discussed, however, systems using a BASIC prior to 4.0 may not use commands listed for BASIC 4.0 only.

BASIC DATA HANDLING COMMANDS

The following commands are discussed in this chapter:

1. LOAD"DR:FN",PA
2. SAVE"DR:FN",PA
3. VERIFY"DR:FN",PA
4. OPEN1FN,PA,SA,"DR:FN,S/P/L,R/W/A LEN"
5. CLOSELFN
6. PRINT#LF, etc.
7. INPUT#
8. GET#LF, etc.

The following BASIC commands are for systems with BASIC 4.0 only.

1. DLOAD"FN"
2. DSAVE"FN"
3. DOPEN#LFN
4. DCLOSE#LFN
5. RECORD#LFN,R,B
6. APPEND

Where:
- LFN = logical file number between 1 and 255.
- FN = file name, supplied by user with maximum of 16 characters.
- DR = drive number 1 or 0
- PA = primary address (device number)
- SA = secondary address (refer to Disk maintenance commands in Chapter 4).
- P = program file
- S = sequential file
- L = relative file
- LEN = relative file length
- R/W = read or write
- B = byte position

LOAD and DLOAD

A program on disk can be loaded into computer memory by a LOAD or DLOAD command. You must specify the program name and primary address. The primary address will default to a 1 if not specified. A successful LOAD or DLOAD will close all open files. Also see "Quickload" at the end of this chapter.

Format for BASIC 2.0 through BASIC 4.0 is as follows:

```
LOAD"DR:FN",LFN
```

Format for BASIC 4.0 only is as follows:

```
DLOAD"FN",<DDR>
```

SAVE and DSAVE

A program in the computer can be transfered to disk by the SAVE or DSAVE command. Data saved by the SAVE or DSAVE command is automatically labelled as a program file (PRG) by DOS. You must specify the drive number, program name, and primary address. The DOS will select a drive if not specified. The primary address will default to a 1 if not specified. The SAVE and DSAVE commands do not allow pattern matching.

Format for SAVE in BASIC 2.0 through BASIC 4.0 is as follows:

```
SAVE"<DR:FN>",DN
```

The following example is an exercise in creating a one line program and saving it on a disk in drive 0.

```
10PRINT"TEST"
SAVE"0:TEST",8
```

Format for DSAVE in BASIC 4.0 only is as follows:

```
DSAVE"<FN>",<DDR>
```

Where: DR = disk drive number (0 or 1) D0 is default.
FN = file name (16 characters or less; blanks are characters)
DN = device number (see primary address in Chapter 4)

VERIFY

The VERIFY command ensures that the data stored under a file name is the same as data stored in the computer's memory.

Formats for the VERIFY command follow:

1. VERIFY"<DR:FN>",PA

2. VERIFY"*",PA

Where: DR = drive number (0 or 1)
FN = file name
PA = primary address (8)

An exercise to issue the VERIFY command follows:

1. Write a short program and store it on a disk in drive 0.

2. Type: VERIFY"0:TEST",8

When verification is complete, the screen will display the following:

```
VERIFY"1:TEST",8
SEARCHING FOR 1:TEST
VERIFYING
OK

READY
```

If an error occurs, resave the program and verify it again.

OPEN and DOPEN

The OPEN command sets up a correspondence between a logical file number and a file on the disk. It also reserves buffer space within the drive for operations on the file being opened. In BASIC 4.0, the DOPEN command may be used to create relative or sequential files.

The format for the OPEN command follows:

OPENLFN,PA,SA,"<DR:FN,FT,mode>"

Where:
- LFN = the logical file number
- PA = the primary address (refer to chapter 4)
- SA = the secondary address, Data files as stated in Chapter 4. It may be any number from 2 through 14 and be input or output as specified in mode.
- DR = the drive number 0 or 1
- FN = the name of the file
- FT = the file type, it may be sequential (SEQ), relative (L), or program (PRG).
- mode = describes how the channel is to be used, it may be either read (R) or write (W) or lengt if relative, or (A) for APPEND.

```
OPEN 2,8,2,"0:FILE1,SEQ,WRITE"

OPEN 3,8,9,"1:TESTDATA,PRG,WRITE"
```

The contents of a file on the 8061/8062 may be replaced by precedin the drive number with an ampersand symbol (@) in the OPEN command.

```
OPEN 3,8,5,"@0:JDATA,SEQ,WRITE"
```

If the specified file does not exist, normal open procedures are used Some open parameters may be assigned a variable name.

```
FL$ ="0:FILEA,SEQ,READ"
OPEN 1,8,14,FL$

FL$ = "0:FILEA"
OPEN 1,8,14,FL$+",SEQ,WRITE"
```

DOPEN Format - The format for DOPEN is as follows:

DOPEN#1<FN,"FN",DDR,LRL(,ONUDN)(,W)>

Where:

LFN,FN,and DR	are the same as defined for OPEN.
LRL	defines the record length as equal to RL.
ONUDN	specifies the device number as equal to DN (with default device being 8)
W	may be specified for write mode. If W is not specified for sequential files, the files will be open to read.

```
DOPEN#2,"<example file>",D1,120
```

When pattern matching, the OPEN and DOPEN commands will open the firs file and read that which fits the pattern. The commands OPEN and DOPE can not be used to open a new file when doing pattern matching. Patter matching is explained under Scratching Files in Chapter 4.

