COMX DOS MANUAL

VERSION 1.3

BY : TSANG CHING SHUEN
CHARLES WONG

DATE : MARCH 20, 1985

COMX WORLD OPERATIONS LTD
R & D DEPARTMENT

APPROVED BY : EDMOND LEUNG

REV.0

TABLE OF CONTENT

Chapter 1

INTRODUCTION

The COMX DOS is a Disk Operating System for utilizing the COMX computers. It is well-suited for a wide range of applications on the COMX computers of almost any size or complexity. Its main features are :

1. Automatically keep track of files.

2. Save and retrieve information.

3. Do a multitude of housekeeping tasks.

4. Utilizes 5-1/4 inch, high-density floppy disk drivers.

5. Can use different types of disk drive, including single side and single track density, double side and single track density, single side and double track density.

6. Provide a maximum of 138 kilobytes of on-line mass memory storage.

7. Can be accessed in high speed that reduce the time of loading and saving of files to a few seconds.

This manual is intended to provide the information necessary to install, maintain, and write BASIC language software with the COMX DOS. It also describes in detail the software features, commands and the structures of the system. It assumes that the reader is already familiar with the COMX BASIC language.

Chapter 2

INSTALLATION

2.1 Equipments needed for the COMX DOS :

2.1.1 a COMX computer

2.1.2 a television set or a monitor

2.1.3 a COMX disk drive

2.1.4 a COMX disk controller card

2.1.5 a system master disk

2.2 Installation :

2.2.1 Without the Expansion Box of COMX

1. Turn off the power switch at the back of the COMX.

2. Plug the disk controller card into the COMX expansion slot which is on the right hand side of the COMX.

3. Turn on the COMX.

4. Turn on the power supply of the disk drive.

5. Put the master disk into the disk drive.

6. Type "DOS CAT", there will be a system message and a catalog of files shown on the screen .

2.2.2 With the Expansion Box of COMX

1. Turn off the power switch at the back of the COMX.

2. Plug the Expansion Box into COMX expansion slot which is on the right hand side of the COMX.

3. Plug the disk controller card in any slot of the Expansion Box.

4. Turn on the COMX.

5. Turn on the power supply of the disk drive.

6. Put the master disk into the disk drive.

7. Type "DOS CAT", there will be a system message and a catalog of files shown on the screen.

Note : Remember to turn off the power switch at the back of the COMX before inserting or removing the card. This is important to prevent any damage to the hardware. If the power is on, the removal or insertion of any card could cause permanent damage to both the card and the COMX.

2.3 Using the equipment correctly :

2.3.1 Dual Disk Drives :

The COMX disk controller card can support dual disk drives. The installation of dual disk drives is the same as a single disk drive. Since the system bootstrap process is default on drive 1, a diskette with the bootstrap program must be put into drive 1 no matter which drive you want to access first.

2.3.2 Power supply :

There are two LED on the controller card for indicating the status of the disk drive. If using with the Expansion Box, the lower LED will not be on before you access the disk drive. If there is no Expansion Box, the lower LED will be on after the power supply is turned on. When the disk drive is accessing, the upper LED will be on too.

2.3.3 Disk Drive and Diskette Handling

The COMX DISK DRIVE is a mechanical device, with motors and moving parts. It is somewhat more delicate than the computer, so handle it with care. Rough handling, such as dropping the drive, can cause it to malfuction.

The diskettes used by COMX DOS are single-density, single/double sided type or double density, single sided type. A diskette can store over 138,000 bytes. Each diskette is a small plastic magnetic coated medium so that information may be stored on and erased from its surface. The coating is similar to the magnetic coating on recording tape. The diskette is permanently sealed in a square black plastic jacket which protects it, helps keep it clean and allows it to spin freely.

To assure trouble-free reading and writing files, the diskettes must be handled and stored with care. To avoid damage to the recording surface and to prevent diskette deformation, the following precautions should be carefully observed.

1. Close the disk guard cover when not in use.

2. Never let anything touch the brown or gray surface of the diskette itself. Handle the diskette by the black plastic cover only.

3. Do not clean the recording surface.

4. Do not bend the diskette or deform it with paper clips or other similiar mechanical devices.

5. To write on a diskette label, use a felt tip pen. Do not press hard. It is best not to write on a label attached to a diskette, but to write on the separate label, then attach it to the diskette.

The operating and storage environment must be compatible with the materials of the diskette. The environment of the diskette should meet the following criteria :

1. No noticeable dirt, dust, or chemical fumes in the immediate area.

2. Temperature between 50 F (10 C) & 115 F (45 C).

3. Relative humidity between 8 and 80 percent.

4. Maximum wet-bulb temperature of 85 F (30 C).

5. No direct sunlight on diskette surface for prolonged periods.

6. No nearby magnetic fields. This means that we should keep them away from electric motors and magnets; they should not be placed on top of electronic devices such as television sets.

2.3.4 Care of the COMX DISK DRIVE and diskette :

Each diskette is a small coated plastic so that information may be stored on and erased from its surface. The coating is similar to the magnetic coating on recording tape. The diskette is permantly sealed in a square black plastic cover which protects it, helps keep it clean and allows it to spin freely. While the diskette are somewhat flexible, actual bending of the diskette can damage it.

Chapter 3

DOS STRUCTURE

.1 Volume Table of Contents

Sector $Ø of track $Ø contains the diskette's Volume Table of Contents, or VTOC. The VTOC stores the following information :

3.1.1 Volume Table of Contents (VTOC)

Track $Ø, Sector $Ø

Byte (Hex)	Description
Ø-7	Volume name
8-F	Volume date
1Ø	Total no. of directory entries
11	Total no. of free blooks used
12	Single or double track density
13	Single or double side
14-59	Block map
5A-69	Directory map

3.1.2 Volume Extension

Track $Ø, Sector $1

Byte (Hex)	Description
Ø-45	Block map extension

2 The Disk Directory

The disk directory information is stored on track Ø sector 2 to 14, each contains 4 directory entries.

3.2.1 Directory Map Format

The following directory map indicates the number of the directory entries in each sector of track Ø. One byte maps to one sector correspondingly.

Example :

Byte Value (Bin) $5A-$69	Description
ØØØØØØØØ	No directory entries
ØØØØØØØ1	Contains 1 directory entry
ØØØØ1111	All 4 directory entries fully use

3.2.2 Directory Entry Format (size : 32 bytes)

Byte (Hex)	Description
Ø	Directory entry number
1-13	File name
14	File attribute
15	File Type
16	Record length (Ø if no fixed length)
17	First physical block number
18-1A	File size
1B-1C	File starting address (default 44ØØ)

3.2.3 File Attribute

Byte Value (Bin)	Description
ØØØØØØØ1	File write protected
ØØØØØØ1Ø	File delete protected

3.2.4 File Type

Byte Value (Bin)	Description
ØØØØØØØ1	Basic program file
ØØØØØØ1Ø	Assembler program file
ØØØØØ1ØØ	Text file
ØØØØ1ØØØ	Binary file
ØØØ1ØØØØ	Assembler and Basic program file
ØØ1ØØØØØ	Forth program file

.3 File Descriptor Format (size : 42)

Start from $BC00

Byte (Hex)	Description
0	User return file number
1	Drive information
2	Directory entry number
3-15	File name
16	File attribute
17	File type
18	Record length
19	First block number
1A-1C	Current read logical byte address
1D-1F	Current write logical byte address
20-21	Current read physical block & sector number
22-23	Current write physical block & sector number
24-26	File size
27-28	File buffer location
29	Starting address (high byte only)

3.3.1 Drive Information

Bit 4 is always on

Byte Value (Bin)	Description
00010100	Drive 0 is used
00011000	Drive 1 is used
00010000	Side 1 is used
00110000	Side 2 is used

3.3.2 File Buffer Status

Byte Value (Bin)	Description
00000001	Sector in buffer
00000010	Sector modified

3.4 System Parameter 1

Start from $BE50

Byte (Hex)	Description
0-9	System registers save location
A	Current descriptor number
B	Number of descriptors created (maximum 6)
C	End of file
D-F	$ABCDEF for has found step rate
10	Language which call the DOS 0 for Basic, 1 for Assembler
11	Read status 0 for read, 1 for read alternate, 2 for read record
12	First dimension value (1 dimensional)
13	Second dimension value (2 dimensional)
14	First dimension value (for user peek)
15	Second dimension value (for user peek)
16	File buffer status
17	Current drive number used
18	Boot flag
19-1B	External DOS flag, $ABCDEF means use of external RAM
1C-1D	External DOS entry point
1E	Store up the stack location
1F	No of external jump table entries
20	Store up the status after R/W sector

5 System Parameter 2

Start from $BF00

Byte (Hex)	Description
0	Number of directory entries used in drive 1
1	Number of blocks occupied in drive 1
2	Single or double track density of drive 1
3	Single or double side of drive 1
4-6	Boot test of drive 1
7	Driver on or of after used (default on)
8	Logical error number
9	Number of directory entries used in drive 2
A	Number of blocks occupied in drive 2
B	Single or double track density of drive 2
C	Single or double side of drive 2
D-F	Boot test of drive 2
10-20	String's a name and array number pointed by file description pointer.

3.5.1 Boot Test

Byte value (hex)	Description
789ABC	Disk has been initialised, but not booted
123456	Disk has been both booted and initialised

Chapter 4

FILE STRUCTURE

4.1 File Structure

In our COMX Disk Operating System, information is recorded on a diskette with 16 sectors per track. And 128 data bytes is stored in each sector. Details of the disk structure are found in the appendix.

