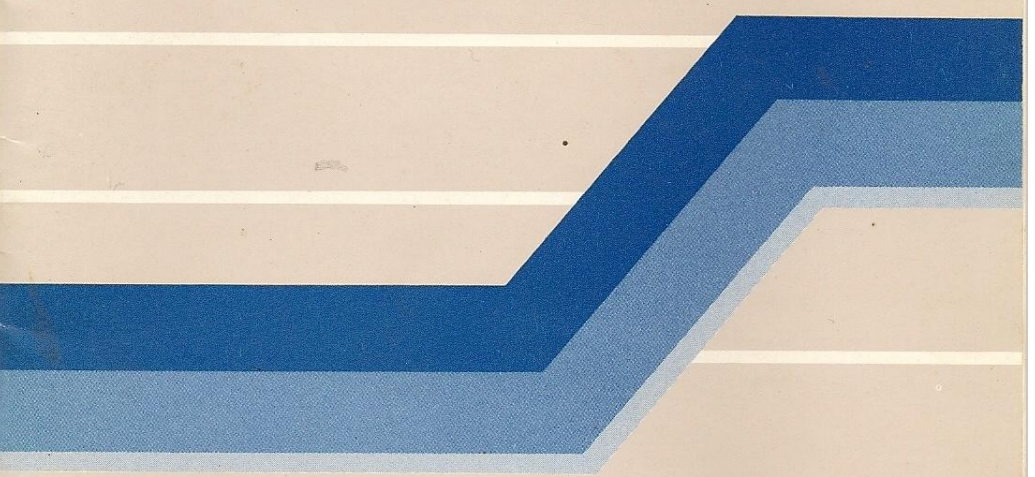
RF500C SERIES

DISK DRIVE

Operational Manual



RF512C,RF501C and RF502C

Operational Manual

TABLE OF CONTENTS

GENERAL

This manual has two primary purposes. The First is to teach you how to use the **BASIC** programming language. The chapters of this manual use examples to accompany with explenation of how the various **BASIC** commands work. The second purpose of this manufal is to serve as a reference guide to disk file structure. The appendices and the specificiations give you more information in detail.

The **RF500C** is a floppy disk drive which allows you to store and retrieve information much more quickly and conveniently than you can with tape.

This manual can be applied to many versions of **BASIC** that operate in different computers. The **BASIC** is a popular language that let you communicate with your **RF500C** floppy disk drive. Because different computer says different **BASIC** language, you must make sure what kind of computer you use, to obey its **SYNTAX** for your convenience.

The table below lists the various Commodore computers and the versions of **BASIC** each contains. Remember, higher version **BASIC** has more sophisticate functions; and can compatible with lower versions.

CBM computer	BASIC interpreter
RET 2000	1.0
VIC-20	2.0
C-64	2.0
CBM 3000	3.0
C-16	3.5
PLUS 4	3.5
CBM 8000	4.0
C-128	7.0

INSTALLATION AND HANDLING

UNPACKING

Your disk drive pac comes with several items. These are :

1. The disk drive (the main box)
2. A serial cable
3. A power adaptor (120 volt AC for USA, 240 volt AC for Europe)
4. This operation manual

Save the packing material in case you wish to transport your drive — or even in the unlikely case that you must return it to your dealer or to the factory for service.

IMPORTANT NOTE

Before connecting or disconnecting anything on the drive or computer, **TURN OFF THE POWER**. This is a must!

CARE OF THE DRIVE AND DISKETTES

The disk drive, unlike the computer, is a mechanical device with motors and moving parts. Therefore it is somewhat more delicate than the computer. Rough handling, such as dropping the drive or having things drop on it, can cause it to malfunction.

The diskette is a plastic disk with magnetic coating, so that information can be stored on and erase out from its surface. The coating is similar to the magnetic coating on recording tape. The diskette is permanently packed in a square black plastic cover which is used to protect it. Please keep it clean and allow it to spin freely. This package is never opened.

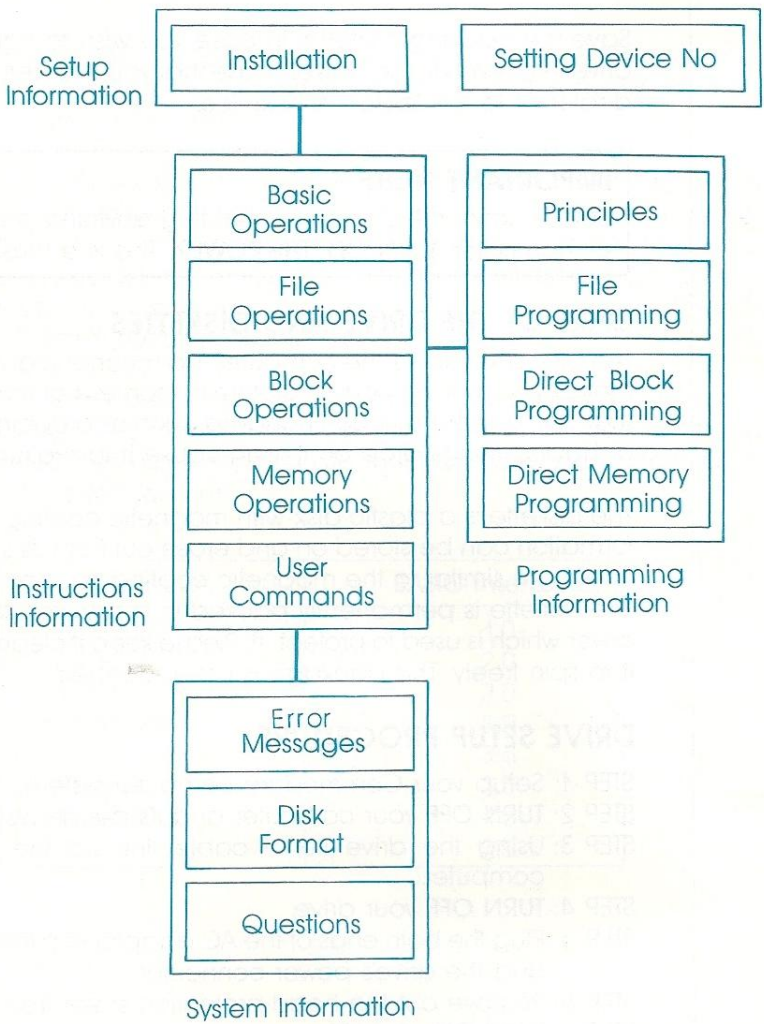
DRIVE SETUP PROCEDURE

STEP 1: Setup your Commodore computer system.
STEP 2: **TURN OFF** your computer and its periphrals.
STEP 3: Using the drive serial cable link up the drive and computer.
STEP 4: **TURN OFF** your drive.
STEP 5: Plug the both ends of the AC adaptor into the wall outlet and the drive's power connector.
STEP 6: Remove out the head-protection sheet from the drive.
STEP 7: **TURN ON** your drive.
STEP 8: **TURN ON** the computer.
STEP 9: It is ready and type: **10 PRINT "DEMO". RETURN**

GETTING STARTED

MANUAL ORGANIZATION

This manual contains general information about, how the drive works and specific information for each operation. The orangization of this manual, refer to the chart. You can quickly find out the information what you need.



