APPENDIX B: SUMMARY OF CBM FLOPPY 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.
21	Sync character not found.
22	Data block not present.
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.
28	Data extends into next block.
29	Disk id mismatch.
30	General syntax error
31	Invalid command.
32	Long line.
33	Invalid filename.
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.
65	No block.
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.

DESCRIPTION OF DOS ERROR MESSAGES

NOTE: Error message numbers less than 20 should be ignored with the exception of 01 which gives information about the number of files scratched with 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 block number, or the header has been destroyed.

21: READ ERROR (no sync character)
The disk controller is unable to detect a sync mark on the desired track. Caused by misalignment of the read/writer head, no diskette is present, or unformatted or improperly seated diskette. 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 block 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 as 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.

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 a protect tab over the notch.

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.

28: WRITE ERROR (long data block)
The controller attempts to detect the sync mark of the next header after writing a data block. If the sync mark does not appear within a pre-determined 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.

30: 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 (invalid command)
The 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)
the file name was left out of a command or the DOS does not recognize it as such. Typically, a colon (:) has been left out of the command.

39: SYNTAX ERROR (invalid command)
This error may result if the command sent to command channel (secondary address 15) is unrecognized by the DOS.

50: RECORD NOT PRESENT
Result of disk reading past the last record through INPUT#, or GET# commands. This message will also occur after positioning to a record beyond end of file in a relative file. If the intent is to expand the file by adding the new record (with a PRINT# command), the error message may be ignored. INPUT or GET should not be attempted after this error is detected without first repositioning.

51: OVERFLOW IN RECORD
PRINT# statement exceeds record boundary. Information is cut off. Since the carriage return is sent as a record terminator is counted in the record size. This message will occur if the total characters in the record (including the final carriage return) exceeds the defined size.

52: FILE TOO LARGE
Record position within a relative file indicates that disk overflow will result.

60: 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 track and sector available with the next highest number. If the parameters are zero (0), then all blocks higher in number are in use.

66: ILLEGAL TRACK AND SECTOR
The DOS has attempted to access a track or block which does not exist in the format being used. This may indicate a problem reading the pointer to the next block.

67: ILLEGAL SYSTEM T OR S
This special error message indicates an illegal system track or block.

70: 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: 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 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. NOTE: BAM = Block Availability Map

72: DISK FULL
Either the blocks on the diskette are used or the directory is at its entry limit. DISK FULL is sent when two blocks are available on the 1541 to allow the current file to be closed.

73: DOS MISMATCH (73, CBM DOS V2.6 1541)

DOS 1 and 2 are read compatible but not write compatible. Disks may be interchangeably read with either DOS, but a disk formatted on one version cannot be written upon with the other version because the format is different. This error is displayed whenever an attempt is made to write upon a disk which has been formatted in a non-compatible format. (A utility routine is available to assist in converting from one format to another.) This message may also appear after power up.

74: DRIVE NOT READY

An attempt has been made to access the 1541 Single Drive Floppy Disk without any diskettes present in either drive.

APPENDIX C: Demonstration Disk Programs

1. DIR

```
4 OPEN2,8,15
5 PRINT"■":GOTO 10000
10 OPEN1,8,0,"$0"
20 GET#1,A$,B$
30 GET#1,A$,B$
40 GET#1,A$,B$
50 C=0
60 IF A$<>"" THEN C=ASC(A$)
70 IF B$<>"" THEN C=C+ASC(B$)*256
80 PRINT"■"MID$(STR$(C),2);TAB(3);"■";
90 GET#1,B$:IF ST<>0 THEN 1000
100 IF B$<>CHR$(34) THEN 90
110 GET#1,B$:IF B$<>CHR$(34)THEN PRINTB$;:GOTO110
120 GET#1,B$:IF B$=CHR$(32) THEN 120
130 PRINT TAB(18);:C$=""
140 C$=C$+B$:GET#1,B$:IF B$<>""  THEN 140
150 PRINT"■"LEFT$(C$,3)
160 GET T$:IF T$<>"" THEN GOSUB 2000
170 IF ST=0 THEN 30
1000 PRINT" BLOCKS FREE"
1010 CLOSE1:GOTO 10000
2000 IF T$="Q" THEN CLOSE1:END
2010 GET T$:IF T$="" THEN 2000
2020 RETURN
4000 REM DISK COMMAND
4010 C$="":PRINT">";
4011 GETB$:IFB$="" THEN4011
4012 PRINTB$;:IF B$=CHR$(13) THEN 4020
4013 C$=C$+B$:GOTO 4011
4020 PRINT#2,C$
5000 PRINT"■";
5010 GET#2,A$:PRINTA$;:IF A$<>CHR$(13)GOTO5010
5020 PRINT"■"
10000 PRINT "D-DIRECTORY"
10010 PRINT ">-DISK COMMAND"
10020 PRINT "Q-QUIT PROGRAM"
10030 PRINT "S-DISK STATUS "
10100 GETA$:IFA$=""THEN10100
10200 IF A$="D" THEN 10
10300 IF A$="." OR A$=">" OR A$=">" THEN 4000
10310 IF A$="Q" THEN END
10320 IF A$="S" THEN 5000
10999 GOTO 10100
```

10. ASSEMBLY LANGUAGE AND THE 1541

If you want to use your 1541 disk drive to manipulate data directly from assembly language you can use the information presented below.