CLOSE and DCLOSE

The CLOSE command closes files opened by an OPEN command. The command DCLOSE closes any file in a DOS 4.0 system opened by the DOPEN command.

NOTE

Do not open the drive door during disk operations; this will leave a file open, unreadable, and in a state that cannot be closed.

When this occurs, the sequential file is unusable, but the relative file is useable although it may be missing the last few updates. The files will indicate they were opened and not closed as follows:

1. Sequential (SEQ), '*' in the directory
2. Relative (REL), the '*' will be absent but the block count will not be updated. The count will be zero if it was just being created or it will remain the same as it was when the last successful close was made.

The format for CLOSE is as follows:

CLOSELFN

Where: LFN = the logical file number

The formats for DCLOSE are as follows:

1. DCLOSE#LFN
2. DCLOSE ONUDN

Where: DN = device number of the disk drive (defaults to 8)

The command DCLOSE closes all active disk files on the drive specified. Examples follow:

1. DCLOSE: closes all files currently open
2. DCLOSE#5: closes only logical file 5

Closing The Command Channel

Closing the command channel closes all disk channels. But the computer does not recognize the closing of files. An example of this error and its correction is displayed on the screen as follows:

```
OPEN 1,8,15                          Command channel is opened

OPEN 3,8,2,"0:FILE1,SEQ,WRITE"       Data channels are opened
OPEN 4,8,5,"0:FILE2,SEQ,WRITE"       for writing.

PRINT#3,"important data"
PRINT#4,"more data"

OPEN 3,4                             Channel is opened to
                                     printer by mistake.

?FILE OPEN ERROR                     Error message is dis-
READY.                               played on the screen.
```

All logical files in the computer are closed because of this error, but the channels in the drive are still open. The drive channels are closed by the following command:

```
OPEN 1,8,15
CLOSE1
```

Closing The Data Channel

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

PRINT#

The PRINT# command transmits a command string to a drive or transmits data to an opened file. To avoid sending unwanted line feed characters to the disk, a semicolon must terminate each PRINT# statement. The DOS 4.0 system does not require the semicolon terminator as files opened by an LFN lower than 128 will automatically suppress line feeds.

For BASIC 4.0 the following format is used:

```
PRINT#LFN,A$
```

Where: LFN<128

This produces the desired value of A$ followed by a carriage return. An example of PRINT# command for other than DOS 4.0 follows:

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

Where: CHR$(13) is the carriage return and it will produce the desired value for JONESABC.

To write several variables to a disk at the same time the following formats are used:

1. PRINT#LFN,A$,B$,C$ — This results in a single variable (A$+B$+C$) being retrieved by the input command.

2. PRINT#1FN,ACHR(13)BCHR(13)C$ — This results in the variables A$,B$,C$ being separated by carriage returns so they may be input as separate variables.

INPUT#

The INPUT# command transfers data from the disk into the computer's memory. The command is only valid in a program that references a logical file. It may also be used to transfer several strings of data at one time.

The formats for INPUT# are as follows:

1. INPUT#LFN,A$

2. INPUT#LFN,A

Where: LFN = a file opened using a secondary address of 0 to 15
A$ = a string variable which contains the data to be transferred.
A = a numeric variable which contains the data to be transferred.

3. INPUT#LFN,A$,B$,C$

NOTE

Format 3 is only used when the input bytes are separated by carriage returns. No single string can contain more than 80 characters if the INPUT# format is used.

Examples of INPUT# formats are as follows:

1. 20INPUT#2,A — Input the next numeric data item and assign the value to variable "A."

2. 10INPUT#8,A$ — Input the next data item as a string and assign it to variable "A$."

3. 60INPUT#7,B,C$ — Input the next two data items. Assign the first data item to numeric variable "B" and the second data item to string variable "C$."

GET

The GET command transfers data bytes from the 8061/8062 into computer memory. GET# is only valid in a program that references an open file. When strings of more than 80 characters have been written to the disk, the GET# command is useful in retrieving these strings. Examples of GET# follows:

1. The following format for an individual byte:

```
GET#LFN,A$
```

2. A program to get several bytes of data and an input string with more than 80 charaters is shown. It results in a string length of 245 characters.

```
10 AA$=""
20 FORI = 1 to 254
30 GET#LFN,A$
40 AA$=AA$+A$
50 NEXT
```

Where: LFN = a file opened by a secondary address 15.
A$ = a string variable containing the data transfered.
AA$ = a string variable containing the data transfered.