Learning how to use the drive and its **BASIC** statements or **DOS** direct commands with emphasis on a few special instructions, and several of them are straightforward extensions of fundamental commands. This manual assumes that user are familiar with Commodore computer, and feel comfortable in writing simple **BASIC** programs.

You can use these commands to instruct the dive as you desire

Basic operations	NEW, HEADER, SAVE, BOOT, LOAD, VERIFY, DIRECTORY, RENAME, SCRATCH, COPY, INITIALIZE, CLOSE, OPEN, etc.
File operations	CLOSE, OPEN, GET#, INPUT#, PRINT#, APPEND, RECORD, CONCAT, AUTO
Block operations	COLLECT, VALIDATE, B-A, B-F, B-R, B-W, B-P, and B-E
Momry operations	M-R, M-W, M-E
User commands	HEAD, SIDE and U

PRINCIPLES

One of the most important advantages of this drive is that information can be stored and retrieved by a name under which it is filed.

The programs that automatically keep track of files, save and retrieve information, and do a multitude of other housekeeping tasks are called the Disk Operating System, usually shortened to "**DOS**".

In this **DOS**, the path which link between the computer and peripheral is called channel. Each peripheral must be assigned for a specific device number (usually **8** for disk drive) and a logical file is specified for data transfer.

There have 15 available channels, and channel 15 is reserved as the command channel. The logical file number can be assigned from 1 to 127. The disk drive device number can be specified from 1 to 32 by software (but only 8,9,10 and 11 can be set by hardware switch). The drive number is always 0 (zero) in single disk drive system.

So, the typical form for sending command to the drive is listed below:

Example: OPEN 1,8,15, "**#**" : **REM** open logical file 1 for device 8 through channel 15 and the message is "**#**".

In **GCR** system, a disk is formatted 35 tracks/side and the optimal sectoring is shown as below:

Track	Sector Assignment
1-17	0-20
18-24	0-18
25-30	0-17
31-35	0-16

In RF512C, the tracks on side one are assigned from 36 to 70 with sector no. as track 1 to 35, respectively.

In every formatted disk, there is a directory which contains disk information (disk name, disk **ID**, file table and bit-map for block status indication, see appendix). The directory file is located on track 18. (For more detail, see appendix)

The information collection of the disk is called file. From the file **VECTOR** table (in the directory), we know the location of where the sector of the file starts. Follow the pointer of each sector then the entire file could be found and the program file will be loaded in computer memory automatically.

The built-in **RAM** is not only support for the system operation but also shorten the internal data processing time. It can enlarge the I/O buffer to speed up the transfer rate.

For more advanced user, he/she can write program on the memory. Then, a multiple processing environment one can enjoy.

Here, is the table of **RAM** usage.

RAM page	Contents
0	system register
1	buffer for GCR
2	drive information
3	#0 buffer zero
4	#1 buffer one
5	#2 buffer two
6	#3 buffer three
7	#4 buffer four

COMMANDS AND STATEMENTS

Syntax — Command name.
Function Discuss the command type and how it works.

Format

Language and version	**KEYWORD** (arguments...) [optional arguments...]

Syntax **The KEYWORDs** appear in upper-case letters.
The **KEYWORDs** are words that are part of **BASIC** or strings that are commands of **DOS.**
The arguments appear in lower-case letters.
Where # means a real or integer variable
$ means a string variable
[ ] means such arguments are optional parameters
( ) means that parentheses must come with variable
" " means argument that within quotation marks must not be variables
Note: In some commands, drive no. (0 or 1), device no. (8, 9, 10 or 11), **RAM** bank no. (0-15), start address location and end address location, which are preceded by letters **D,U,B,P,** and **TO P,** respectively.
In wildcard string, "?" stands for any character in that position, and "*" stands to full the rest of filename with "?"s.

Example: We wish you can satisfy on our discussion and try each example joyfully.

NEW/HEADER

Function New is a direct command for formatting a disk.
HEADER is an interactive command for formatting or to clear the directory of a disk.

Format

BASIC	string$ = "N[EW] : diskname, id"
BASCI 7.0	HEADER "diskname [,id]" [,D drive#] [,U device#]

Syntax Remember, NEW is a direct command that should be sent through the command channel and its mnemonic is N.
But HEADER is an interactive type interpret command.
When it works, echo message is "ARE YOU SURE?".
The answer may be "Y" or "N".
When id code isn't assigned, then the disk directory is cleared (if your disk is already formatted).
The diskname must not longer than 16 characters.
The id code is any 2 characters identifier.
The device# default value is 8.
The drive# default value is 0.
Where the preceding letters D and U please refer to SYNTAX.

Examples: Using NEW command

```
CLOSE 1: OPEN 1,8,15, "N: TEST DISK, 1A"
```

Using HEADER command

```
HEADER "TEST DISK, 1A"      (Type that line)
ARE YOU SURE? Y             (Answer "Y" for yes and
                             wait for "ready" prompt
```

SAVE/BSAVE/DSAVE

Function These interpreter commands save a copy of the specific computer memory to the disk as a file.

Format

BASIC	SAVE "filename", device# SAVE filename$, device#
BASCI 7.0	DSAVE "filename"[,D drive#] [,U device#] BSAVE "filename"[,D drive#] [,U device#] B bank#,P start# TOP end# DSAVE (filename$) [,D drive#] [,U device#]

Syntax The filename must be a string that is enclosed in the quotation marks and can't be the same as any existing file's name; otherwise, you should type as following:

SAVE "@: filename", device#

Where @: means the old file is replaced by the new one.
The device# default value is 8.
The drive# default value is 0.

BSAVE command is specified for directly save the computer memory onto the disk as a file. Where the preceding letters D, U, B and P please refer to SYNTAX.

Examples:
```
SAVE "DEMO",8
DSAVE "DEMO"
BSAVE "PICTURE",D0,U8,P8192 TO P 16384
SAVE "@: DEMO",DV:REM(The "DEMO" file is updated on the disk drive number DV).
```

RUN/BOOT

Function These interpreter commands can load and execute program file.
The **BOOT** command is specified for binary file.