Here is a list of subroutines that provide the start of memory locations in each of the Kernal routines. These routines are used in conjunction with the assembly language command JSR to jump to that subroutine location in memory:

```
SUBROUTINE
SETLFS  = $FFBA     ; set logical, physical & secondary addresses
SETNAM  = $FFBD     ; save length & address of filename
OPEN    = $FFC0     ; open a logical file
CLOSE   = $FFC3     ; close a logical file
CHKIN   = $FFC6     ; open a channel for input
CLRCH   = $FFCC     ; clear all channels
BASIN   = $FFCF     ; get a byte from a file
BSOUT   = $FFD2     ; output a character to the screen
```

For a more complete description as to what each routine does and what parameters are passed to and from each routine, see your Commodore 64 or VIC-20 Programmer's Reference Guide.

Now, for a practical application of the subroutines listed above, here is a sample program using those routines to read a sequential file on a disk. Assume that you have stored the filename "TEST" at $C000.

```
INIT                ; initialize:
LDA #$04            ; filename length
LDX #$00            ; low byte of filename address
LDY #$C0            ; high byte of filename address
JSR SETNAM          ; save length & address of filename
LDA #$03            ; logical address
LDX #$08            ; device number
LDY #$00            ; secondary address (0 = read seq. file)
JSR SETLFS          ; set logical, physical & secondary addresses
JSR OPEN            ; open logical file
LDX #$03            ; set x-register to logical address
JSR CHKIN           ; open input channel
your program info   ; get data and print it one byte at a time
JSR BASIN           ; get one byte
BEQ END             ; if 0 then end of file or error
JSR BSOUT           ; output character to the screen
JMP your prog. info ; loop
END
LDA #$03            ; set accumulator to logical address
JSR CLOSE           ; close file
JSR CLRCH           ; clear channels and reset defaults
RTS                 ; end of assembly language program
```


APPENDIX A: DISK COMMAND SUMMARY

General Format: PRINT#file#, command

COMMAND	
NEW	"N
COPY	"C:new file=:orginal file
RENAME	"R:new name=old name
SCRATCH	"S:file name
INITIALIZE	"I
VALIDATE	"V
DUPLICATE	not for single drives
BLOCK-READ	"B-R:" channel; drive; track; block
BLOCK-WRITE	"B-W:" channel; drive; track; block
BLOCK-ALLOCATE	"B-A:" drive; track; block
BLOCK-FREE	"B-F:" drive; track; block
BUFFER-POINTER	"B-P:" channel; position
USER1 and USER2	"Un:" channel; drive; track; block
POSITION	"P" CHR$(channel#) CHR$(rec#lo) CHR$(rec#hi) CHR$(position)
BLOCK-EXECUTE	"B-E:" channel; drive; track; block
MEMORY-READ	"M-R" CHR$(address lo) CHR$(address hi)
MEMORY-WRITE	"M-W" CHR$(address lo) CHR$(address hi) CHR$(#chars) "data"
MEMORY-EXECUTE	"M-E" CHR$(address lo) CHR$(address hi)
USER Commands	"Un"

NEW to format N:name,id

NEW erase directory N:name

4. DISK COMMANDS

OPEN AND PRINT

Up 'til now, you have explored the simple ways of dealing with the disk drive. In order to communicate with the disk drive more fully, you have to touch on the OPEN and PRINT# statements in BASIC (more details of these commands are available in your VIC 20 or Commodore 64 User's Guide or Programmer's Reference Guide). You may be familiar with their use with data files on cassette tape, where the OPEN statement creates the file and the PRINT# statement fills the file with data. They can be used the same way with the disk, as you will see in the next chapter. But they can also be used to set up a command channel. The command channel lets you exchange information between the computer and the disk drive.

FORMAT FOR THE OPEN STATEMENT:

OPEN file#, device#, channel#, text$

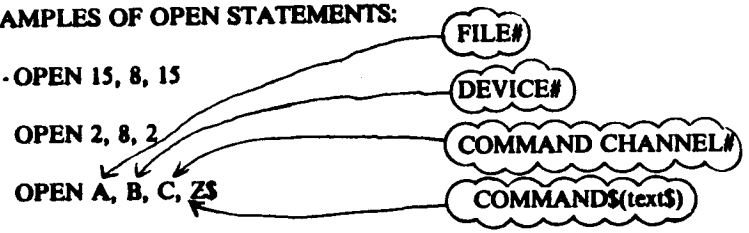
The file# can be any number from 1 to 255. This number is used throughout the program to identify which file is being accessed. But numbers greater than 127 should be avoided, because they cause the PRINT# statement to generate a linefeed after the return character. These numbers are really meant to be used with non-standard printers.

