ACORN
SYSTEM 5 HANDBOOK

ACORN
COMPUTER

400,056/1/1st March 1983

The product described in this manual and products for use with it, are subject to continuous development and improvement. All information of a technical nature and particulars of the product and its use (including the information and particulars in this maual) are given by Acorn Computers in good faith. However, it is acknowledged that there may be errors or omissions in this manual. A list of details of any amendments or revisions to this manual can be obtained upon request from Acorn Computers Technical Enquiries. Acorn Computers welcome comments and suggestions relating to the product and this manual.

All correspondence should be addressed to:

Technical Enquiries
Acorn Computers Limited
Fulbourn Road
Cherry Hinton
Cambridge
CB1 4JN

All maintenance and service on the product must be carried out by Acorn Computers authorised dealers. Acorn Computers can accept no liability whatsoever for any loss or damage caused by service or maintenance by unauthorised personnel. This manual is intended only to assist the reader in the use of the product, and therefore Acorn Computers shall not be liabel for any loss or damage whatsoever arising from the use of any information or particulars in, or any error or omission in, this manual, or any incorrect use of the product.

WARNING: THE COMPUTER MUST BE EARTHED

IMPORTANT: The wires in the mains lead for the computer are coloured in accordance with the following code:

GREEN AND YELLOW	EARTH
BLUE	NEUTRAL
BROWN	LIVE

As the colours of the wires may not correspond with the coloured markings identifying the terminals in your plug, proceed as follows:

The wire which is coloured green and yellow must be connected to the terminal in the plug which is marked by the letter E, or by the safety earth symbol ¥ or coloured green, or green and yellow.

The wire which is coloured blue must be connected to the terminal which is marked with the letter N, or coloured black.

The wire which is coloured brown must be connected to the terminal which is marked with the letter L, or coloured red.

HOW TO USE YOUR SYSTEM 5 HANDBOOK

INTRODUCTION

This handbook describes the Acorn System 5 computer and is divided into two parts: Software and Hardware. These parts are sub-divided into sections on the Disc Operating System, and hardware details of the individual boards.

NEW USERS

The handbook is technically complete but may initially be baffling to new users. New users wishing to get their System 5 up and running should go straight to Disc Operating System, Part 2, Section 1. This section shows how to switch on the machine and use the Disc Operating System facilities,

NOTE

There are references in the Disc Operating System section to System 5 Basic. Please note that Basic is not supplied as standard with System 5 machines configured as Econet File Servers.

SYSTEM 5 HANDBOOK

CONTENTS

Written and Produced for Acorn Computers Limited by Arch Technical Publications Limited,
St. Lukes House, 5, Walsworth Road, Hitchin, Herts.

DISC OPERATING SYSTEM
DESCRIPTION

CONTENTS

Written and Produced for Acorn Computers Limited by Arch Technical Publications Limited,
St. Lukes House, 5, Walsworth Road, Hitchin, Herts.

CONTENTS (Cont'd)

Paragraph **Page**

CONTENTS (Cont'd)

Paragraph **Page**

TABLES

FIGURES

1. INTRODUCTION

Your System 5 is supplied with a Disc Operating System (DOS) built in to its main memory (ROM). The DOS provides you with:

— Control of your screen and printer.

— Screen editing facilities.

— The means to save and load programs, data and text to and from disc.

- File management facilities.

All of the above facilities are available to you directly from the keyboard, or you can "embed" them within your programs.

1.1 DISC HANDLING AND SAFE KEEPING

Programs data and text that you have stored on disc may represent hours, weeks or months of effort. By careful and methodical handling and the use of simple security procedures your programs, data and text stored on disc will not be lost. The following security procedures should be observed:

- DO NOT REMOVE the circular magnetic disc from it's square black protective jacket.
- DO NOT TOUCH the exposed magnetic surfaces.
- AVOID DUST; keep the discs in their protective jackets and store in a storage box when not within the drive.
- DO NOT BEND, drop them, or rest heavy objects on them.
- KEEP THEM AWAY from strong magnetic fields such as those generated by televisions, monitors, tape recorders, transformers, telephones and calculators.

  AVOID excessive heat, moisture and direct sunlight.
- USE ONLY FELT TIPPED PENS to write on the labels and don't press too hard.
- INSERT DISC CAREFULLY; if a disc rotates noisily, open drive door and adjust the disc position.

You will want to protect some discs against accidental overwriting. Each disc has a Write Protection Notch, see Figure 1. When the Write Protection Notch is covered with a self adhesive tab (supplied with each box of discs) you cannot overwrite any programs, data or text stored on that disc.

You may have individual programs, data or text stored on disc that you want to write protect, while writing to other areas on the same disc. Your DOS can do this for you, but we will describe how this is done later.

Finally, it is recommended that you keep copies of discs containing important programs, data or text.