Format

BASIC 7.0	RUN "filename"[,D drive#] [,U device#] RUN (filname$) BOOT ["filename"] [,D drive#] [,U device#]

Syntax The filename must be a string (i.e. name of program file).
The device# default value is 8.
The drive# default value is 0.
If no filename is assingned to the **BOOT** command, then the boot program file is implied.
Where the preceding letter **D** and **U** please refer to **SYNTAX**.

Example: **RUN "DEMO"** : **REM "DEMO"** on device 8
BOOT (If a **CP/M** master disk is inserted, your Commodore **128** will operate under **CP/M** mode.)

LOAD/BLOAD/DLOAD

Function These interpreter commands load a file (e.g. program) from disk to computer memory.

Format

BASIC	LOAD "filename", device#[,command#] LOAD filename$, device#[,command#]
BASIC 7.0	DLOAD "filename"[,D drive#] [,U device#] BLOAD "filename"[,D drive#] [,U device#] [,B bank#] [,P start#] DLOAD (filename) [,D drive#] [,U device#]

Syntax The filename must be a string that is enclosed in the quotation marks and such string is also a name of disk file.

The device# default value is 8.

The drive# default value is 0.

The command# default value is 0 that means the program file is loaded at the **BASIC** program start location.

When the command# is 1, then the program file will be loaded at the same location where it was saved from.

BLOAD command is specified for binary file to load to computer quickly. Where the preceding letters **D,U,B** and **P** please refer to **SYNTAX.**

Examples:
```
LOAD  "DEMO", 8
DLOAD "DEMO"
BLOAD "PICTUE",D0,U8,0,P8192
LOAD  "*",8,1:REM (This command is suitable for
                   loading games.)
```

VERIFY/DVERIFY

Function These interpreter commands are used to compare the program file on disk with the program memory in computer.
An "OK" echo will respone for verify pass.

Format

BASIC	VERIFY "filename"[,device#] [,command#]
BASIC 7.0	DVERIFY "filename"[,D drive#] [,U device#]

Syntax The filename must be a string that is enclosed in quotes and such string is also a name of disk file.
The device# default value is 8.
The drive# default value is 0.
The command# default value is 0, that means the program file is verified with the BASIC program. When the command# is 1, then the program file will be compared with the location from which it was saved.
Where the preceding letters D and U please refer to SYNTAX.

Example:
```
VERIFY "DEMO",8
OK
```

DIRECTORY/CATALOG

Function These interpreter commands display the disk directory without destory program memory.

Format

BASIC	LOAD"$",device# CATALOG [D drive#] [,U device#] [,"wildcard"]
BASIC 7.0	DIRECTORY [D drive#] [,U device#] [,"wildcard"] DIRECTORY [D drive#] [,U device#] [(wildcard$)]

Syntax The device# default value is 8.
The drive# default value is 0.
A sub-directory can be selected by the wildcard (if it is presented).
Where D,U and wildcard please refer to SYNTAX.

Example: F3 : Press function key F3. (for C128)
DIRECTORY and its contents is displayed.
DIRECTORY "D*": Only file which with "D" for the first letter will be shown.

LOAD "$",8
LIST: You will get the same result but your BASIC program will be erased.

RENAME

Function This interpreter command is used to change the filename without affect its contents.
R is a mnemonic of rename direct command.

Format

BASIC	String$="R[ENAME] : : newfile=oldfile"
BASIC 7.0	RENAME [D drive#,] "oldfile" TO "newname" [,U device#] RENAME [D drive#,] (oldfile$) TO (newfile$) [,U device#]

Syntax Remember, there has no single instruction in BASIC 2.0 that's equilvent to RENAME, so the first format can only work in multi-command form.
The newfile and oldfile must be strings (i.e. oldfile should be some name of disk file).
The device# default value is 8.
The drive# default value is 0.
Where the preceding letters D and U please refer to SYNTAX.

Example: RENAME "CHECKER" TO "AGAIN"
(Using DIRECTORY to confirm the result.)

SCRATCH

Function This interactive interpreter command is used to erase file by delete it from directory.
S is the mnemonic of scratch direct command.

Format

BASIC	string$ = "S[CRATCH] : filename"
BASIC 7.0	SCRATCH "filename"[,D drive#] [,U device#]

Syntax Remeber, there has no single instruction in **BASIC** 2.0 that's equilvent to **SCRATCH**, so the first format can only works in multi-command .form.
The filename must be a string (i.e. should be some name of disk file).
The device# default value is 8.
The drive# default value is 0.
Where the preceding letters **D** and **U** please refer to **SYNTAX.**

Example: **SCRATCH "AGAIN"**
"ARE YOU SURE?" (Simple answer "**Y**" or "**N**")
"01, **FILES SCRATCHED**, 01, 00" (Echo message)

COPY/BACKUP

Function These interpreter commands copy file(s) or the whole disk to another file or disk in the same unit so they can be considered as files combination commands.
C is the mnemonic of copy direct command.

Format

BASIC	string$="C[OPY] : copyfile=sourcefile 1 [, sourcefile 2]...
BASIC 7.0	COPY "sourcefile"[,D drive#] TO "copyfile" [D drive# ,U device#] BACKUP D sourcedrive# TO D copydrive# [,U device#]

Syntax Remember, there has no single instruction in **BASIC** 2.0 that's equilvent to **COPY** or **BACKUP**, so the first format can only works in multi-command form.
The copyfile and sourcefile must be strings (i.e. sourcefile should be some name of disk file).
If no filename is assigned to **COPY** command, then **COPY** all files action will take.
The device# default value is 8.
The drive# default value is 0.
Where the preceding letter **D** and **U** please refer to **SYNTAX.**

Example:
```
10 CLOSE 1
20 OPEN 1,8,15, "C : DEST=DEMO,CHECKER"
30 CLOSE 1
```
(You can check the directory to confirm the size of **DEST** that is equal to the length of **DEMO** plus **CHECKER**'s length.)
In single drive system, **BACKUP disk is nonsence.**

INITIALIZE

Function This direct command is a warm start function for the drive. Other then warm start, it also can re-read the current disk's **BAM** for more reliable operation.
I is the mnemonic of intialize.

Format

BASIC	string$="**I**[**NITIALIZE**] [drive#]"

Syntax Remember, there has no single instruction in **BASIC** for execution.
The drive# default value is 0.

Example:
```
10 CLOSE 1 : OPEN 1,8,15, "I0"
20 INPUT# 1,N,M$ : PRINT N;M$
30 CLOSE 1
```

CLOSE/DCLOSE/DCLEAR

Function These interpreter commands is used to close a logical file or files.