Note:

if LEN(A$)=0 THEN A$=CHR$(0)
if the character is a binary 0 (CHR$ 0) then the length of A$ is set to 0.

APPEND#

The APPEND# command writes additional data to the end of a sequentia file. The pointers must be positioned beyond the current end of file Additional data may then be written and the file closed.

For BASIC 4.0 the following format may be used:

APPEND#<LFN>,"<FN>"[,<DR>][on <DN>]

Where: LFN = logical file number <255
FN = filename
DR = drive number; defaults to 0
DN = unit; defaults to 8

RECORD

This command is available only in BASIC 4.0 systems. The RECORD command is used before a PRINT#, INPUT#, or GET# command to positio the file pointer to the desired record (and byte) of a relative file If the record pointer is set beyond the last record and a PRINT command is used, an appropriate number of records are generated t expand the file to the desired record length. The format for th RECORD# command follows:

RECORD#LFN,R,B

Where: LFN = a logical file opened by a DOPEN command
R = the desired record number. R may be either a variable name or a value. If R is a variable name it must be enclosed in a parentheses. 0<=R<=65535
B = byte position desired within the record. Byte positioning is optional. 1<B<=254

An example of the RECORD# command, used with INPUT#, follows:

1. 10 RECORD#1,120 RECORD command selects the record.
2. 20 INPUT#1,A$ input the next data item as a string and assign it to variable A$.

Point (record) - PRINT#LFN,"P"+CHR$(sa)+CHR$(lrn)+CHR$(hrn)+CHR$(pos)

--- Secondary Address of an open relative file

--- Must be a single character

--- Channel 15 (command channel)

Where: lrn = low record number = low byte (shown in decimal) the two byte binary record #.
hrn = high record # number = high byte of the two byte record number.
pos = position within the record (1 relative)

Records numbers are relative to one; for example:

1.

Record n	CHR$(lrn)	CHR$(hrn)
1	1	0
85	85	0
256	0	1
512	0	2
530	18	2
260	4	1

2. PRINT#LFN"P"CHR$(sa)+CHR$(rp-INT(rp/256)*256)+CHR$(INT(rp/25
+CHR$(pos)

Where: sa = secondary address
rp = record position
pos = position within the record

QUICKLOAD FEATURE

This command loads the first file from drive 0 into computer memory and executes it. The quickload command must be the first command given after a cold start to ensure that the first program on the disk is read. If quickload is not used immediately after coldstart, the last program file succesfully accessed will be loaded. Available in BASIC 4.0 only. To issue the quickload command perform the following:

1. Turn computer power off and back on again.
2. Insert the disk into drive 0.
3. Press SHIFTed RUN STOP.

The computer loads the first program from disk into computer memory. The DLOAD and RUN commands are automatically executed when the quickload command is used. The screen displays the following as quickload executes:

```
*** COMMODORE BASIC 4.0 ***

31743 BYTES FREE

READY.
DL "*

SEARCHING FOR 0:*
LOADING
READY.
RUN
```

COPYING TAPE TO DISK

A BASIC program stored on a cassette tape can be copied to disk by performing the following steps:

Function	Display
1. Load the file from the tape to computer memory.	LOAD"DEMO" PRESS PLAY ON TAPE #1 OK SEARCHING FOR DEMO FOUND DEMO LOADING READY.
2. Create a program file containing the program on the disk.	SAVE"1:DEMO",8 VERIFY"1:DEMO",8 SEARCHING FOR 8:DEMO VERIFYING OK READY
3. Clear the computer memory (BASIC new command)	NEW READY
4. Load the program into computer memory from the disk	LOAD"1:DEMO",8 SEARCHING FOR 1:DEMO LOADING READY.
5. Run the program to verify it has been loaded into the computer from the disk correctly.	RUN

Chapter 6

ADVANCED DISK PROGRAMMING

INTRODUCTION

This chapter provides detailed information about DOS the structure and disk utility commands. The utility commands provide the programmer with low-level functions that may be used for special applications such as special disk handling routines and access techniques.

The Disk Operating System (DOS) manages all transfers between the computer and the 8061/8062. The DOS program resides in ROM in the 8061/8062 drives.

DOS CHANNELS

The file system is organized in channels that are opened by a BASIC command as stated in Chapter 5. As the channels are opened, the DOS assigns a work space to that channel and allocates two or three disk I/O buffer areas. If work space or buffers are not available a no channel error message is displayed. The DOS also uses the channel structure to search the directory, delete files, and copy files.