All files information are stored from sector 2 to sector 14 of track 0. There are in total 13 sectors. Each file requires 32 bytes to hold their information, so one sector holds four files and a maximum of 52 files can be created. It can be summarized that track 0 is used by the system, and it has the following usage :

Area	Usage
Track 0, sector 0	Volume table and block map
Track 0, sector 1	Block map extension
Track 0, sector 2 to 14	Directory entry format
Track 0, sector 15	The handler

Starting from track 2, real content of the files are stored. The directory entry format is discussed below.

In the previous paragraph, each file needs 32 bytes to hold their information. Each byte has it's special usage as listed

Byte Number	Usage
1	Directory entry number
2 - 20	File name (19 character)
21	File attribute
22	File type
23	Fixed record length
24	First physical block number
25 - 27	File size
28 - 29	File starting address
30 - 32	No use

4.1.1 Directory Entry Number

These are the numbers given to each file as their order in the disk which ranged from 1 to 52 in an ascending order. These numbers can be seen in the left most column in the catalog.

4.1.2 File Name

The filename of the files are stored in this area, a maximum number of 18 characters can be used.

4.1.3 File Attribute

This marks the software write protective status of each file :

%00000001	write protect
%00000010	delete protect
%00000011	both

They are found in the rightmost column of the catalog :

D - delete protect
W - write protect
D,W - delete and write protect

4.1.4 File Type

Our DOS can provide storage for six types of files, details of which will be discussed later. Firstly, how to distinguish them from each other in the Directory entry format will be described.

%00000001	basic file	(BAS)
%00000010	assembler file	(ASM)
%00000100	text file	(TEX)
%00001000	binary file	(BIN)
%00010000	assembler and basic file	(A&B)
%00100000	forth file	(FTH)

Their abbreviations - as in the brackets, can be looked up under the column title 'TYPE' of the catalog.

4.1.5 Record length

This specifies the fixed record length if it is a random access text file. If it is not a text file or just a sequential text file, a zero will be filled in this byte. However, it cannot be seen in the catalog.

4.1.6 First Physical Block Number

The first track number and sector number storing the file can be calculated from this information (Note : one block consists of eight sectors. There are two blocks in one track.)

4.1.7 File size

Three bytes are occupied to store the size of the file. They are in byte-count instead of block-count. After changing to decimal and rounding up to the nearest kilobyte, they are retrieved under the heading 'SIZE' of the catalog.

Note : During the changing procedure, the bytes will be rounded up to get an integer result. For example, file size ranging from 0 to 1024 will be rounded up to 1K bytes.

4.1.8 File Starting Address

Two bytes are occupied for storing this address - the high byte and the low byte. The default value is $4400 for basic file, basic & assembly file, and forth file. The value is user defined in assembly and binary. On the other hand, there is no starting address for text file since it is not a program itself, only data and information are stored.

4.2 Text File

Sometimes floppy disks are used to store information that is not a program . For example, a telephone directory, a price list, or a mailing list. A text file, sometimes called a date file allows the user to do this and more. The word "TEX" will appear in the column "Type" of the catalog if that is a text file.

The DOS commands like LOAD, SAVE, and RUN may not be used with text files. An attempt to do this will cause a SYSTEM CRASH. It is because the above commands are assumed to deal with other files which will be discussed later. Text files have no starting and ending addresses, treating them as a program causes the computer malfunction.

Date and information are read from and write to a text file by means of several DOS command, they will be discussed in the next chapter :

CREATE
OPEN
READ
WRITE
RDALT
WRALT
PUTRD
PURWR
APPF
CLOSE

Unlike other DOSs, COMX DOS commands for text files can be used in the immediate-execution mode, they need not necesarily be executed within a program.

Furthermore, other DOS commands :

DEL
REN
CAT

and our utilities :

COPY
LOCK
UNLOCK

Work with the text files in the same way as other program files. Our utility, MERGE, works only on text files.

4.3 Sequential and Random Access Text Files

Text files are seperated into sequential and random access type. Both types of text files store strings of ASCII codes to represent data, but in different formats. The file consists of data or strings seperated by RETURNS. The RETURN is sent at the end of every string variable. Note that it is a character rather than an action that it usually represents, its ASCII code is OD in hexidecimal.

Each item of data, ending with its RETURN character, is called a field. A field is stored in the text file as a series of characters represented by their ASCII codes.

The concepts of the two types of text files will be discussed below and their usage, format and examples will be shown in the next chapter.

4.3.1 Sequential Files

A rough diagram of sequential file is shown below, the symbol ' represents the RETURN character.

CHARACTER	T	H	I	S	'	I	S	'	C	O	M	X	'	3	5	'
ASCII	54	48	49	53	ØD	49	53	ØD	43	4F	4D	58	ØD	33	35	ØD
FILE BYTE	Ø	1	2	3	4	5	6	7	8	9	1Ø	11	12	13	14	15
FIELD			Ø			1			2					3		

From the above diagram, it is shown that each field is stored immediately after the RETURN character of the preceding field. When stored on the diskette, these fields may be of different lengths (the word THIS takes 5 bytes including the RETURN.) A sequential text file is stored on the diskette as one continuous series of ASCII-coded characters, i.e. a chain of fields with no gaps left between them.

4.3.2 Random Access Text Files

The following diagram shows how the structure of random access file with a fixed record length of 5 deals with its data.

```
CHARACTER  T  H  I  S     I  S           C  O  M  X     3  5
ASSCII    54 48 49 53 00 49 53 00 00 00 43 4F 4D 58 00 33 35 00 00 00
FILE BYTE  0  1  2  3  4 !5  6  7  8  9 !10 11 12 13 14!15 16 17 18 19
FIELD            0       !       1      !       2     !       3
```

Each random access file, when created, must be given a fixed record length. This record length decides the length of data.

As in the above diagram, each record equally contained 5 bytes no matter they are empty or not.

Unlike sequential files, the records in a random-access file are of specified FIXED LENGTH. (Every time the user write data to it, a fixed defined space is set aside on the diskette for a complete, standard length record, whether or not the record is all filled with ASCII codes.) Also, there is no need for random access files to have $0d to seperate records in between. Random access files is not an efficient way to use the diskette spaces. However, since these files are set up in such a regular fashion, it is fast and easy to retrieve or modify information from any part of the file. Random access files should be used to deal with data that require a fast access or with data that are updated fairly often.

4.3.3 Create and Retrieve Information from Two Kinds of Files

Ramdom-access files are created and retrieved in a manner very much similar to that of sequential files. The main difference is that certain commands require additional parameters : CREATE requires R(x) at the end where x represents the fixed record length, while commands PUTRD and PUTWR in use with the random access files determine the pointers to reach the desired record by these fixed record length.

Formats and sample programs will be presented and discussed in details in the following chapter of THE USAGE OF THE DOS COMMANDS.

Note 1 : In general, fixed record length do not apply to numeric arrays. Text files that bear ONLY numeric arrays are all default to be random access files of a fixed record length of FOUR. It is because every numeric variable needs four bytes to hold them (see diagram below). When creating a random access file that is used to hold both string varible and numeric variable, fixed the record length GREATER THAN OR EQUAL TO FOUR. Writing a record of ten characters into a random access file with a fixed record length of eight will cause the last two characters to be CUT AWAY.

The following diagram is to show the structure of a text file which stores numeric variables of 1,2 & 3. It is a random access file with a fixed record length of 4.

```
-----------------------------------
!00!00!00!01!00!00!00!02!00!00!00!03!
-----------------------------------

  record 1  !  record 2 !  record 3 !
```

Note 2 : Numeric array - only carries numeric numbers. e.g 1,2,3,4.

String array - carries alphanumeric character e.g. A,B,A1,Z8.

Chapter 5

DOS COMMAND

This chapter describes in details each system command available on the COMX DOS. For ease of use, the system command descriptions are given in a standard format which includes the command name, its purpose, its format, its action, and examples.

In the description for each command, the square brackets [and] indicate optional input. The diskette is assumed to be in drive 1, so that the command names do not have to be followed by specific drive numbers.

5.1 DOS CAT

(A) Purpose :

Display on the screen the number of free blocks, the disk name and the creation date. A list of all files on the disk presented next.

(B) Format :

DOS CAT [/DR]

When each file is displayed, an abbreviation of its file type and the number of blocks used are shown for each file. These file types are :

BAS	BASIC program file
ASM	Assembler program file
A&B	Assembler and BASIC program file
FTH	Forth program file
BIN	Binary file
TEX	Text file

If the file is protected, it will be displayed with an indicator of its protecton type. The protection types are :

W Write protected, the file cannot be written over.

D Delete protected, the file cannot be deleted.

(C) Example :

Put the master disk into drive 1 and type "DOS CAT"

DOS CAT

A list of information on the disk will be shown on the screen :

```
    DISK NAME   : MASTER       05-11-84
    FREE BLOCK  : 099          DBL SIDES

FILE NAME            TYPE SIZE
---------            ---------

01 : BOOT            ASM   001   D,W
02 : INIT            ASM   002   D,W
03 : DEMO            BAS   026   D,W
04 : LOCK            ASM   001   D,W
05 : UNLOCK          ASM   001   D,W
06 : COPY            ASM   001   D,W
07 : HELP            ASM   002   D,W
08 : MERGE           ASM   005   D,W
```

5.2 DOS SAVE

(A) Purpose :

If there is no such file by the file name specified on the diskette in the specified or default drive, a file will be created on that diskette and the current BASIC program will be stored under the given file name if no other option after the file name other than the drive number is chosen.