Format

BASIC	CLOSE file#
BASIC 7.0	DCLOSE [# file#] [,U device#] DCLEAR [D drive#] [,U device#]

Syntax file# is the specific file to be closed.
If no file# is specified in **DCLOSE**, then all open disk files are closed.
DCLEAR clear all open channel on disk drive.
The device# default value is 8.
The drive# default value is 0.
Where the preceding letters **D** and **U** please refer to **SYNTAX**.

Example:
```
10 For I=1 to 10
20 CLOSE I
30 NEXT
```

REM: close logical file from 1 to 10

OPEN/DOPEN

Function OPEN is an interpreter command, which create a file by opening a channel for computer-drive communication.
DOPEN is a BASIC 7.0 interpret file control command.

Format

BASIC	OEPN file#,device#,channel#[,string$] OEPN file#,device#,channel#[,"filename,type,mode"]
BASIC 7.0	DOPEN#file#, "filename[,SorP]", [,Lrecord length#] [,D drive#] [,U device] [,W]

Syntax The file# value is from 1 to 255.
The device# default value is 8.
The drive# default value is 0.
The channel# volue is from 1 to 15

type: S for sequential file
P for program file
U for user file
L for length of a relative file
A for append
mode: R for read
W for write
M for modify

Where the preceding letters D and U please refer to SYNTAX.

Example: DOPEN#3,"TABLE", L128

(open file number 3 relative file "TABLE" with record length 128 characters.)

GET#/INPUT#

Function These interpreter commands input data from a file to the computer.

Format

BASIC	GET# file#, variable list INPUT# file#, variable list

Syntax Remeber don't separate the keywords with #; otherwise, the computer will give you different response.

The file# must corresponding to the opened logical file. The variables format of GET# and INPUT# please refer to GET and INPUT in BASIC.

Example: refer to chapter FILE PROGRAMMING

PRINT#

Function This interpreter command output data to the specific logical file.

Format

BASIC	PRINT# file#;output list

Syntax Remember don't separate the keyword with #, otherwise, the computer will give you different respone.

The file# must corresponding to the opened logical file. Through that file, the output list data can be transfer to the specific device.

Example:

```
10 OPEN 15,8,15
20 PRINT# 15, "N:TEST,ID"
30 CLOSE 15
```

REM open file no.15, send NEW command to the drive.

APPEND

Function **APPEND** is an interpreter command and **A** is the mnemonic of append direct command, both can reopen an specific sequential file to append data.

Format

BASIC	string$ = "0:filename,A"
BASIC 7.0	APPEND# file#,"filename" [,D drive#, U device#] APPEND# file#,(filename$) [,D drive#,U device#]

Syntax Remember there has no single instruction in **BASIC 2.0** that's equilvent to **APPEND**, so the first format can only work in multi-command form.
The filename must be a string (i.e. name of some disk file).
The device# default value is 8.
The drive# default value is 0.
Where the preceding letters **D** and **U** please refer to **SYNTAX**.

Example: Refer to chapter **FILE PROGRAMMING**

RECORD

Function This interpreter command can set the relative file pointer to any byte of any record in the file.

Format

BASIC	string$ = CHR$ (80)+CHR$ (96+CHANNEL#)+ CHR$[lo-record#]CHR$ (hi-record#) [+CHR$(byte#)]
BASIC 7.0	RECORD# file#,record#[,byte#]

Syntax The file# must correspond to the opened logical file.
The record# value is from 1 to 65535
The byte# value is from 1 to 254
If record# is higher than the preset file record number, additional record (**INPUT#** mode) will be return where record#=256* (hirec#)+(lorec#).

Example: Refer to **FILE PROGRAMMING**

COLLECT/VALIDATE

Function These interpreter commands free asterisk files in directory to get more space for user.
V is the mnemonic of validate direct command.

Format

BASIC	string$ = "V[ALIDATE]"
BASIC 7.0	COLLECT [D drive#] [,U device#]

Syntax Remember, there has no single instruction in **BASIC 2.0** that's equilvent to **BASIC 7.0's COLLECT**, so the first format can only works in multi-command form.
The device# default value is 8.
The drive# default value is 0.
Where the preceding letters **D** and **U** please refer to **SYNTAX**.

Example:
```
10 OPEN 1,8,15
20 PRINT# 1, "V0"
30 CLOSE 1
```

Normally you won't have any asterisk before the file name in directory, so nothing should free.

CONCAT

Function This interpreter command is used to attach data file to the end of the main data file.

Format

BASIC 7.0	CONCAT [D drive#,] "trailfile" TO [D drive#,] "mainfile" [,U device#]

Syntax The trailfile and mainfile must be strings (i.e. should be some names of disk file).
The result of the concatenated file will be saved back to mainfile.
The device# default value is 8.
The drive# default value is 0.
Where the preceding letter **D** and **U** please refer to **SYNTAX**.

Example:
```
CONCAT "LATER" TO "YEAR".
```

AUTO

Function This is a rarely used direct command. It loads the machine program from an user file into the drive and then execute it as function **RUN** does in **BASIC**. **&** is the mnemonic of such command.

Format

BASIC	string$ = "**&** filename"

Syntax Remeber, there has no single instruction in **BASIC** for execution.
The filename must be a string (i.e. should be name of user file).

Example: Few lines of command isn't enough to implement this **ATUO** command.
Create an user file in **AUTO** file format. Then execute your drive program with **&**.

B-A (BLOCK-ALLOCATE)

Function This direct command can make specific block mark as "in use" to avoid overwritten.

Format

BASIC	string$ = "**B-A**";drive#;track#;block#

Syntax This command must be sent through command channel.
The drive# is always zero in single drive system.
The range of track# and block# please refer to **PRINCIPLES**.

B-F (BLOCK-FREE)

Function This direct command can make specific block free by going to update the **BAM**.
Or, it do the inverse work of **BLOCK-ALLOCATE**.

Format

BASIC	string$="B-F";drive#;track#;block#

Syntax This command must be sent through command channel.
The drive# is always zero in single drive system.
The range of track# and block# please refer to **PRINCIPLES**.

B-R (BLOCK-READ)

Function This direct command can read specific block of data on the.

Format

BASIC	string$="B-R";drive#;channel#;track#;block#

Syntax This command must be sent through command channel and cooperate with data channel to read disk data.
The drive# is always zero in single drive system.
The range of track# and block# please refer to **PRINCIPLES**.
The channel# must belong to some opened data channel.