The secondary addresses are given in the OPEN statement, as described in Chapter 4. The DOS uses these secondary addresses as a channel number. The user number assigned to a channel references a work area; it is not related to the DOS ordering of channels.

LOAD and SAVE statements transmit secondary addresses of 0 and 1 respectively as noted in Chapter 5. The DOS interprets these secondary addresses as a LOAD or SAVE function. The LOAD or SAVE function can be avoided by using any number between 2 and 14. Up to nine channels may be opened to receive data.

DISK UTILITY COMMAND SET

The disk utility command set is listed in Table 6-1.

Table 6-1. Disk Utility Command Set

These commands are intended for use during diagnostics only.

COMMAND	ABBREVIATION	FORMAT
MEMORY-WRITE	M-W	"M-W"adl/adh/nc/data
MEMORY-READ	M-R	"M-R"adl/adh/nc
MEMORY-EXECUTE	M-E	"M-E"adl/adh

Where: adl = the low byte of the address
adh = the high byte of the address
nc = the number of characters; 1 through 34*
data = the actual data in Hexidecimal.
(Transmitted by use of CHR$ function)

CHR$(1) sends the binary equivalent of Hexidecimal 01,(decimal 1).

The three commands listed in Table 6-1 are byte-oriented so that you may use machine language programs. BASIC statements may be used to access information through the memory commands when the CHR$ function is used. The system will only accept the abbreviations shown. It will not accept the colon symbol(:).

MEMORY-WRITE

This command gives direct access to DOS memory. The command also allows special routines to be down-loaded to the 8061/8062 and then executed using the MEMORY EXECUTE command. Each command allows up to 34 bytes to be deposited. The low byte must precede the high byte in the address. An example of the command format M-R:"Adl/adh/nc/data follows:

"M-W:CHRS$(00)CHR$(18)CHR$(4)CHR$(32)CHR$(17)CHR$(96)

Writes four bytes to $1200 or decimal 4608.

MEMORY-READ

This command accesses bytes pointed to by the address in the command string. The command allows variables from the DOS or the contents of the buffers to be read. Command M-R changes contents of the error channel as it is used to transmit data to the computer. The next GET# command from the error channel (secondary address 15) transmits the byte. Do not execute an INPUT#. until a DOS command other than a memory command is executed. An example of the MEMORY-READ command "M-R"adl/adh/nc follows:

```
"M-R"CHR$(128);CHR$(0);CHR$(4)
```

Accesses the four bytes located at ($0080 or decimal 128)

MEMORY-EXECUTE

This command executes subroutines in DOS. Return to the DOS is accomplished by terminating the subroutine with RTS($60). An example of the command format "M-E:"adl/adh follows:

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

This statement requests the execution of code beginning at $3180.

USER Command

The USER (U) command causes the DOS power up (vector) message to be stored in the error channel. The message identifies the product and version number.

Chapter 7

ADVANCED FILE HANDLING

INTRODUCTION

This chapter briefly describes the organization of relative files on the 8061/8062. Also, a discussion of techniques and sample programs to create, expand, and access relative files is included.

RELATIVE FILE GENERAL DESCRIPTION

The relative file operation reduces the time to access a file. Access is accomplished in two ways: (1) by reading each sector in order, or (2) by reading the sectors randomly. If the GET or INPUT commands are used without a RECORD command, the sectors will be accessed in order. Refer to Chapter 5 for individual descriptions of the GET, INPUT, and RECORD commands.

Relative File Organization

A relative file contains three types of sectors as follows:

1. Super Side Sector
2. Side Sector
3. Data Sector

A minimum relative file contains one super side sector, one side sector, and one data sector.

Super Side Sector

The super side sector contains the address of the first sector within each 'group' of the file. A group consists of six side sectors as described in the next paragraph. The maximum number of groups is 91 in the largest relative file.

Side Sector

The side sector contains the addresses of up to 120 data sectors. Side sectors also contain a table of pointers to the other five side sectors within that group.

An algorithm computes where the pointer is set when a record number is given by a record command. The information in the referenced side sector locates the data block pointer and reads the data block containing the record desired.

Data Sector

The data sectors are linked as follows:

1. The first byte is the sector address (actually the head and sector combined). It will be a Hex 'FF' if no more data sectors exist.

2. The second byte is the track address of the next sector or it can be the pointer to the last byte in the sector if EOL. This definition is the same as a data sector in a sequential file.

Expanding Relative Files

To expand a relative file, set the pointer to the last record you want to store. PRINT data to the file. The DOS will expand the file to accomodate the new data. For example, if a file contains 100 records and you want to add 100 records, set the pointer to 200. A 50-record-not-present error message will occur as a warning not to use an INPUT or GET command. PRINT to record 200 and the DOS will expand the file to include side sectors and data sectors necessary to contain the file. The size is only limited by the capacity of the disk.