(B) Format :

DOS SAVE,<" F.N. "> [,F] [,@SADR] [,@EADR] [,B] [/DR]

If the chosen disk already contains a file with the specified file name, the DOS will allow you overwriting it depending on your answer to "Y" or "N".

(C) Examples :

To save a BASIC program :

Format : DOS SAVE,<" F.N. "> [/DR]

Example : DOS SAVE, "NEW PROGRAM"

To save an assembler program :

Format : DOS SAVE,<" F.N. ">,@SADR,@EADR [/DR]

Where SADR and EADR are the program starting & ending address respectively.

Example : DOS SAVE,<"GAME">,@4400,@6000

To save a FORTH program :

Format : DOS SAVE,<" F.N. ">,F [/DR]

Example : DOS SAVE,"SCREEN 1",F

To save programs written in assembler and BASIC.

Format : DOS SAVE,<" F.N. ">,@SADR [/DR]
Where SADR is the starting address of the assembler program.

Example : DOS SAVE,"SOURCE",@5000

Note : The starting address for the assembler program must be smaller than that for the BASIC program.

Save a binary file :

Format : DOS SAVE,<" F.N. ">,@SADR,@EADR,B [/DR]
Where SADR is the starting address of the program, and EADR is the ending address of the program.

Example : DOS SAVE,"FILE.OBJ",@5000,@9000,B

5.3 DOS LOAD

(A) Purpose :

Load the program file with the specified name on the disk in the specified or default drive.

(B) Format :

DOS LOAD,<" D.N. "> [/DR]

NOTE : DON'T USE THIS COMMAND ON TEXT FILES.

(C) Example :

DOS LOAD,"DEMO"

5.4 DOS RUN

(A) Purpose :

Load the BASIC program file with the specified name on the disk in the specified or default drive, then run the loaded program. "DOS RUN" can also be used to execute programs but it will only run the BASIC part of that program.

(B) Format :

DOS RUN,<" F.N. ">

(C) Example :

DOS RUN,"COMX"

5.5 DOS DEL

(A) Purpose :

Removes the file with the specified name if the file is not delete protected.

(B) Format :

DOS DEL,<" F.N. "> [/DR]

(C) Example :

DOS DEL,"COPY"

A SCENARIO : DOS CAT, DOS SAVE, DOS LOAD,DOS RUN,AND DOS DEL

The following is a table as it might appear on the screen of your COMX. Suppose the system master disk is in your disk drive 1. Types in DOS CAT.

```
:DOS CAT

   DISK NAME   : MASTER      Ø5-11-84
   FREE BLOCK  : Ø99         DBL SIDES

FILE NAME                TYPE SIZE
---------                ---------

Ø1 : BOOT                ASM   ØØ1   D,W
Ø2 : INIT                ASM   ØØ2   D,W
Ø3 : DEMO                BAS   Ø26   D,W
Ø4 : LOCK                ASM   ØØ1   D,W
Ø5 : UNLOCK              ASM   ØØ1   D,W
Ø6 : COPY                ASM   ØØ1   D,W
Ø7 : HELP                ASM   ØØ2   D,W
Ø8 : MERGE               ASM   Ø05   D,W

:NEW       (Erases the program in memory)
:1Ø FOR 1=1 TO 1Ø
:2Ø PRINT I,
:3Ø NEXT I
:DOS SAVE,"TEST"         (Save the program and give it a name
                         'TEST')

:DOS CAT

    DISK NAME   : MASTER       Ø5-11-84
    FREE BLOCK  : Ø98          DBL SIDES

FILE NAME                TYPE SIZE
---------                ---------

Ø1 : BOOT                ASM   ØØ1   D,W
Ø2 : INIT                ASM   ØØ2   D,W
Ø3 : DEMO                BAS   Ø26   D,W
Ø4 : LOCK                ASM   ØØ1   D,W
Ø5 : UNLOCK              ASM   ØØ1   D,W
Ø6 : COPY                ASM   ØØ1   D,W
Ø7 : HELP                ASM   ØØ2   D,W
Ø8 : MERGE               ASM   Ø05   D,W
Ø9 : TEST                BAS   ØØ1   D,W
```

(The file 'TEST' is in the directory)


```
:NEW       (Erases the program in memory)
:DOS LOAD,"TEST"    (Load the program called 'TEST' to the
                    memory)
:LIST
1Ø FOR 1=1 TO 1Ø
2Ø PRINT I,
3Ø NEXT I

:NEW
:DOS RUN,"TEST"
1         2         3         4         5
6         7         8         9         1Ø

:DOS DEL,"TEST"
:DOS CAT

   DISK NAME   : MASTER           Ø5-11-84
   FREE BLOCK  : Ø99              DEL SIDES

FILE NAME                TYPE SIZE
---------                ---------

Ø1 : BOOT                ASM   ØØ1   D,W
Ø2 : INIT                ASM   ØØ2   D,W
Ø3 : DEMO                BAS   Ø26   D,W
Ø4 : LOCK                ASM   ØØ1   D,W
Ø5 : UNLOCK              ASM   ØØ1   D,W
Ø6 : COPY                ASM   ØØ1   D,W
Ø7 : HELP                ASM   ØØ2   D,W
Ø8 : MERGE               ASM   Ø05   D,W
```

5.6 DOS URUN

(A) Purpose :

Load the ASSEMBLER program file with the specified name on the disk in the specified or default drive, then run the loaded program.

(B) Format :

```
DOS URUN,<"F.N.">
```

(C) Example : DOS URUN,"INIT"

5.7 DOS REN

(A) Purpose :

Finds the specified file name on the disk and changes its name. The file's contents will not be affected.

(B) Format :

```
DOS REN,<" OLD F.N. ">,<" NEW F.N. "> [/DR]
```

(C) Example :

```
DOS REN,"DEMO","DEMONSTRATION"
```

5.8 DOS NEW

(A) Purpose :

This command is used for initialising the disk system. It is used whenever a file is being opened after the system reset. We also have to use DOS NEW to re-init the DOS system if there is an error message after using commands DOS RUN, DOS SAVE, or DOS LOAD.

(B) Format :

```
DOS NEW
```

(C) Example :

```
DOS NEW
```

5.9 Text File Commands

5.9.1 (A) Creating a text file

A text file needs to be created to manage records, data, and information.

(B) Format : Dos create,*wn,"filename"[,/R(x)][/2]

Work number (wn) is used to identify the text file, operations on files like read, write and close depend on the work number given. Six text files can be created at the same time (work number 1-6). So, inconvenience of typing the filename each time is avoided.

The option [/2] is used if the text file is created on drive 2 of the dual drive. This option is not applicable to a single-drive user. The option [,/R(x)] is used for random-access files, where x is the fixed record length. It needs to be specified only in the DOS CREATE command. Re-opening of the files do not require such a command.

The command CREATE also do the job of OPEN (which will be discussed later), so CREATE actually create the text file and open it.

(C) Example : To create two text files, one sequential file named COMX on drive 1, one random-access file named COMX 35 with fixed record length of 3 on drive 2.

Format :
```
Dos create,*1,"COMX"
Dos create,*2,"COMX 35",/R(3)/2
```

Note : Access of these two files are identified by their work number. ie. 1 for COMX and 2 for COMX 35.

5.9.2 Closing the file

Every text file have to be closed after accessing. No data can read from and write to a text file before opening and after closing. As mentioned earlier, you should CLOSE the file with it's work number.
Format : DOS CLOSE,*wn

Example : To close the two files created above.

Format :
```
DOS CLOSE,*1
DOS CLOSE,*2
```

Note : Files opened must be closed afterward. Failure to CLOSE a file that was opened and written to by a write command may result in loss of data.

5.9.3 (iii) Opening a file

(A) Text files have to be opened before anything can be written to or read from it. Open the file with a work number (between 1 to 6). It does not need to be the same work number as in creation.

(B) Format : DOS OPEN,*wn,"FILENAME"[/2]

The object file must exist in the directory. Use CREATE if it is a new file. The OPEN set aside a buffer in the memory to handle the file's input and output, the system is also ready to read and write from the BEGINNING of the file. On the other hand, CLOSE released the file's buffer.

(C) Example : To open the two files

```
Format  : DOS OPEN,*1,"COMX"
          DOS OPEN,*2,"COMX 35"/2
```

5.9.4 Writing a Files

(A) Record and data can only be written into a text file instead of saving it as other files do. Before writing a file, it must be opened. The file also has to be closed after it has been accessed. You must use the same work numbers for WRITE and CLOSE as the ones used during OPEN.

(B) Format : DOS WRITE,*wn,A$(x),B$(x,y),C(x),
D(x,y)....

Records have to be written into the text files in the form of arrays, no matter it is a string array or a variable array (numeric array). A maximum of 8 arrays can be written at the same WRITE command, and the size of each array depends on user memory. The usage of single-dimensional or two-dimesional arrays is the same. For example, two-dimensional array of A$(2,255) write the same size as two single dimensional array of A$ (255) and B$(255). The x and y value can be a number or a variable.

For random-access files, there is no need to specified it's fixed record length again. After opening a text file which is random-access file type, records being written into that file will have a fixed length.