The device# of the disk is usually 8.

The channel# can be any number from 2 to 15. These refer to a channel used to communicate with the disk, and channels numbered 0 and 1 are reserved for the operating system to use for LOADing and SAVEing. Channels 2 through 14 can be used for data to files, and 15 is the command channel.

The text$ is a string that is PRINTed to the file, as if with a PRINT# statement. This is handy for sending a single command to the channel.

EXAMPLES OF OPEN STATEMENTS:



The PRINT# command works exactly like a PRINT statement, except that the data goes to a device other than the screen, in this case to the disk drive. When used with a data channel, the PRINT# sends information into a buffer in the disk drive, which LOADs it onto the diskette. When PRINT# is used with the command channel, it sends commands to the disk drive.

FORMAT FOR SENDING DISK COMMANDS:

OPEN 15, 8, 15, command$

or

PRINT# 15, command$

NEW

This command is necessary when using a diskette for the first time. The NEW command erases the entire diskette, it puts timing and block markers on the diskette and creates the directory and BAM. The NEW command can also be used to clear out the directory of an already-formatted diskette. This is faster than re-formatting the whole disk.

FORMAT FOR THE NEW COMMAND TO FORMAT DISK:

PRINT#15, "NEW:name,id"

or abbreviated as

PRINT#15, "N:name,id"

FORMAT FOR THE NEW COMMAND TO CLEAR DIRECTORY:

PRINT#15, "N:name"

The name goes in the directory as the name of the entire disk. This only appears when the directory is listed. The ID code is any 2 characters, and they are placed not only on the directory but on every block throughout the diskette. That way, if you carelessly replace diskettes while writing data, the drive will know by checking the ID that something is wrong.

COPY

This command allows you to make a copy of any program or file on the disk drive. It won't copy from one drive to a different one (except in the case of dual drives like the 4040), but it can duplicate a program under another name on the drive.

FORMAT FOR THE COPY COMMAND:

PRINT# 15, "COPY:newfile= oldfile"

or abbreviated as

PRINT# 15, "C:newfile= oldfile"

The COPY command can also be used to combine two through four files on the disk.

FORMAT FOR COPY TO COMBINE FILES:

PRINT# 15, "C:newfile= oldfile1, oldfile2, oldfile3, oldfile4"

EXAMPLES OF COPY COMMAND:

PRINT# 15, "C:BACKUP= ORIGINAL"
PRINT# 15, "C:MASTERFILE= NAME, ADDRESS, PHONES"

RENAME

This command allows you to change the name of a file once it is in the disk directory. This is a fast operation, since only the name in the directory must be changed.

FORMAT FOR RENAME COMMAND:

PRINT# 15, "RENAME:newname=oldname"

or abbreviated as

PRINT# 15, "R:newname=oldname"

EXAMPLE OF RENAME COMMAND:

PRINT#15, "R:MYRA=MYRON"

The RENAME command will not work on any files that are currently OPEN.

SCRATCH

This command allows you to erase unwanted files and programs from the disk, which then makes the blocks available for new information. You can erase programs one at a time or in groups by using pattern matching and or wild cards.

FORMAT FOR SCRATCH COMMAND

PRINT# 15, "SCRATCH:name"

or abbreviated as

PRINT# 15, "S:name"

If you check the error channel after a scratch operation (see below), the number usually reserved for the track number now tells you how many files were scratched. For example, if your directory contains the programs KNOW and GNAW, and you use the command PRINT# 15, "S:?N?W", you will scratch both programs. If the directory contains TEST, TRAIN, TRUCK, and TAIL, and you command the disk to PRINT# 15, "S:T*", you will erase all four of these programs.

INITIALIZE

At times, an error condition on the disk will prevent you from performing some operation you want to do. The INITIALIZE command returns the disk drive to the same state as when powered up. You must be careful to re-match the drive to the computer (see chapter 2).

FORMAT FOR INITIALIZE COMMAND:

PRINT# 15, "INITIALIZE"

or abbreviated as

PRINT# 15, "I"

VALIDATE

After a diskette has been in use for some time, the directory can become disorganized. When programs have been repeatedly SAVEd and SCRATCHed, they may leave numerous small gaps on the disk, a block here and a few blocks there. These blocks never get used because they are too small to be useful. The VALIDATE command will go in and re-organize your diskette so that you can get the most from the available space.

Also, there may be data files that were OPENed but never properly CLOSEd. This command will collect all blocks taken by such files and make them available to the drive, since the files are unusable at that point.