I/O OPERATION

When a channel is opened to an existing file, the DOS will position to the first record. A DOPEN statement need not have the record length variable if the file is in existence. If the record length is supplied, DOS will check the record size in the original DOPEN statement when the file was created. If they do not match, a 50-record not present- error message will be displayed.

A relative file requires three memory buffers from the system; a sequential file requires two. There are twenty-four buffers in the system. Four being used in the directory searches and internal functions. Only six relative files can be open at one time. Nine sequential files can be open at one time.

Spanning Data Blocks

Spanning relative files reduces the number of read/write operations needed to retrieve data. If a record begins in one block and ends in another, DOS reads the correct number of segments.

The records of most relative files span across data blocks. Exceptions are record size 1, 2, 127 and 254. These divide evenly into 254, (data block size) making spanning unnecessary. This method of spanning has an advantage in that system memory overhead is only for generating side sector blocks in the relative file. When a record is written into by a PRINT# statement, it is only written when DOS has moved beyond that particular data block containing the record. This can occur through sucessive printing to sequential records or when positioning to another record outside of that particular block.

Terminating

As the DOS generates new data blocks for relative files, the requested record number is compared to the number of available data blocks on the disk. If the requested number is greater than the number of blocks left, a 52-file-too-large error message is displayed.

The DOS terminates printing to a record by detecting the EOI signal generated by each PRINT# statement. If the print statement exceeds the number of characters specified by the record size, a 51-record overflow error message is displayed. Data overflow is truncated to fit the number of characters specified by the record size and DOS positions to the next record in sequence. If the print statement specifies fewer characters than the record size, the remaining bytes are filled with nulls.

To store binary data which may contain nulls (0's), a record terminator such as a carriage return should be used. The record size must be increased by one character to accomodate the terminator. If a record terminator is not used, reading binary data with trailing zeros will result in the zeros being truncated. In this case, the EOI signal is generated by DOS when the last non-zero character is transmitted.

HANDLING A RELATIVE FILE

The following examples of programs and commands are used, in systems with BASIC 2.0 through BASIC 4.0 DOS, to handle relative files.

Creating A Relative File

The file should be initialized the first time it is opened to obtain faster subsequent access and to assure that the DOS reserves sufficient space on the disk for future data. A sample initialize program follows:

```
110 DOPEN#1,"FILE1",D0,150
120 GOSUB 190
130 RECORD#1,100
140 GOSUB 190
150 PRINT#1,CHR$(255):
160 GOSUB 190
170 DCLOSE#1
180 END
190 IF DS<20 THEN RETURN
200 PRINTDS$
210 IF DS=50 THEN RETURN
220 STOP
```

Where: Line 110 creates a file with the name FILE1 and a record length of 50.

Lines 120, 140 and 160 cause the error handling subroutine to be executed. You should check the error channel after each disk operation).

Line 130 positions the file pointer to record number 100 (it does not exist yet), an error message 50 record not present is displayed as a warning and not an error condition. This is a normal message when a new record is accessed for the first time; it indicates that a get or input operation should not be attempted.

Line 150 writes record number 100 (pointer in line 130 present here). During the write operation, the DOS detects that records 1 through 9 do not exist yet, and initializes them by placing CHR$(255) in the first character.

Line 170 closes the file and allocates space in the BAM and directory.

Lines 190 through 220 are error subroutines.

1. If DS is less than 20, no error exists; so line 190 returns control to the main program.

2. Line 200 prints the error message and line 210 returns to the main program if the message is 50 record not present.

3. Line 220 halts the program if an unexpected error occurs (such as a read error), and allows a correction to be made.

Typical Program to Expand A Relative File

When the file has been created (as in the previous example) and it is to be expanded, then line 130 in the program is changed to read as follows:

```
130 RECORD#1,200
```

This revised line-130 input increases the records from 100 to 200 as the pointer is now at a record not yet existing. When the file is expanded in this manner the required side sectors are also created. This allows the first 100 records to remain the same, but records 101 through 200 will contain zeros. When DOPEN is used on an existing file, and if the file length was not changed, it is not necessary to enter the file length again.

Accessing A Record

Accessing a file is simplified by the use of relative files. The RECORD command is used to position the pointer to the desired record before performing the read or write operation. To write data to a specific record in the file, a constant may be used in line 130 to position the pointer as follows:

```
RECORD#1,25
```

A representation of record 25 appears as follows:

```
          1         2         3         4         5
12345678901234567890123456789012345678901234567890
--------------------------------------------------
     record 25 data goes here
--------------------------------------------------
```

Where: * represents a carriage return (CHR$(13)).