Note : If you write a string "WWWWW" into a random-access file with a fixed record length of 3; only "WWW" can actually be written into the file.

Caution : Don't write an empty array ! If A$(1) have not been defined and it is written into the text file, it may damage the file, or even the diskette. If the COMX computers is carelessly reset during WRITING, the array(s) of the WRITE command will be destroyed and part of the array(s) will be written into the file.

The following is an example in the BASIC language to demonstrate WRITE files. The purpose is to manage five person's NAME, AGE, and JOB using a sequential file.

(C) Example :

```
10 REM ** DEMO PROGRAM **
15 DIM B(5)
20 FOR I = 1 TO 5
30 INPUT "NAME "A$(I)
40 INPUT "AGE "B$(I)
50 INPUT "JOB "C$(I)
60 NEXT I
70 DOS CREATE,*,"DEMO"
80 DOS WRITE,*1,A$(5),B(5),C$(5)
90 DOS CLOSE,*1
```

```
line 20 to 60 get the data
line 70 creates the file named "DEMO"
line 80 writes the records into the file
line 90 closes the file
```

5.9.5 WRALT

(A) WRALT (WRITE ALTERNATE) is one of the DOS command similar to WRITE. The format of them is exactly the same.

(B) Format : DOS WRALT,*1,A$(x),B$(y)....

Write alternate means write the arrays into the text file alternately. Write the first record in the first array, then the first record in the second array and so on. Until all the first record of each array are written, the second ones will be handled in the same manner. If the WRITE command is used instead of WRALT as in the example above, all five records of the A$ have to be written into the file before dealing with the second array.

Note : The value of x should be the same as y while using WRALT.

Caution : In using WRALT, the system only takes the array size of the first into account, ie. the value of x. For example, if x=4 and y=5, only four strings in A$ and four string in B$ will be written alternately into the text file. If x=5 and y=4, five strings will be written into the file, the extra string of the B$(5) can be anything, so the file may be damaged.

The advantage of using WRALT can be illustrated by changing the WRITE command of the example above to WRALT.

(C) Example :

```
10  REM ** DEMO **
15  DIM B(5)
20  FOR I = 1 TO 5
30  INPUT "NAME"A$(I)
40  INPUT "AGE"B(I)
50  INPUT "JOB"C$(I)
60  NEXT I
70  DOS CREATE,*1,"DEMO"
80  DOS WRALT,*1,A$(5),B(5),C$(5)
90  DOS CLOSE,*1
```


It this example, five groups of data are stored in the text file. Each pair of data will include the names, and then the ages, and also his job as in the following format.

```
NAME (1)    AGE (1)    JOB (1)
NAME (2)    AGE (2)    JOB (2)
NAME (3)    AGE (3)    JOB (3)
NAME (4)    AGE (4)    JOB (4)
NAME (5)    AGE (5)    JOB (5)
```

The records will be in this format by using WRITE instead.

```
NAME(1)   NAME(2)   NAME(3)   NAME(4)   NAME(5)
AGE(1)    AGE(2)    AGE(3)    AGE(4)    AGE(5)
JOB(1)    JOB(2)    JOB(3)    JOB(4)    JOB(5)
```

5.9.6 PUTWR in Random Access Text Files

(A) The PUTWR (put write) command is used with random access files only. In random access file, every record has a fixed length. The position of every record in the file can be easily calculated. To set the write pointer, the desired record location has to be put in the write pointer.

Note : Read/Write pointer - system pointers which point to specific areas in the files. The READ/WRITE command will follow these pointers; access and perform the read/write process.

(B) Format : DOS PUTWR,P

P is the pointer of the record. The first record begins at zero. If the file has a fixed record length of 5, the pointer of the first record is 0 the tenth record of the file will be 50.

(C) Example :

```
10 REM ** DEMO OF PUTWR **
20 FOR I = 1 TO 10
30 READ A$(I)
40 NEXT
50 DOS CREATE,*1,"DEMO",/R(9)
60 DOS WRITE,*1,A$(10)
70 DOS CLOSE,*1
80 DATA "MARISA","PETER","THOMAS","COLLIN"
85 DATA "GORDON","EDWARD","VERONICA","CHARLES"
90 DATA "DONNY","MCENROE"
```

After the above program has been executed, there are ten names in the file with a fixed record length of nine, although their actual length is not the same. For example, to update THOMAS to CONNORS, the position of THOMAS in the file has to be calculated. And the value is then put into the write pointer and replaced with CONNORS. The following shows this operation.

Example :

```
10 REM ** UPDATE **
20 DOS OPEN,*1,"DEMO"
30 P=2*9
40 DOS PUTWR,P
50 S$(1)="CONNORS"
60 DOS WRITE,*1,S$(1)
70 DOS CLOSE,*1
```

Line 30 does the calculation, THOMAS is placed at number 3, there are two records before it. So its pointer is 0+9+9, getting the sum of 18.

The following illustration will clarify this point :

```
CHARACTERS      MARISA   PETER     THOMAS    COLLIN   GORDON
                ^       !^        !^        !^       !^
R/W POINTERS    0        9         18        27       36
```

Note : After "CONNORS" has been written into the file, the write pointer points to the record after it. (i.e. COLLIN) So, in order to make changes on other names afterwards, it is not necessary to calculate the pointer from the beginning. Only DOS CLOSE will set all pointers to zero. The PUTRD command do the similar job which will be explained afterwards.

5.9.7 GETWR in Random Access Text Files

(A) This command (get write) is self-explanatory. The write pointer will match the current record as they are being written into the file. For example, if there are already seven records in the file with a fixed record length of 8, the write pointer will be set to 56.

(B) Format : DOS GETWR,*WN,P

Similar to PUTWR, this command will put the result into the variable. It can be retrieved by BASIC command PRINT P.

Note : the variable P used in PUTRW and GETWR is just an example. Any other variable name will do.

5.9.8 READ FILES

(A) The DOS command READ allows the user to retrieve the record in the text files. Like WRITE, must as well be preceded by OPEN CLOSE after being used. READ the files by the work number given by the OPEN command.

(B) Format : DOS READ *wn A$(x),
B$(x,y),C(x),D(x,y),....

Same as the procedure of the WRITE command, records have to be read into the memory in the form of arrays.

(C) Example : To retrieve the text file - DEMO with ten names created in the above example.

```
10 REM DEMO PROGRAM
20 DOS OPEN,*1,"DEMO"
30 DOS READ,*1,Y$(10)
40 DOS CLOSE,*1
50 FOR H = 1 TO 10
60 PRINT Y$(H)
70 NEXT H
80 END
```

After this example is run, the following result will appear.

```
MARISA
PETER
CONNORS
COLLIN
GORDON
EDWARD
VERONICA
CHARLES
DONNY
MCENROE
```

NOTE 1 : The string name is user defined. If the record is written into the file by the command 'DOS WRITE,*1,R$(10)'. It is not necessary to retrieve the record by R$, the user can use any other string names like Y$ in the above example.

NOTE 2 : A variable record, like D(10), is not necessary to do the dimension declaration : DIM D(10) in BASIC. THE DOS COMMAND READ AND RDALT WILL DO ALL DIMENSION FOR THE USER.

5.9.9 RDALT

RDALT (read alternate) is a DOS command used in pair with WRALT.

(B) Format : DOS RDALT,*wn,A$(10),B$(10)
Similarly RDALT read the first record of all arrays before the second ones.

(C) Example : To retrieve the information by the command RDALT.

```
10 DOS OPEN,*1,"DEMO"
20 DOS RDALT,*1,A$(5),B(5),C$(5)
30 DOS CLOSE,*1
40 FOR I = 1 TO 5
50 PRINT A$(I),B(I),C$(I)
60 NEXT I
```

The following result will appear :

```
NAME 1     AGE 1      JOB 1
NAME 2     AGE 2      JOB 2
NAME 3     AGE 3      JOB 3
NAME 4     AGE 4      JOB 4
NAME 5     AGE 5      JOB 5
```

Caution : Do not use RDALT to read a text file written by the WRITE command. Also, do not use READ to retrieve records written by the command WRALT. Use READ to retrieve records written by WRITE and RDALT for records written by WRALT.

5.9.10 PUTRD AND GETRD in Random Access Files

(A) PUTRD (put read) and GETRD (get read) are similar to PUTWR and GETWR. Their nature and principle are the same except the pointer is the read pointer instead of the write pointer.

(B) Format : DOS PUTRD,*wn,P
DOS GETRD,*wn,P

(C) Example : To retrieve only the sixth record of the "DEMO" example.

```
10 REM DEMO PROGRAM
20 DOS OPEN,*1,"DEMO"
30 P=54
40 DOS PUTRD,*1,P
50 DOS READ,*1,T$(1)
60 DOS CLOSE,*1
70 PRINT T$(1)
```

The answer 'EDWARD' will be printed out when the above program is executed.

Note : GETRD and GETWR needs a variable to hold them; but we can directly use a value for command PUTRD and PUTWR. For example, in line 40 of the above example, we can use DOS PUTRD, *1,54.

5.9.11 Appending Files

(A) The DOS command APPF (append file) adds text to the end of a text file. This is particularly useful to extend the information in a sequential text file. This command performs an OPEN on a file that already exists, then sets the read pointer and write pointer to the byte at the end of file.