B-W (BLOCK-WRITE)

Function This direct command can write data on the specific block.

Format

BASIC	string$="B-W";drive#;channel#;track#;block#

Syntax This command must be sent through command channel and cooperate with channel# to write data onto the disk.
The drive# is always zero in single drive system.
The range of track# and block# please refer to PRINCIPLES.
The channel# must belong to some opened data channel.

B-P (BLOCK-POINTER)

Function This direct command can move the buffer pointer to individual byte within a block.

Format

BASIC	string$="B-P";channel#;byte#

Syntax This command must be sent through command channel and cooperate with data channel#.
The byte# is the buffer pointer value from 0 to 255.
The channel# must belong to some opened data channel.

B-W (BLOCK-WRITE)

Function This direct command can write data on the specific block.

Format

BASIC	string$="B-W";drive#;channel#;track#;block#

Syntax This command must be sent through command channel and cooperate with channel# to write data onto the disk.
The drive# is always zero in single drive system.
The range of track# and block# please refer to PRINCIPLES.
The channel# must belong to some opened data channel.

B-P (BLOCK-POINTER)

Function This direct command can move the buffer pointer to individual byte within a block.

Format

BASIC	string$="B-P";channel#;byte#

Syntax This command must be sent through command channel and cooperate with data channel#.
The byte# is the buffer pointer value from 0 to 255.
The channel# must belong to some opened data channel.

B-E (BLOCK-EXECUTE)

Function This direct command can execute the buffer which was contained in specific block

Format

BASIC	string$="B-E";drive#;channel#;track#;block#

Syntax This command must be sent through command channeland cooperate with data channel# to get the block code for execution.
The drive# is always zero in single drive system.
The range of track# and block# please refer to **PRINCIPLES**
The Channel# must be some opened data channel.

M-R (MEMORY-READ)

Function This direct command can direct read drive memory contents at any location.

Format

BASIC	string$="M-R"CHR$(lo#)CHR$(hi#)CHR$(length#)

Syntax This command must be sent through command channel.
The lo# means the lower byte of the start address.
The hi# means the higher byte of the start address.
The length# means the number of data which will be read.

M-W (MEMORY-WRITE)

Function This direct command can write data onto the drive RAM.

Format

BASIC	string$ = "M-W"CHR$(lo#)CHR$(hi#)CHR$ (length#) data....

Syntax This command must be sent through command channel.
The lo# means the lower byte of the start address.
The hi# means the higher byte of the start address.
The length# means the number of data which will be written.

M-E (MEMORY-EXECUTE)

Function This direct command can execute at any location either in RAM or ROM.

Format

BASIC	string$ = "M-E"CHR$(lo#)CHR$(hi#)

Syntax This command must be sent through command channel.
The lo# means the lower byte of the execute location.
The hi# means the higher byte of the execute location.

HEAD/SIDE (For RF502C only)

Function Due to many existing program disks are formatted in single side from; if one want to operate with two single side disks, who should use two single side disk drives (like RF501C). Using this command, you can do the same thing with only one drive.

Format

BASIC	string$="US side#"

Syntax Remember, this command must be sent through the command channel.
Where side# means the side or head number (0 or 1).
As reference, please note on
User Command "U0>H side#"

Example: use US in formatting.

```
10 CLOSE 1:OPEN 1,8,15
20 FOR I=0 TO 1
30 PRINT# 1, "US" CHR$(I)
40 PRINT# 1, "N:TEST,ID":NEXT I
50 CLOSE 1
```

USER COMMANDS

Introduction

Other than the high level command, the internal DOS provides some user routines for advanced user.

Common User Command

Format

BASIC	string$="U chara$"

Syntax This command must send through command channel.

Chara$	Function
A or 1	As "B-R" but read from first byte of block
B or 2	As "B-W" but initial buffer pointer to one
C or 3	As "M-E" RAM $500
D or 4	As "M-E" RAM $503
E or 5	As "M-E" RAM $506
F or 6	As "M-E" RAM $509
G or 7	As "M-E" RAM $50C
H or 8	As "M-E" RAM $50F
I+	1541 mode
I-	1540 mode
J or :	Reset vector

Extension "U0" Commands for RF512C only

Format

BASIC	string$="U0"chara$,command$

Syntax These commands must send though command channel.

chara$ bit 7 6 5 4 3 2 1 0	function
$\overline{T}$ E B S 0 0 0 0	1 Read for CP/M
$\overline{T}$ E B S 0 0 1 0	2 Write for CP/M
0 0 0 S 0 1 0 0	Inquire for CP/M header
P I D S 0 1 1 0	3 Format
$\overline{W}$ 0 0 0 1 0 0 0	4 Set-up sector format for CP/M
F 0 0 S 1 0 1 0	5 Query for CP/M sector sequence
G C X 0 1 1 0 0	6 Get/Set status
0 0 0 1 1 1 1 0	7 Utility
Y 0 0 1 1 1 1 1	8 Fastload

S: side
B: transfer to buffer
E: ignore error
T: transfer to computer
D: double side
I: Index Sectoring
P: Partial format
W: Write
F: offset
C: change disk
G: Get status only
Y: file type
X: don't care

(string$) means CHR$ (string#)

NOTE

1,2. Read for CP/M
command$=dest-track$)+(dest-sector$)+(noof-sector$) [+(next-track$)]

3. Format
command$=CHR$(0)=id$ for GCR format
i.e. id$ is a two characters string
command$=(128+logical-sector$)+(sector-skew$)+(sizes$)+(last-track$)+(sector/track$)+(logical-1st-track$)+(offset-track$)+(fill$)+table$ for CP/M format

size	byte/sector
0	128
1	256
2	512
3	1024

4. Set-up sector format for CP/M
   if W=1 command$=(sector-skew$)

5. Query for CP/M sector sequence
   if F=1 then command$=(sector-seq$)

6. Get/set status C for exchange disk
   if G=0 then command$=(new-status$)

7.

command$	function
"S"+CHR$(sector interleve	DOS-sector interleave
"R"+CHR$(retries)	DOS-retries
"T"	ROM test
"M1"	2 MHz clock rate
"M0"	1 MHz clock rate
"H0"	side 0
"H1"	side 1
CHR$ (device)	reset device no.