Accessing A Byte

Individual bytes within a record are accessed by the following program:

```
110 DOPEN#1,"FILE1",D0
120 GOSUB 900
130 RECORD#1,25,1
140 GOSUB 900
150 PRINT#1,"FIELD 1"
160 GOSUB 900
170 RECORD#1,25,30
180 GOSUB 900
190 PRINT#1,"FIELD2"
200 GOSUB 900
210 RECORD#1,25,30
220 GOSUB 900
230 PRINT#1,"FIELD 3"
240 GOSUB 900
250 DCLOSE#1
260 END
900 IF DS<20 THEN RETURN
910 PRINT DS$
920 STOP
```

Where: Lines 130, 170, and 210 cause the file pointer to be moved to different places within record number 25.

A representation of record 25 after the program is executed follows

```
          1         2         3         4         5
12345678901234567890123456789012345678901234567890
--------------------------------------------------
Field 1* Field 2*             Field 3*
--------------------------------------------------
```

Where: * represents a carriage return(CHR$(13)).

Fields must be written in sequence as writing to a byte at the beginning destroys the the rest of the record in memory. For example: writing byte 3 before byte 2 and then writing byte 2 will destroy byte 3 .

Typical Program to Retrieve Data

The carriage return is recognized as a terminator by the BASIC input statement so data may be retrieved by the following program:

```
110 DOPEN#1,"FILE1",D0
120 GOSUB 290
130 RECORD#1,25
140 GOSUB 290
150 RECORD#1,25,1:GOSUB 290
160 INPUT#1,A$:GOSUB 290
170 RECORD#1,25,10:GOSUB 290
180 INPUT#1,B$:GOSUB 290
190 RECORD#1,25,30:GOSUB 290
200 INPUT"1,C$:GOSUB 290
210 DCLOSE#1
220 END
290 IF DS<20THEN RETURN
300 PRINT DS$
320 STOP
```

Where: Lines 160, 180, and 200 cause the stored values on the disk to be read and stored in A$, B$, and C$,respectively.

Typical Program to Access Record

A record that was determined during a program operation may be accessed. The routine to access the record asks the operator for a record number and data and it will write the data to the file as follows:

```
100 PRINT"TYPE RECORD NUMBER AND DATA
105 INPUT R,D$
110 DOPEN#1,"FILE1",D0
120 GOSUB 190
130 RECORD#1,(R)
140 GOSUB 190
150 PRINT$1,D$
160 GOSUB 190
170 DCLOSE#1
180 END
190 IF DS<20 THEN RETURN
200 PRINTDS$
220 STOP
```

Where: Line 130 positions the file pointer to record number (r) that is specified by the user. The variable must be enclosed in parentheses.

Line 150 causes the data stored in D$ to be stored on the disk.

SEQUENTIAL ACCESS

The RECORD command may be omitted if a sequential search is to be made. Sequential operation may serve better when a large data base is being written to the disk. When the program has been dimensioned as an array which contains 100 elements they are to be written to the disk in record numbers 1 through 100 of FILE1. This is accomplished with the following program segment:

```
110 DOPEN#1,"FILE1",D0
120 GOSUB 190
130 FOR I=1 TO 100
150 PRINT#1,D$(I)
160 GOSUB 190
165 NEXT I
170 DCLOSE#1
180 END
190 IF DS<20 THEN RETURN
200 PRINT DS$
220 STOP
```

The pointer is set to record 1 when the file is opened. File 1 will be the first file written and if no RECORD command is executed, the DOS moves the pointer to the next record after each print. In this program the DS$ is written to records 1 through 100 of the file.

Chapter 8

THE DOS SUPPORT PROGRAM

INTRODUCTION

The utility disk has a program called DOS Support (also referred to as the Universal Wedge). This program is loaded into memory to make loading and running programs stored on a disk easier. All systems with BASIC 2.0 through BASIC 4.0 may use the following format to load the DOS Support program.

1. Make a cold start by resetting the power on the computer.
2. Insert the utility disk into drive 0.
3. Type: LOAD"*",8. Press RETURN.

   The screen displays the following:

```
READY.
LOAD"*",8
SEARCHING FOR *
LOADING
READY.
(cursor)
```

4. Type:"RUN". Press RETURN.

The program will be loaded into the top of computer memory. The DOS Support program uses special symbols to simplify entry of disk commands. A user quick reference guide lists DOS and basic commands in Chapter 9.

DOS SUPPORT > AND @ SYMBOLS

The DOS Support program makes using disk commands with PRINT#LFN or enclosing the filename in quotation marks no longer required. Disk commands are preceded by the greater than (>) symbol or an ampersand (@) symbol. The following examples use the greater than (>) symbol.

1. >I0 is the same as PRINT#1,"I0"
2. >S0:FILE1 is the same as PRINT#15,"S0;FILE1"

The OPEN statement is not required before a statement.