(B) Format : DOS APPF,*1,"F.N"[/2]

(C) Example : To add five more names to the file "demo" created above.

```
10 REM DEMO
20 FOR I = 1 TO 5
30 READ Y$(I)
40 NEXT I
50 DOS APPF,*1,"DEMO"
60 DOS WRITE,*1,Y$(5)
70 DOS CLOSE,*1
80 DATA "LENDL","LLOYD","MARTINA"
90 DATA "FLEMING","BORG"
```

The five names will add to the back of the file after the program has run.

Chapter 6

UTILITIES

6.1 The Utilities

Utilities are aids that help the user to deal with the files. There are eight utilities in the master diskette :

BOOT	(contains the handler)
INIT	(initializes a new disk)
LOCK	(a demonstration of all the DOS commands)
UNLOCK	(looks files)
COPY	(unlocks files)
HELP	(error codes enquiry)
MERGE	(merge text files)

6.2 BOOT

This contains the handler that is used by both the system and the user. The use of handler will be fully explained in the next chapter. This utility CANNOT be removed, otherwise, the handler is lost and all the DOS commands will not work because they are called through the handler.

6.3 Using the INIT Program to Format a New Disk

6.3.1 Operation Procedure

1. Insert the master disk to drive 1 or driver 2.

2. Type 'DOS URUN,"INIT" [/DR]', the option of DR is the drive number of the master disk.

3. There will be a message "INSERT FORMAT DISK THEN PRESS ANY KEY" appears on the screen. Put the new disk into drive 1 or drive 2 and press any key, the master may then be taken away from the drive.

4. Now you will see the question "ARE YOU SURE? (Y/N)" type "Y" to continue the program and "N" to abort it.

5. When you see "DRIVE NO. D(1 OR 2) ?" answer the drive number by typing "1" or "2".

6. After the message "DOUBLE TRACK DENSITY ? (Y/N)" type "Y" for double track density (96 TPI) and "N" for single track density (48 TPI).

7. When you see "SINGLE"(1) OR DOUBLE(2) SIDE ?" answer by typing "1" or "2". If you have chosen double track density, this question will not appear.

8. The next question will be "ARE YOU SURE ? (Y/N)". It is the same as the first question, if you type "N", the program will be aborted.

9. The new diskette will be formatted. After formatting, you will be asked for the name of the new diskette :
"DISK NAME (MAX.8 CHARS) ?"
Type in any character for the name of your new diskette, but the maximum number of the characters is 8.

10. The last question will be "DATE (DD-MM-YY) ?" Enter the date your new diskette is created, the maximum number of characters is 8.

After formatting, the new diskette can be used to boot up the system and you can find a file call "BOOT" on the new diskette. Do not delete the "BOOT" file, otherwise, the diskette cannot be used to boot up the system.

Example :

```
:DOS URUN,"INIT"
INSERT FORMAT DISK THEN PRESS ANY KEY

ARE YOU SURE ? (Y/N) [Y]
DRIVER NO. (1 OR 2) ? [2]
DOUBLE TRACK DENSITY ? (Y/N) [N]
SINGLE(1) OR DOUBLE(2) SIDE ? [2]
ARE YOU SURE ? (Y/N) [Y]
DISK NAME (MAX.8 CHARS) ? [NEW DISK]
DATE (DD-MM-YY) ? [01-01-85]
```

The content in square blacket at the end of each question should be entered by the user.

6.3.2 Verification

There is a verifying program after initialisation. This routine checks all the tracks and bytes in the diskette just initialised.

a) The 'O.K' message indicates that the track is perfect and ready for use.

b) All empty tracks were filled with zeros during initialisation. Message 'BAD' means certain bytes on that track is not a zero. They will be shown afterwards. You are suggested to initialise the diskette once more.

c) The message 'R,W' error indicates that the track was damaged. There are some difficulties in reading and writing on that track. An $FF will be filled into the relative block which means that block is useless. So, the number of free blocks will decrease by 1 with one R/W error.

d) SEEK error means the driver cannot find that track. The result is the same as an R/W error.

6.4 Using the DEMO Program

Remove the write protective strap from the diskette and insert the master diskette into drive 1 or drive 2. Type 'DOS RUN,"DEMO" [/DR]', the option of DR is the number of the drive which the master diskette has been inserted.

At the beginning of the program, a symbol table will be displayed on the screen. The symbols indicates the format of each command. The menu of the program will be displayed after pressing the space bar. Then you can choose any of the options in the menu.

(DOS MENU)

1. FUNDAMENTAL COMMANDS

   CAT,SAVE,LOAD,RUN,DEL

2. TEXT FILE COMMANDS

   CREATE,OPEN,CLOSE,WRITE,READ
   WRALT,RDALT,APPF,GETRD,PUTRD
   GETWR,PUTWR

3. OTHER COMMANDS

   REN,NEW,URUN

4. SYMBOLS

5. ERROR CODES

6. EXIT

WHICH OPTION (1-6) ?

You can choose any of the above numbers.

In the demonstration, characters displayed in white indicate those characters which are typed in by the user. The cyan characters represent the computer print out.

This program will be destroyed after running, so if you want to run the program again after exit, please load it again from the disk.

6.5 Software write Protect - Lock and Unlock

There are two kinds of software protection : delete protection and write protection. Delete protected means cannot be removed and it is indicated by the alphabet (D) at the rightmost column of that file; it is that file cannot be written over and indicated by (W) at the rightmost column. Write protection is used to deal with text files only. If a file is both write and delete protected, a (d,W) will appear at the rightmost column.

6.5.1 Lock Files

1. Insert the master diskette and type DOS URUN,"LOCK".

2. A question "FILENAME ?" which requires the object file name will be asked.

3. There are three options : delete protect the file only; write protect only and both. Just type in the option number, there is NO NEED to type the return key.

4. Finally, input the driver number of the object file. 1 for drive and 2 for drive 2; again there is no need to type the RETURN key.

6.5.1.1 Hardware Write Protect

The LOCK command allows the user to protect a particular file. To ensure that ALL files on a certain diskette will not be overwritten and cause losses, it is necessary to write protect a diskette by covering up the squarish write-protect notch on the side of the disk. Stick-on adhesive labels are supplied for this purpose on purchasing diskettes, but anyway, any piece of sturdy tape will do.

Caution : It is dangerous to repeal the write-protect label of the master disk. If it is accidentally damaged, the system cannot be boot up again. It is saver to run the "DEMO" program by loading it first. Then substitutes the master diskette by a working diskette before "DEMO" runs.

6.5.2 UNLOCK FILES

The procedure is exactly the same as LOCK but it's purpose is just the opposite. File has to be UNLOCK before it can be removed or written.

6.6 COPY FILES

COPY is a utility to make copy from an existing file to a non-existing file. Sometimes it is used to make backup of important files. All files can be copied.

1. Place the master diskette into the drive and type DOS URUN,"COPY".

2. Input the source file name and type return.

3. Input the source file's drive number, there is NO NEED to hit return.

4. Then input the object file name and type return

5. Input the object file's drive number, there is NO NEED to enter RETURN.

Note : Single-drive user CANNOT copy files from a disk to another. He cann ONLY copy files within the same disk.

6.7 Using the help Program to Find the Meanings of the Error Codes

6.7.1 Operation Procedure

1. Insert the master diskette to drive 1 or drive 2.

2. Type 'DOS URUN,"HELP" [/DR]', where the option of DR is the drive number of the master diskette.

3. There will be a message "INPUT ERROR CODE :" displayed on the screen. Type the error codes that you want to find.

4. After the error messages are displayed, "INPUT ERROR CODE:" will appear again, the next error can be entered. You can abort it by pressing the RETURN key, then there will be a message "ERROR NOT FOUND" printed on the screen and the program aborted.

6.7.2 EXAMPLES :

```
:DOS URUN,"HELP"
INPUT ERROR CODE:[123]
PARENTHESIS MISSING

INPUT ERROR CODE:[1Ø1]
RECORD NUMBER EXCEEDS FILE SIZE

INPUT ERROR CODE:(CR)
ERROR NOT FOUND
```

The contents in the square blackets at the end of each question are to be entered by the user.

6.8 MERGE

6.8.1 Operation Procedure

MERGE is a utility for text files ONLY. It's purpose is to link up text file sequentially.

1. Insert the master disk and type DOS URUN,"MERGE"

2. It will ask for a filename by displaying the message "DESTINATION FILENAME ?". Destination file means the object file. Input a filename which must not already exist and press return.

3. Then input the destination file's drive number.

4. The message :

```
INPUT YOUR OBJECT FILE IN SERIES
PRESS CNTL 'S' WHEN FINISHED

FILENAME ?
```

will be displayed.

Then you can input the object files. Type in the first file to be merged; press return, then input it's drive number after the message 'DRIVE NO:'

5. The merge procedure now begins, 'READING ...' will be displayed out when the program read the content of the object file. 'WRITING...' will appear on the screen when the content is being written into the destination file.

6. After the files have been merged, the message 'FILENAME' will appear again to request the next file that is going to be merged. Drive number has to be input also.

7. Procedures 5 and 6 will be perform again and again. After finished merging all the object files, press cntl 'S' when 'FILENAME ?' is displayed and the whole job is done.

Note 1 : For single-drive user, the destination file and all the object fils MUST be on one diskette. For dual-drive user, files can be located at either drive 1 or drive 2.

Note 2 : empty files cannot be merged, otherwise the message, 'FILE SIZE EQUAL TO ZERO' will appear and the process will be aborted.