8. Y=1 sequential file Y=0 program file command$=filename$

echo message		Meanings
For GCR	For MFM	
0.1	128,129	OK
2	130	Sector not found
3	131	Miss Address Mark
4	—	Block not found
5	—	Block checksum error
—	133	Data CRC error
6	134	Format error
7	135	Verify error
8	136	Write protect
9	137	header checksum error
10	—	Data too long
11	—	ID Mismatch
—	139	Disk change
14	142	SYNTAX error
15	143	Device not present

FILE AND DIRECT PROGRAMMING

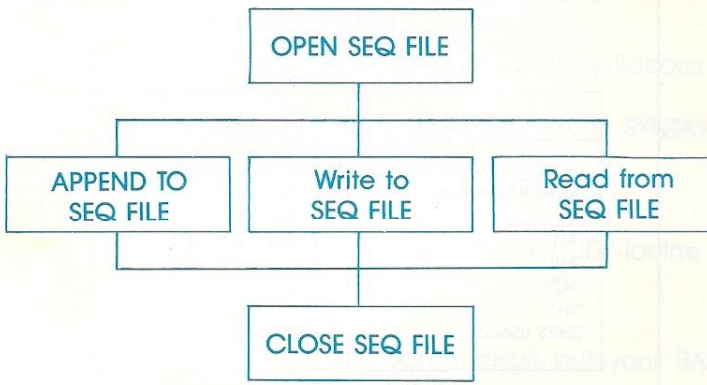
INTRODUCTION

There have two types of data file structure, sequential file and random access file in this disk operating system.

Where sequential file are working with continuous steam of data and it must read/write file from the beginning to end. But random access file can be able to access certain record without waiting for entire file read/write.

Sequential file

Here is the flow chart of how to create and manipulate a sequential file.



Opening a file is the very beginning of file programming; otherwise, data loss, occurance of error or even demage another file will be happened. After a read, write or append action is finished, closing the file to complete such operation is a must.

Other then read, write and append function; in **C-128's BASIC 7.0**, the **CONCAT** command can concatenate two data files together. This provide a way to get rid of large data-base overhard processing time.

This demo program will show you some routines in sequential file programming.

```
10  CLOSE 1:OPEN 1,8,15:PRINT"TYPE END WHEN FINISH"
20  OPEN 2,8,2, "@:SEQ,S,W"
30  INPUT "NAME-"; NAME$:INPUT# 2,NAME$
40  INPUT "TEL-:"; TEL:INPUT# 2, TEL
50  IF NAME$ = "END" THEN 30
60  CLOSE 2:CLOSE 1
110 CLOSE 1:OPEN 1,8,15
120 OPEN 2,8,2,"SEQ,S,R"
130 INPUT# 2,NAME$,TEL
140 PRINT NAME$,TEL
150 IF ST <>64 THEN 120
150 CLOSE 2:CLOSE 1:PRINT"READ FINISH"
```

LINE 10 and 110:	**OPEN** Command channel
LINE 20 and 120:	**OPEN SEQ** file data channel Where @ in line 20 for aviod overwritten error
LINE 30,40:	Input data by user then put into file
LINE 130:	Get data back from file
LINE 140:	Printout data
LINE 50:	Check input finish or not
LINE 150:	Check End of file status
LINE 60 and 150:	**CLOSE** all files

From the program, status variable **ST** is used to get message of file operation. The following table lists the meanings of **ST**.

ST-bit	MEANING
0	Time out on talker.
1	Time out on listener.
2	Cassette data error.
3	Cassette data error.
4	Verify error.
5	Cassette checksum error.
6	**EOF** detected.
7	Device not present.

For detail in file format, please refer to **APPENDIX.**

There has an alternate type of sequential file — USER file. This type of file uses byte as information element and set ST to 64 as EOF.

Here is a simple program in using USER file.

```
  INPUT
10  CLOSE 2:OPEN 2,8,2"USER,U,W"
20  CMD 2
30  GET K$:PRINT K$;
40  IF K$    CHR$(17)THEN 30:REM CTR-Q
50  CLOSE 2

  OUTPUT
10  CLOSE 2:OPEN 2,8,2, "USER,U,R"
20  GET# 2,K$:PRINT K$;
30  IF ST    64 THEN 20
40  CLOSE 1
```

RANDOM ACCESS FILE

It is also called **RELATIVE FILE** in this system.

Programmer can easy build-up the file structure into fields and records without take care of the whole file in program.

The relative file is established by six side-sectors and each side-sector has it data pointer that point to individual data block. According to its format, some **DOS** utlities can't work normally.

The following demo program shows you how to create, wirte and read the relative file.

```
10  DOPEN# 1, "REL,L20"
20  RECORD# 1,30
30  DCLOSE 1
40  FOR I=1 TO 10
50  DOPEN# 1, "REL,L20"
60  INPUT"NAME— ";NA$
70  IF LEN(NAME$)>10 THEN 60
80  INPUT "TEL—:";TEL$
90  IF LEN(TEL$)>10 THEN 80
100 D$=NAME$+CHR$(13)+TEL$
110 RECORD# 1,D$
120 PRINT# 1,D$
130 DCLOSE 1
140 NEXT I
150 FOR I=1 TO 10
160 DOPEN# 1, "REL,20"
170 RECORD# 1, (I),1
180 INPUT# 1,NAME$,TEL$
190 PRINT NAME$; "—", TEL$
200 DCLOSE 1
210 NEXT I
```

LINE 10 to 30: Create **REL** file
LINE 40 to 140: Write ten pairs of data onto it
LINE 150 to 210: Read data back from file

Because user can easily redefine the record length. So, no append function is needs.

DIRECT - BLOCK PROGRAMMING

Other then file operation, user can directly mainpulate the disk itself with very less involute in programming.The way is using the drive RAM as buffer for data processing.

Demo for block read, write and its pointer.

```
10  INPUT "DISK ID",X$
20  CLOSE 1:OPEN 1,8,15
30  CLOSE 2:OPEN 2,8,2, "#"
40  PRINT# 1, "UI:2 0";18;0
50  PRINT# 1, "B-P";2;162
60  GET# 2,I$:GET#2,D$:ID$=I$+D$
70  PRINT"OLD ID IS";ID$
80  PRINT# 1, "B-P";2;162
90  PRINT# 2,X$
100 PRINT# 1, "U2:2 0";18;0
110 CLOSE 2:CLOSE 1
```

LINE 10: Input new disk ID code
LINE 20: Open command channel
LINE 30: Open buffer channel
or one can put a specific buffer number after "#"
LINE 40: Using U1 to substitute block-read on track 18 sector 0
LINE 50: Set block-pointer to 162 (i.e. ID code location)
LINE 60: Get ID
LINE 70: Display the old ID
LINE 80: Again, set pointer to 162
LINE 90: Input new ID into buffer
LINE 100: Using U2 to substitute block-write on track18 sector 0
LINE 110: Close file

Demo for block-assign function

```
10  CLOSE 1:OPEN 1,8,15
15  PRINT# 1, "IC6"
20  PRINT# 1, "B-F:"0;18;1    :REM  free the block
30  PRINT# 1, "B-A:"0;18;1    :REM  allocate such block
40  GOSUB 100                 :REM  OK message
50  PRINT# 1, "B-A:"0;18;1    :REM  allocate it again
60  GOSUB 100                 :REM  no block now
70  END
```

```
100 IMPUT# 1,EN,EM$,ET,ES
110 PRINT EN,EM$,ET,ES
120 RETURN
```

This simple program shows the **BLOCK-FREE** and **BLOCK-ALLOCATE** functions. If a block is been free, it can be allocated without error; otherwise, no block available error willl be appeared

And there has a rarely use **B-E** command. Be sure which buffer will take up the execute block, before you run your machine program.