The > DOS symbol is used to read a directory as follows:

1. >$0 means display the entire directory of drive 0.
2. >1:Q* means display all the files on drive 1 that begin with Q.

To avoid scrolling the directory, press the space bar to stop the listing. To continue, press any key. To stop the listing and return to BASIC, press the RUN STOP key.

Another use of > is to request error messages as follows.

> is equivlent to the following program:

```
10 OPEN2,8,15
20 INPUT#2,a$,b$,c$,d$
30 PRINTA$+","+B$+","+C$+","+D$
@-----
```

LOADING A PROGRAM WITH THE /

The backslash (/) loads a program from disk. Disks in both drives are searched if the drive number is not specified. An example of using the backslash follows:

/ACCT loads the program "acct" into computer memory.

LOADING AND RUNNING A PROGRAM WITH AN UP ARROW

The up arrow (↑) loads a program from a disk and executes it. Both disks are searched if necessary. An example of using the up arrow is as follows:

↑JDATA loads and runs the program JDATA.

DOS SUPPORT LIMITATIONS

Limitations of the DOS Support program are as follows:

1. The program must be reread from disk when computer power is reset.

2. DOS SUPPORT symbols may only be used when communicating with the disk in the direct mode. Symbols may not be used in a program.

3. The disk directory can be printed by the following commands.

```
LOAD"$0",8
OPEN4,4:CMD4:LIST
PRINT#4:CLOSE4
```

The primary address (PA) is the physical device number (8,9,10,11) an 8061/8062 depending on the way it is strapped. If a primary addre is not entered, the address defaults to 8.

Chapter 9

ERROR MESSAGES, AND QUICK REFERENCE GUIDE

INTRODUCTION

This chapter contains a summary of the DOS error messages displayed on Commodore computers.

ERROR MESSAGES

Error messages can be displayed as follows:

For CBM Series 2001
CBM Series 3000
with
BASIC 3.0

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

For CBM Series 4000
CBM Series 8000
with
BASIC 4.0

```
PRINT DS$  (cursor)
```

Where: A = error message number
B$= error message
C = track 0-76
D = head and sector
(002)

-- sector 00 through 25
-- head 0 or 1

SUMMARY OF ERROR MESSAGES

0	OK, no error exists.
1	Files scratched response, not an error condition.
2-19	Unused error messages: should be ignored.
20	Block header not found on disk.
23	Checksum error in data
24	Byte decoding error.
25	Write-verify error.
26	Attempt to write with write protect on.
27	Checksum error in header
29	Disk ID mismatch.
30	General syntax error.
31	Invalid command.
32	Long line.
33	Invalid file name
34	No file given.
39	Command file not found.
50	Record not present.
51	Overflow in record.
52	File too large.

60 File open for write.
61 File not open.
62 File not found.
63 File exists.
64 File type mismatch.
66 Illegal track or sector.
67 Illegal system track or sector.
70 No channels available.
71 Directory error.
72 Disk full or directory full.
73 Power up message or write attempt with DOS mismatch.
74 Drive not ready,(door open or disk not initialized).

DOS ERROR DESCRIPTIONS

Errors below 20 are to be ignored except for 01 which is the number of files scratched by the scratch command.

20: 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.

23: Read Error (checksum error in the data block)

Indicates an error in one or more of the data bytes. Data has been read into the DOS memory but the checksum over the data is in error. May also indicate a grounding problem.

24: Read Error (byte decoding error)

Data or header has been read into DOS memory but a hardware error has been created due to an invalid bit pattern in the data byte. May also indicates a grounding problem.

25: Write Error (write-verify error)

Generated if the controller detects a mismatch between the written data and the data in the DOS memory.

26: Write Protect On

Generated when the controller has been requested to write a data block while the write protect switch is open. Typically caused if the write protect tab is not placed over the notch.

27: Read Error (checksum error in header)

Controller detected an error in the header of the requested data block; it
has not been read into the DOS memory. May also indicate a ground problem.

29: Disk ID Mismatch

Generated when controller has been requested to access a disk which has not been initialized. May also be that the disk has a bad header.

30: Syntax Error (general syntax)

DOS cannot interpret commands sent to the command channel. Typically caused by an illegal number of file names or patterns are incorrectly used, i.e., two file names may appear on the left side of the copy command.

31: Syntax Error (invalid command)

DOS does not recognize command. Command must start in the first position.

32: Syntax Error (long line)

Command sent is longer than 58 characters.

33: Syntax Error (invalid file name)

Pattern matching is invalidly used in the OPEN or SAVE command.

34: Syntax Error (no file given)

File name was left out of a command or DOS does not recognize it as such. Typically it is a colon (:) left out of the command.