Note 3 : The Utility merges files by unit of 20K, files bigger than 20K have to go through the process 'READING' and 'WRITING' more than one time. For example, an object file of 64K has to be merge four times :

```
READING ...
WRITING ...     (merge 20K)

READING ...
WRITING ...     (merge 20K)

READING ...
WRITING ...     (merge 20K)

READING ...
WRITING ...     (merge the rest of 4K)
```

6.8.2 EXAMPLE : To merge 'T1','T2','T3' in drive 1 to form 'TT' in drive 2.

```
: DOS URUN,"MERGE"

DESTINATION FILENAME ? [TT]  [HIT RETURN]
DRIVE NO.: [2]

INPUT YOUR OBJECT FILES IN SERIES
PRESS CNTL 'S' WHEN FINISHED

FILENAME : [T1] [HIT RETURN]
DRIVE NO. : [1]

READING ...
WRITING ...

FILENAME : [T2] [HIT RETURN]
DRIVE NO. : [1]

READING ...
WRITING ...

FILENAME : [T3] [HIT RETURN]
DRIVE NO. : [1]

READING ...
WRITING ...

FILENAME : [CNTL S]
:READY
```

The job is done, the result file (destination file),'TT', will be on drive 1.

6.9 A General Note on Utilities :

User has to press RETURN after they input the filename but this is NOT necessary after inputing the drive numbers.

Chapter 7

HOW TO USE THE "HANDLER" TO CALL THE SUBROUTINES IN DOS

7.1 Introduction

THIS SECTION HELPS THE PROGRAMMERS TO WRITE UTILITIES BY MAKING USE OF SOME OF THE SUBROUTINES IN THE DOS PROGRAM.

IN ADDITION TO "CALLING THE DRIVER", THIS SECTION ENABLES PROGRAMMERS TO FULLY UTILIZE THE SUBROUTINES IN DOS AND WRITE THEIR UTILITY PROGRAMS.

Two kinds of routines can be called. The first one is the normal subroutines in DOS, the second is the routines that execute the DOS commands as defined in the previous chapters.

Programmers can use any one of the two methods to call the former. The first method is to fill the handler block, which will be defined later, and call "handler"--addr-> $B72C, the resultant parameters will be passed back by the handler block. The second method is to fill the parameters (normally registers) and call "handler2"--addr->$B700, the resultant parameters will be passed back by designated registers and memory space.

Calling the second kind of routines, in addition to the command code, programmers should fill the tokens needed by that command following the command code. Calling "handler1", the command defined(eg. close, open etc.) will be executed.

The command codes of the first kind of routines are from 0 to 127, those of the second kind of routines are from 128 to 143. If the command code is not in the range, error code 137 will be displayed on the screen.(in DOS version 1.3, only 0 to 36, 128 to 133 are used as command codes)

.2 Command Code Listing :

The following is the command code listing :

COMMAND CODE	ROUTINE NAME	PARAMETERS PASSED	RESULT	PURPOSE
1	CKFN	RB POINT TO FILE NAME TO BE CHECKED. FN. MUST START WITH THE TOKEN OF OPEN QUOTATION MARK & END WITH THE TOKEN OF CLOSE QUOTATION MARK.	FILE NAME SHOULD BE FOUND, ELSE ERROR 130 R7, R9, RA, RE, RF CHANGED	
2	CKFN+3	D=2 RB PTR. TO FILE	FILE FOUND, D=Ø RETURNED NOT FOUND, D=1.	
3	CKFN-3	RB PTR.TO FN. TO BE CHECKD	FN. SHOULD NOT BE FOUND, ELSE, ERROR. R7, R9, RA, RE, RF CHANGED.	
5	FD.ADD	FILE MUST BE OPENED OR CREATED	R8 STORE THE FILE DESCRIPTION ADDR. R9 CHANGED.	
1Ø	SCHDIR	RF.Ø=SECTOR NO.	SEARCH THE DIR SECTOR LOAD IT INTO $BE8Ø. (THE BEGINNING OF BUFFER) R9 PTR. TO DE.ATTR OF THAT DIR. RE, R9, RF CHANGED.	
18	PHYRW	D=DRIVE/HEAD SIDE, MSB: Ø-READ 1=WRITE RC.Ø=SECOR NO. RC.1=TRACK NO. RE=BUFF, PTR.	THE DESIGNATED SECTOR IS LOADED FROM DISKETTE TO THE BUFFER OR STORED FROM BUFFER. R9, RE CHANGED.	

23	FILLFD	R8=SOURCE R7=DESTINATION RF.Ø=BYTE COUNT	DATA MOVED FROM SOURCE TO DESTINATION. NO. OF DATA BYTES MOVED IS DEFINED BY BYTES COUNT.
24	PHYBLK	D=Ø:READ PTR. D=3:WRITE PTR. FILE ALREADY OPENED OR CREATED.	SECTOR NO. AND BLOCK NO. IS CALCULATED FROM THE READ OR WRITE POINTER AS DEFINED BY D. THE RESULTS STORED IN THE APPROPRIATE LOCATIONS OF FD.(eg. FD.wphy) RF, R8, RE CHANGED. IF PTR. >FILE SIZE, 1Ø3 ERR.
25	SRHFRBLK	FILE ALREADY OPENED OR CREATED	SEARCH FREE BLOCK,MARK THAT BLOCK MAP. UPDATE THE FREE BLOCK NO. UPDATE THE FD.WPHY R9, R8 CHANGED.
28	RWSETR	D=Ø : READ D=3 : WRITE FILE ALREADY OPENED OR CREATED	SECTOR DESIGNATED IS READ TO BUFFER. BUFFER CONTENT IS WRITTEN FROM BUFFER TO DESIGNATED SECTOR. RE, R8, RC, RF CHANGED.
128	SAVE	DESCRIBED LATER	USAGE IS SAME AS THE NORMAL COMMAND.
129	LOAD	DESCRIBED LATER	USAGE IS SAME AS THE NORMAL COMMAND.
130	RUN	"	"
131	DEL	"	"
132	OPEN	"	"
133	CREATE	"	"
134	APPF	"	"
135	URUN	"	"
136	CLOSE	"	"
137	GETRD	"	"
138	PUTRD	"	"
139	GETWR	"	"

140	PUTWR	DESCRIBED LATER	USAGE IS THE SAME AS THE NORMAL COMMAND
141	READ	"	"
142	WRITE	"	"
143	NEW	"	"

7.3 Structure of Handler Block

The structure of the handler block is shown below.

ORG $BE00

ADDR.	NAME	BYTE	DESCRIPTION
BE00	HD.BLK7	2	STORE REGISTER 7
BE02	HD.BLK8	2	STORE REGISTER 8
BE04	HD.BLK9	2	STORE REGISTER 9
BE06	HD.BLKA	2	STORE REGISTER A
BE08	HD.BLKB	2	STORE REGISTER B
BE0A	HD.BLKC	2	STORE REGISTER C
BE0C	HD.BLKD	2	STORE REGISTER D
BE0E	HD.BLKE	2	STORE REGISTER E
BE10	HD.BLKAD	1	STORE D ACCUMULATOR
BE01	HD.BLKF	2	STORE REGISTER F
BE03	HD.FN	30	RESERVE FOR STORING FILE NAME & TOKENS.

7.4 Examples

Examples of calling dos subroutines:

A. Method 1 (calling handler2--$b700):

PHYBLK can be called by DOS by using method 1. Since rf, r8 and re will be changed after the execution of the subroutine, they should be saved first in the program. Assume the file has already been opened and the read pointer given.

```
: :
: :
glo rf; stxd
ghi rf; stxd
ghi r8; stxd
glo re; stxd
ghi re; stxd
ldi 0 ; call $b700; dc 24
: :
: :
```

B. Method 2 (calling handler##$b702c)

Before calling the handler, the handler block must be filled first.

In the same case of the previous example, only D should be filled.

```
: :
: :
ldi $be; phi re
ldi $10; pl0 re
ldi 0  ; str re
call $b720 ; dc 24
: :
: :
```

After executing the instructions, sector no. and block no. will be stored into the appropriate memory space in the file descriptor.

C. Calling the DOS command by handler($b72c)

Assume the DOS CREATE,*1,"ABC"/2(CR) is needed to be executed. The tokens are as follows :

```
  DOS CREATE,*      1      '  "  A  B  C  "  /       2    CR
TOKENS :    CA D2 00000001 C2 CF 41 42 43 CE CB D2 00000002 0D
```

Another examples :

```
  DOS READ,*      1      ,  @  4  4  0  0  ,  @  0  0  1  1  CR
TOKENS :    CA D2 00000001 C2 D0 34 34 30 30 C2 D0 30 30 31 31 0D
```

Now, open a existing file in an assembly program : DOS OPEN,*1,"ABC"

```
: :
: :
call $b72c; dc 132
dc $CA,$D2,$00,$00,$00,$01,$C2,$CE,$41,$42,$43,$0D
: :
: :
```

NOTE : 132 IS THE COMMAND CODE FOR "OPEN", $D2 MEANS THE FOLLOWING NUMBER IS A CONSTANT. THE TOKENS WE NEED TO FILL ARE THOSE AFTER THE COMMAS FOLLOWING THE DOS COMMANDS.