DIRECT - MEMORY PROGRAMMING

By analogy with **PEEK**, **POKE** and **SYS** in **BASIC**, it is easy to handle **B-R**, **B-W** and **B E** in **DOS**.

```
10  INPUT "TYPE A WORD";K$
20  L=LEN(K$)
30  CLOSE 1:OPEN 1,8,15
40  PRINT# 1"M-W"CHR$(05)CHR$(05)CHR$(L)K$:CLOSE
45  OPEN 1,8,15
50  PRINT# 1, "M-R"CHR$(05)CHR$(05)CHR$(L)
60  FOR I=1 TO L:
70  GET# 1,K$
80  PRINT K$
90  NEXT I:PRINT"—DOES THIS CORRECT?"
100 CLOSE 1
110 OPEN 1,8,15
120 PRINT# 1, "M-E" CHR$ (160)CHR$(234)
130 CLOSE 1
```

In line 40, it use **M-W** as **POKE** to put string K$ into drive's **RAM**. Then, it use **M-R** as **PEEK** to get such string back. Finally, the program use line 120 to execute a drive **SYS — 5472**

APPENDIX A
QUESTIONS AND ANSWERS

Q: How to do if the **LED** never light?
A: Plug the AC adaptor and din cable correctly; make sure that the power is okay. Switch the drive power on.

Q: How to do if the drive active lamp doesn't turn off?
A: Turn off the drive and then disconnet it with the computer. Power on the drive again. If it's all right then follow the manual to do so.

Q: How to do if the drive active **LED** blinks?
A: normally, this is an error action. First of all, disconnect the drive with the computer and power up it again. If it still blinks then some service for the drive is need.

Q: If the drive cannot read a well prepared disk, what's wrong?
A: Turn off your drive and insert the head-protection sheet into it. Then, try to read your disk again.

Q: "THE DEVICE IS NOT PRESENT" message is always shown, why?
A: Check the din cable first, then set the drives selector to 8 (or 9,10,11) to match your program.

Q: The drive doesn't respone to the computer.
A: Check the din cable first and reset the drive by power on it again. Redo your procedure.

Q: Why does the **LED** blink while program is in progress?
A: Type "**PRINT DS**" or **RUN** the error routine to get the reason. Sometimes, there is not an error for the **LED** blinking, unless it blinks continually.

Q: How to avoid error occurance?
A: Please follow these instructions below for convenience in operating.
Don't write on a protected disk.
Don't open an opened file.
Don't forget to close a file when operations are completed.
Don't forget to open data file for block programming.
Don't use channel other than 15 for command communication.
Don't crave too much on the disk to avoid "**NO BLOCK**" error.
Don't operate with any nonexistence file, track or sector.

APPENDIX B
DISK FORMAT

This disk hierarchy

DIRECTORY—HEADER
FILE VECTOR TABLE—PROGRAM FILE
SEQUENTIAL FILE
USER FILE
RELATIVE FILE—SIDE SECTOR
DATA SECTOR

DIRECTORY HEADER (locate at track 18 sector 0)

BYTE	CONTENTS
0	Next track
1	Next sector
2	Usually 65
3	dx128
4t	noof free sector on track t
4t+1	BAM of sector 0 to 7 of track t
4t+2	BAM of sector 8 to 16 of track t
4t+3	BAM of sector 17+23 of track t
144 to 143+L	disk name with length L
144+L to 159	160 if L<16
162	first ID
163	second ID
164	Usually 160
165	Usually 50
166	Usually 65
167 to 170	Usually filled with 160
171 to 179	Usually filled with 0
180 to 191	string "BLOCK FREE."
192 to 220	Usually filled with 0
t'+185	no of free sector on tract t'

Where d is the double side flag (128 for double side; 0 for single side).
t is the track no. from 1 to 35.
t' is the track no. from 36 to 70.

Side one BAM (for RF512C only) (locate at track 53 sector 0)

BYTE	CONTENTS
3t-108	**BAM** of sector 0 to 7 of track t'.
3t-107	**BAM** of sector 8 to 16 of track t'.
3t-106	**BAM** of sector 17 to 23 of track t'.
105 to 255	usually filled with 0.

Where t' is the track no. from 36 to 70. and use 1 in **BAM** to indicate avaible block.

FIVE VECTOR TABLE (LOCATE AT TRACK 18 SECTOR 1)

BYTE	CONTENTS
0	next track
1	next sector
30f+2	128C+64K+type
30f+3	start track of file f
30f+5 to 30f+L+4	file name with length L
30f+L+5 to 30f+20	160 if L<16
30f+21	relative file first side sector's track
30f+22	relative file first side sector's sector
30f+23	relative file's record length
30f+24 to 30f+27	unused
30f+28	start track of replacement @ file
30f+29	start sector of replacement @ file
30f+30	low-byte of no. of blocks in file f
30f+31	high-byte of no. of blocks in file f

Where f is the file entry number
C is the file closed flag (1 for close)
K is the locked file flag (1 for locked)

type	file
0	Deleted
1	Sequential
2	Program
3	User
4	Relative

PROGRAM FILE

BYTE	CONTENTS
0	next track, or zero indicate tail block
1	next sector
2	low-byte of load address in start block and program data in others
3	high-byte of load address in start block and program data in others
4 to 255	program data and use three zero for **EOF**

SEQUENTIAL FILE

BYTE	CONTENTS
0	next track or zero indicate tail block
1	next sector
2 to 255	file data and use 13 as terminator

RELATIVE FILE
SIDE SECTOR

BYTE	CONTENTS
0	next track
1	next sector
2	side sector no
3	record length
2s+4	track of s th side sector
2s+5	sector of s th side sector
2b+14	track of b th data sector
2b+15	sector of b th data sector

Where s means side sector; range from 0 to 5
b means data block; range from 1 to 120

DATA SECTOR

BYTE	CONTENTS
0	next track
1	next sector
2 to 255	use 0 to fill up all record and place 255 in the first byte of an empty records

BOOT SECTOR (locate at track 1 to sector 0)

BYTE	CONTENTS
0	43
1	42
3 to 6	boot pointer
7 to 255	Booting message, name of program file; where use zero as separator.