39: Syntax Error (invalid command)

Generated when command sent to the command channel (secondary address 15) is not recognized by the DOS.

50: Record Not Present

Result of the disk reading past the last record through INPUT# or GET# commands. Message will also occur after positioning to a record beyond the end of file when in a relative file. This is also used as a warning not to issue an input or a get command when expanding the file with a PRINT# command.

51: Overflow In Record

PRINT# statements exceed record boundry. Information is truncated. May be due to adding extra character for the carriage return and not allowing for it.

52: File Too Large

Record position within a relative file indicates that the disk will overflow.

60: Write File Open

Generated when a file that has not been closed and is being opened for reading.

61: File Not Open

Generated when a file is being accessed and it has not been opened in the DOS. Sometimes the message is not generated as the request is totally ignored.

62: File Not Found

The requested file does not exist on the disk.

63: File Exists

The file name being created already exists on the disk.

64: File Type Mismatch

The file type does not match the file in the directory for the requested file.

66: Illegal Track and Sector

The DOS has attempted to access a track or sector that does not exist in the format being used. It may indicate a problem in reading the pointer to the next block.

67: Illegal System T or S

This special error message indicates an illegal system track or sector.

70: No Channel

The requested channel is not available or all channels are in use. A maximum of five sequential files may be open at one time to the DOS. Direct access channels may have six opened files.

71: Directory Error

The BAM does not match the internal count. There is a problem in the BAM allocation or the BAM has been overwritten in the DOS memory. The disk may be initialized again to restore the BAM in memory but some active files may be terminated by doing this. BAM = Block Availability MAP.

72: Disk Full

Error message is sent when two blocks are available on the drive to allow the current file to be closed. The only limit is determined by the disk size.

73: DOS Mismatch (73,CBM DOS V7.0, 8061/8062)

DOS configurations, 1 and 2 are read compatible but not write compatible. A disk formatted to one program can not be written to by a drive that does not have the same formatting program. Diskettes with different formats can read the data and transfer it to memory so it can be copied or the disk can be reformatted.

74: Drive Not Ready

An attempt has been made to access either the 8061 or 8062 with an incorrectly formatted disk in either drive. The "formatter" program on the utility disk must be used.

QUICK REFERENCE: DISK COMMANDS

This quick reference guide simplifies finding various DOS Support commands in BASIC 3.0 and BASIC 4.0 systems.

BASIC 3.0 commands are upward compatible with BASIC 4.0. All disk commands used with a 2040 floppy disk drive are upward compatable with the 8061 and 8062 drives. Table 4 lists commands in BASIC 3.0, BASIC 4.0 and in Universal DOS Support. BASIC 4.0 is made easier to use by the use of the DOS Support Language as follows:

1. The DSAVE and DLOAD commands eliminate the need to specify the device number each time files are stored or retrieved.

2. Directory display is now a one-step procedure and does not interferes with the program in memory.

3. It is no longer necessary to write a program to read the error channel. The variable DS$ contains the error message.

Table 9-1. Users Quick Reference - Disk Commands

BASIC 3.0	UNIVERSAL DOS SUPPORT	BASIC 4.0
SAVE"DR:FN",8	SAVE"DR:FN",8	DSAVE,"FN",DDR (drive defaults to 0)
LOAD"DR:FN",8	/DR:FN (searches both drives)	DLOAD"FN",DDR (drive defaults to 0)
LOAD"*",8 RUN	*	SHIFTed RUN STOP RUN
LOAD"DR:FN",8	DR:FN	DLOAD"FN",DDR run
10 OPEN1,8,15 20 INPUT#1A,B$,C,D 30 PRINTA,B$,C,D	>RETURN	?DS$ or ?DS (DS$ is number of error only)

NOTE: Assume that OPEN1,8,15 has already been typed for all of the PRINT commands in the following formats. Commands may be spelled out or abbreviated by the first letter as illustrated.

(initalize) PRINT#1,"IX"	>IX	PRINT#1,"IX"
(validate) PRINT#1,"VDR"	>VD	COLLECT DDR
(scratch) PRINT#1,"SDR:FN"	>SDR:FN	SCRATCH"FN",DDR
(copy all) PRINT#1,"CDDR=SDR"	>CDDR=SDR	COPY DSDR TO DDDR
(copy one file) PRINT#1,"CDDR:DFN= DR:SFN	>CDR:DFN=DR:SFN	COPY DDR,"SFN" to DDR,"DFN"
(concatenate) PRINT#1,CDR:DJN= dr:sfn1,dr:sfn2,...	>CDR:DFN=DR:SFL, dr:sfn2,...	CONCAT DDR,"SFN" to DDR,"DFN"
(rename file) PRINT#1,"RDR:DFN= SFN	>RDR:DFN=SFN"	RENAME DDR,"SFN" to "DFN"