APPENDIX A

SUMMARY OF DOS COMMANDS

* SYMBOLS

DR - DRIVE NUMBER
F.N. - FILE NAME
F - FORTH FILE TYPE
B - BINARY FILE TYPE
SADR - STARTING ADDRESS
EADR - ENDING ADDRESS
WN - WORKING NUMBER
R(N) - FIXED RECORD LENGTH
VAR - VARIABLE

* GENERAL COMMAND

1. DOS CAT [/DR]

2. DOS SAVE,"F.N" [,F] [,@SADR] [,@EADR] [,B] [/DR]

3. DOS LOAD,"F.N" [/DR]

4. DOS RUN,"F.N" [/DR]

5. DOS URUN,"F.N" [/DR]

6. DOS DEL,"F.N" [/DR]

7. DOS REN,"F.N","F.N" [/DR]

8. DOS NEW

* TEXT FILE COMMAND

9. DOS CREATE,*WN,"F.N" [,/R(N)] [/DR]

10. DOS OPEN,*WN,"F.N" [/DR]

11. DOS CLOSE, [*WN]

12. DOS WRITE, [*WN] [,/R(N)] ARRAYS

13. DOS READ, [*WN] [,/R(N)] ARRAYS

14. DOS WRALT, [*WN] ARRAYS

15. DOS RDALT, [*WN] ARRAYS

16. DOS APPF,*WN,"F.N" [/DR]

17. DOS GETRD, [*WN,] VAR

18. DOS PUTRD, [*WN,] VAR

19. DOS GETWR, [*WN,] VAR

20. DOS PUTWR, [*WN,] VAR

APPENDIX B

COMX 35 DOS ERROR MESSAGE

100 TEXT FILE RECORD LENGTH IS NOT FIXED

101 RECORD NUMBER EXCEEDS FILE SIZE

102 DISK READ ERROR

103 READ OR WRITE POINTER EXCEEDS FILE SIZE

104 SECTOR READ OR WRITE ERROR

105 SEEK ERROR

106 DISKETTE WRITE PROTECTED

107 FILE LOCKED

110 SLASH MISSING

111 END OF COMMAND NOT FOUND

112 WORK ID. NUMBER ERROR

113 RECORD LENGTH MISSING

114 MORE THAN 8 ARRAYS ALREADY EXIST

115 STRING OR ARRAY VARIABLE ERROR

116 NOT INTEGER OR TOO LARGE INTEGER EXIST

117 NOT A FIXED RECORD LENGTH FILE TYPE

118 STRING EXCEEDS RECORD LENGTH

120 ADDRESS SYNTAX ERROR

121 DRIVE NUMBER ARGUMENT OR SYNTAX WRONG

122 NOT ENOUGH DISK MEMORY

123 PARENTHESIS MISSING

124 INPUT FILE NAME ERROR

25 COMMA MISSING

26 COMMAND DOES NOT EXIST

27 SYNTAX ERROR ON 'SAVE' COMMAND

28 QUOTATION MARK MISSING

29 FILE NAME ALREADY OPENED OR EXISTS IN DIRECTORY

30 FILE NOT FOUND IN DIRECTORY

31 NO SUCH KIND OF PROTECTION

32 MORE THAN 5 FILES ALREADY OPENED

33 FILE NAME NOT YET OPENED

34 MORE THAN 51 DIRECTORY ENTRIES

35 FILE WRITE PROTECTED

36 DISK OR FILE NON-COPYABLE

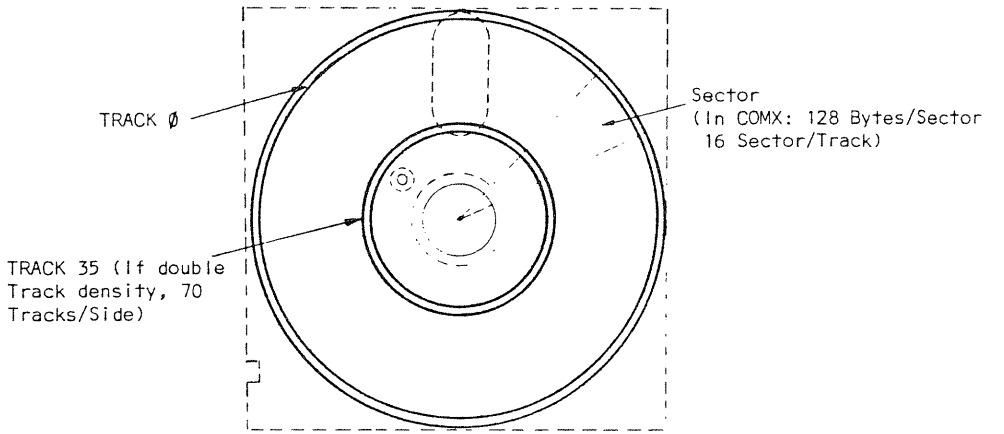
37 NO SUCH HANDLER COMMAND CODE

APPENDIX C

DISK STRUCTURE

THIS SECTION EXPLAINS THE RELATIONSHIP AMONG "SIDE", "TRACK", "SECTION" AND "TRACK DENSITY".

The relationship among track, sector and track density is shown in the figure below.



* If a double-sided diskette is used in a drive with double side feature, double-side option can be selected in 'INIT'.

Fig. 1

As shown in the diagram, track is the cycle lying on the diskette, the outer most circle is track Ø. One track can be divided into blocks called sector, normally one sector consists of 128 to 256 bytes, which is predefined in the format procedure.

Normally, 48 tpi(track per inch) stands for single track density, 96 tpi stands for double track density. In the formatting procedure, 35 tracks are formed if we select single track desity, otherwise 70 tracks are formed. (NOTE : ONLY USE double track density diskette in the double track density disk drive.)

If the disk drive is with double side feature, double side option can be selected.

APPENDIX D

THE COMX DISK DRIVE can be accessed directly from machine language through the use of the DRIVER subroutine, which is part of the DOS.

Every diskette initialized by the COMX DISK DRIVE is separated into 35 tracks for side 1 and side 2. Basically, the tracks are arranged in separate concentric circles, with the large hole in the centre of the diskette forming the common center of the circles. Track Ø is on the outer edge of the diskette, while track 34 is nearest to the center. The head of the disk drive moves to different tracks on the diskette, where it either reads information off the diskette, or writes information onto the diskette.

There are 16 sectors on each track of the diskette. The sectors within a track are individually numbered, consecutively, Ø to 15 around the diskette. Sectors allow the user to work with single blocks of 128 bytes.

To use the DRIVER subroutine, the user must fill up the information to the COMMUNICATION BLOCK. Then call the starting address of the DRIVER subroutine (at location $C002).

COMMUNICATION BLOCK

Location	Description
$BF30	Command code Ø = RESTORE -- SET HEAD TO TRACK Ø 1 = SEEK -- SEARCH DESIRED TRACK 2 = READ SECTOR -- SEEK TRACK & READ ONE SECTOR 3 = WRITE SECTOR -- SEEK TRACK & WRITE ONE SECTOR
$BF31	Drive number and side number $14 = DRIVE 1 & SIDE 1 $34 = DRIVE 1 & SIDE 2 $18 = DRIVE 2 & SIDE 1 $38 = DRIVE 2 & SIDE 2
$BF32	Track numbr Ø-34
$BF33	Sector number Ø-15
$BF34	Single or double track density Ø = single track density $FF = double track density

$BF35	High byte of buffer location
$BF36	Low byte of buffer location
$BF37	Stepping rate Ø = 3Ø ms 1 = 2Ø ms 2 = 12 ms 3 = 6 ms
$BF38	Previous drive
$BF39	Current drive track number
$BF3A	Previous drive track number
$BF3B	Read or write status
$BF3C	Seek status

APPENDIX E

OOTING PROCEDURE OF COMX DOS V1.2

. When the first DOS command is executed, a subroutine in ROM hich is called 'boot0' will run. This subroutine load the ector 15 of track 0 on the disk to the memory location of $4000 f RAM.

. Then the program counter will set to $4000 and execute the rogram which has been already loaded. This subroutine is called bootl'. The 'bootl' program loads the sector 0 to sector 7 of rack 1 on the disk to the memory location of $B700 to $BAFF of AM. This part is called 'boot2'.

. The 'boot2' contains three things. The first thing is the andler, it is used for users to call the DOS subroutines. econd thing is the jump table, this table contains many entry oints of subroutines. The entry points is used for calling by he handler. The third thing is a subroutine which will be xecuted after running 'bootl'.

. The subroutine of 'boot2' is for setting some flags and arameters of the system.

APPENDIX F

The hardware structure of the floppy disk controller is shown clearly in the figure below. The circuit is composed of 5 major parts : decoder circuit, latch circuit, a BKB EPROM chip, clock circuit, driver circuit and the floppy disk controller chip (1770). Please refer to fig. 2 & fig. 3.