AUTO FILE (USER TYPE FILE)

BYTE	CONTENTS
0	low-byte loading address
1	high-byte loading address
2	L
3 to 3+L	auto assembly program
4+L	checksum

Where checksum is the sum of value of byte (from 0 to 4+ L with carry.

APPENDIX C

SETTING THE DEVICE NUMBER

Each periphral in Commodore Home Computer System has its own device number. The device number can be assigned from 1 to 31 and the default value for the drive is 8 (this is preset by factory).

There provide two ways to change device number, one is by programming, the another by switch the DIP selector (on back pannel). Normally, this is not necessary to do so.

Software method:
Method I:

OFF all drives first, then power up the drive which will be changed device number. And RUN the following routine.

```
10 CLOSE 1:OPEN 1,8,15
20 INPUT "NEW DEVICE NO.-";N:=N=N+20
30 PRINT# 1, "M-W"CHR$(120)CHR$(0)CHR$(2)CHR$(N)CHR$(N)
40 CLOSE 1
```

After this program has been done, power up the another drive.

Method II:

Again, power on one drive only, then **RUN** the following routine.

```
10 INPUT"NEW DEVICE NO.— ";N
20 CLOSE 1
30 OPEN 1,8,15 "U0>";CHR$(N):CLOSE 1
```

After this program has been done, power up the another drive.

Hardware method:

On the back pannel, there has **DRIVE SELECTOR** micro-switch. Set the switch as the table shown before power up the drive. The the drive's device number setting is ready.

DRIVE SELECTOR				device no.
4	3	2	1	
ON	ON	ON	ON	8
OFF	OFF	ON	ON	9
ON	ON	OFF	OFF	10
OFF	OFF	OFF	OFF	11

APPENDIX D

ERROR MESSAGE

How to read the error:

The easiest way to get drive error code is: **PRINT DS** or simple type ? **DS**<Return>

If you want to get more information about the error, it is recommanded to use the following routine:

```
10 CLOSE 1
20 OPEN 1,8,15
30 INPUT# 1,EN,EM$,ET,ES
40 CLOSE 1
50 PRINT EN,EM$,ET,ES
```

Where **EM** is the error code.
EM$ is the error message.
ET is the error track.
ES is the error sector.

Error code	Error Message	
0	OK	No error occur.
1	FILES SCRATCHED	An echo message for SCRATCH command.
20	READ ERROR	Block header not found.
21	READ ERROR	Sync mark absent.
22	READ ERROR	Improper data block.
23	READ ERROR	Checksum error.
24	READ ERROR	Decoding error
25	WRITE ERROR	Verify error.
26	WRITE PROTECT ON	But you try to write.
27	READ ERROR	Header error.
28	WRITE ERROR	Sync mark time out.
29	DISK ID MISMATCH	Foreign disk insert.
30	SYNTAX ERROR	DOS can't interpret command.
31	SYNTAX ERROR	Can't execute command.
32	SYNTAX ERROR	Command length too long.
33	SYNTAX ERROR	Invalid filename.
34	SYNTAX ERROR	Miss file name. Usually, ":" is omitted.
39	SYNTAX ERROR	Miss autoboot file.
50	RECORD NOT PRESENT	Miss data record.
51	OVERFLOW IN RECORD	Data length too long.
52	FILE TOO LARGE	No more room for relative record.
60	WIRE FILE OPEN	Open an un-close file.
61	FILE NOT OEPN	Try to attempt an un-open file.
62	FILE NOT FOUND	File doesn't exist.
63	FILE EXISTS	Try to create an already exists file.
64	FILE TYPE MISMATCH	Improper file operationg.
65	NO BLOCK	Try to allocate an already occupied block.
66	ILLEGAL TRACK OR SECTOR	Nonexistence track or sector.
67	ILLEGAL TRACK OR SECTOR	Sector linker point to an nonexisting block.
70	NO CHANNEL	No more channel onhand
71	DIR ERROR	BAM mismatch between disk and drive memory.
72	DISK FULL	Only has less than three block free.
73	(dos version)	DOS mismatch.
74	DRIVE NOT READY	Disk unformatted.

APPENDIX E
SPECIFICATIONS

Power Supply: PW 916 — AC/AC adaptor 36W
Input: 120 VAC 60Hz or
220 VAC 50Hz
Output: 9VAC 1.5Amp
16VAC 0.8Amp

Storage:
GCR: 683-block/side
256-byte/block
35-track/side
MFM: 130K-byte/side; 128-byte/sector
160K-byte/side; 256-byte/sector
180K-byte/side; 512-byte/sector
200K-byte/side; 1K-byte/sector
40-track/side
(For RF512C only)

Operating clock rate: 1MHz (RF501C, RF502C)
2MHz (RF512C)

Interface: 6-pin Din serial port
with daisy-chain dual port.

Magnetic head: One 48 TPI head (RF501C)
Two 48 TPI heads (RF502C, RF512C)

Internal memory: 16K-byte ROM/2K-byte RAM (RF501C, RF502C)
32K-byte ROM/2K-byte RAM (RF512C)

Detection: Write protect detection
Outermost track detection
Hardsector index detection

Programming language: 6500 assembly language

Indication: Red LED for power on
Green LED for active

APPENDIX F

RADIO AND TELEVISION INTERFERENCE

The equipment described in this manual generates and uses radio frequency energy. If it is not installed and used properly, that is in strict accordance with our instructions, it may cause interference to radio and television reception.

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

You can determine whether your computer is causing interference by turning it off. If the interference stops, it was probably caused by the computer. If your computer does cause interference to radio or television reception, you can try to correct the interference by using one or more of the following measures:

— Turn the TV or radio antenna until the interference stops.
— Move the computer to one side or the other of the TV or radio.
— Move the computer farther away from the TV or radio.
— Plug the computer into an outlet that is on a different circuit from the TV or radio. (That is, make certain the computer and the TV or radio are on circuits controlled by different circuit breakers or fuses.)

If necessary, you should consult your dealer or an experienced radio/television technician for additional suggestions. You may find the following booklet prepared by the Federal Communications Commission helpful:

"How to Identify and Resolve Radio-TV Interference Problems"

This booklet is available from the U.S. Government Printing Office, Washington, DC 20402, Stock number 004-000-00345-4.

- First published 1987
- Printed in Hong Kong.