NOTE :

1. DS=1, Q=1, OUT2 COMMAND IS ISSUED, THEN DATA LATCH TO F3.

2. Q=0, OUT2 OR INP2 IS ISSUED, THEN 1770 IS SECLECTED.

3. WHEN ADDRESS IS FROM C000 TO DFFF OR DD0 TO DDF, F4 IS SECTECTED.

4. CLOCK TO 1770(F5) IS 8 MHZ+-1%.

5. WHEN THE CONTROLLER CARD IS SELECTED, DS=1 & D1 IS ON.

6. WHEN DISK DRIVE IS SELECTED F9 PIN 15 ALWAYS LOW(ENABLE).

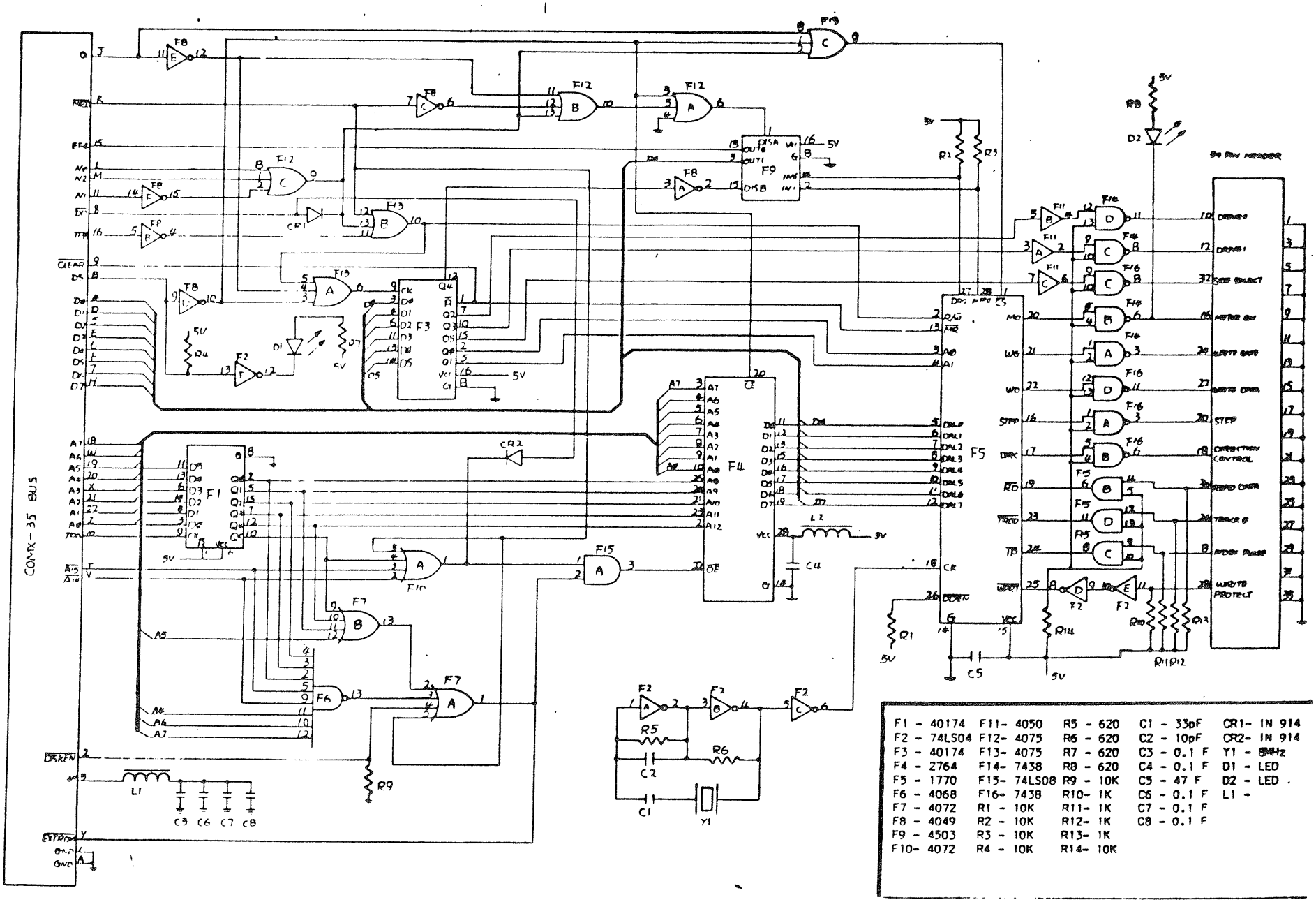
7. F1 IS USED TO LATCH THE HIGH BYTE OF THE ADDRESS AT THE RISING.

8. F3 IS USED TO LATCH THE SELECT CODES WHICH SELECT THE REGISTERS USED IN THE CONTROLLER 1770, THE DRIVE BEING USED AND WHICH SIDE OF THE DISK TO BE USED.

9. F9 LATCHES THE DRQ & INTRQ SIGNALS FROM THE CONTROLLER 1770 IN ORDER TO KEEP TRACK OF THE STATUS OF THE CONTROLLER.

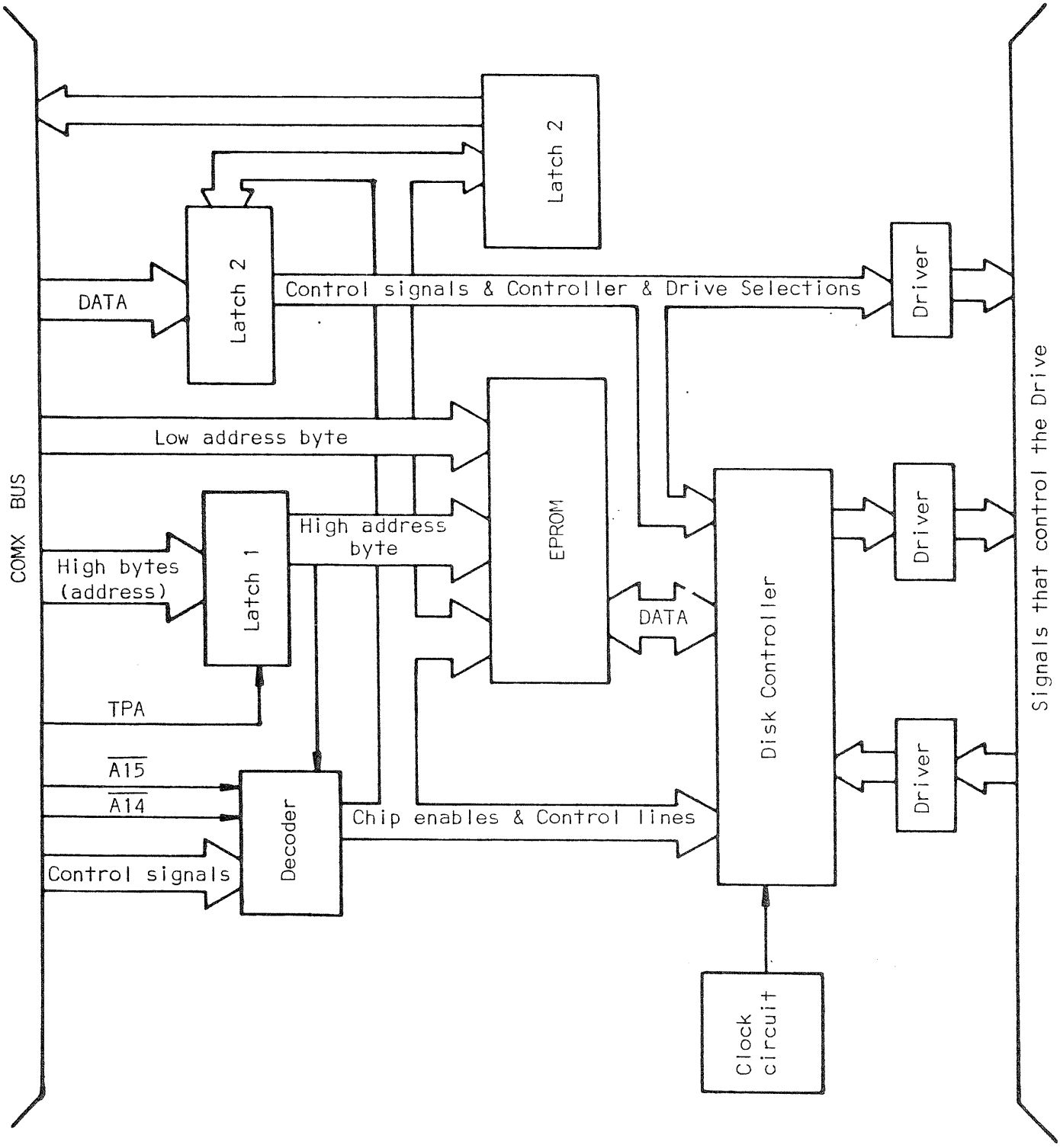
10. F4 IS AN EPROM, IN WHICH THE DOS IS PROGRAMED. AFTER THE DISK CONTROLLER CARD IS SELECTED, F4 IS SELECTED. IN EXECUTING THE DOS, OE' IS ENABLED TOO.

11. ABOUT THE 1770 SIGNAL LINES, REFER TO THE 1770 DATA SHEET.



F1 - 40174	F11- 4050	R5 - 620	C1 - 33pF	CR1- IN 914
F2 - 74LS04	F12- 4075	R6 - 620	C2 - 10pF	CR2- IN 914
F3 - 40174	F13- 4075	R7 - 620	C3 - 0.1 F	Y1 - 8MHz
F4 - 2764	F14- 7438	R8 - 620	C4 - 0.1 F	D1 - LED
F5 - 1770	F15- 74LS08	R9 - 10K	C5 - 47 F	D2 - LED
F6 - 4068	F16- 7438	R10- 1K	C6 - 0.1 F	L1 -
F7 - 4072	R1 - 10K	R11- 1K	C7 - 0.1 F	
F8 - 4049	R2 - 10K	R12- 1K	C8 - 0.1 F	
F9 - 4503	R3 - 10K	R13- 1K		
F10- 4072	R4 - 10K	R14- 10K		

Block Diagram of the Disk Controller Card



* The bus labelled 'DATA' are all connected together.

Fig. 2