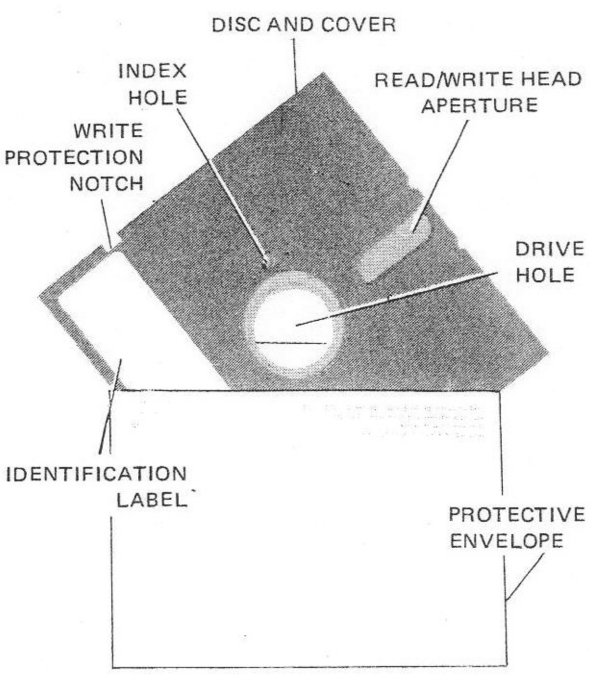


Figure 1. The Mini-Floppy Disc

1.2 WHAT IS SUPPLIED IN THE SYSTEM 5

Your System 5 will have been supplied according to your requirements, but will include a minimum of the following items:

HARDWARE
- Rack, PSU and Backplane
- 6502A Processor Board
- 32K Dynamic RAM Board
- 80 x 25 VDU Interface Board or Teletext VDU Interface Board
- Floppy Disc Controller Board
- One Floppy Disc Drive

SOFTWARE
- Resident DOS
- A DOS utilities disc (see Section 5)

A list of available hardware and software options is given in pare 1 of System 5 System Description.

Note that a different version of DOS is required for different VDUs and different manufacturer's disc drives.

1.3 THE DISC OPERATING SYSTEM

Although the DOS is primarily concerned with disc access, handling and management, it also contains the System 5 Operating System (OS). This controls your screen and printer and provides you with screen editing facilities. Handling of the DOS from the keyboard is described in Sections 2, 3 and 4. Handling of the DOS from programs is described in Section 7.

The DOS recognises your screen and printer control, and screen editing instructions by control key entries from your keyboard (CRTL-A, CRTL-B, etc.).

The DOS recognises your disc access, handling and management instructions by DOS command entries from your keyboard or from within programs.

The DOS also recognises machine code program names. When the file name is entered from the keyboard the DOS loads the named machine code program. The DOS has an auto-start facility so that programs will be loaded and immediately run. The DOS will also execute a series of commands via an Exec file using the Exec command.

2. GETTING STARTED

Several terms will be unfamiliar to you — but don't worry. At first, for the sake of simplicity, we will gloss over some terms that we use but they will be explained later.

Programs, data or character strings (text) are handled by the DOS as Files. When you store a File on disc you must give the program a File Name which, when you subsequently call the program, data or text, the DOS will recognise.

The DOS will also recognise the following commands:

CAT	Catalogue disc
DELETE	Delete file
DIR	Direct Catalogue to memory
DRIVE	Select drive
EXEC	Display or print as text and execute
GO	Load and run machine code program
INFO	Display or print file specification
LOAD	Load file to memory
LOCK	Write Protect a file
MON	Start display or print of file specification messages
NOMON	Stop display or print of file specification messages
OPTION	Enable/disable auto-start facilities
RUN	Run program, file
SAVE	Save file to disc from memory
SET	Set file qualifier
SHUT	Shuts all open files
SPOOL	Opens file for copy
TITLE	Give title to disc
UNLOCK	Remove a file Write Protection
USE	Qualify which group of files that may be accessed

File names can be used directly as Load and Run commands.

You should refer to Section 4 to determine the precise keystrokes for each DOS command, and always follow a command with a RETURN.

2.1 SWITCHING ON

Switch your System on at the wall socket and at the red rocker switch at the lower left hand end of the computer front panel. If you have an Acorn keyboard, the right hand lamp at the front edge of the keyboard will be lit indicating that System power is On.

To enter DOS, press the DELETE key, followed by the BREAK key. If you have a non-Acorn keyboard, press and hold the DELETE key and press the BREAK key. You should now be given the screen message:

```
Acorn Dos
*_
```

The "Acorn Dos" indicates that the DOS is active. The * indicates that the System is waiting for a keyboard entry, and the flashing cursor indicates the print position of the next character to be input.

2.2 THE KEYBOARD

Your keyboard is similar to a typewriter keyboard, though some keys have special functions.

2.2.1 Shift Key

Each character key will enter either an upper case or lower case character. Pressing the Shift key while pressing a character key enters the upper case character.

2.2.2 Shift Lock Key

Pressing the Shift Lock key locks the keyboard into upper case entry. Pressing and releasing the Shift key releases the keyboard to lower case entry. If you have an Acorn keyboard, the centre lamp on the right hand side of the keyboard will light when in shift lock.

2.2.3 Caps Lock (^v) Key

BASIC programs and DOS commands are recognised only when entered as upper case letters and (lower case) numbers. This involves frequent use of the Shift and Shift Lock keys. The Caps Lock mode on the Acorn keyboard allows you to enter DOS commands and BASIC programs with minimum use of the Shift or Shift Lock keys.

If you have an Acorn keyboard, it has a push-on push-off key marked ^v. This key sets the keyboard into the Teletype (CAPS LOCK) mode. The CAPS LOCK mode allows the user to enter capital letters and numbers without having to use the shift key.

Press the ^v, key. The left hand lamp at the front edge of the keyboard should now be lit, confirming that the keyboard is in the CAPS LOCK mode. If the lamp is out press the ^v- key again.

To return to normal operation from CAPS LOCK, press the ^v button.

2.3 KEYBOARD ENTRIES

DOS commands (and BASIC programs) are entered using upper case letters and numbers. Until the RETURN key is pressed corrections can easily be made.

2.3.1 Character Deletion (DELETE Key)

Incorrect characters can be removed by pressing the DELETE key, in which case the character to the left of the cursor is erased and the cursor moved back one space. The cursor may be positioned by pressing and holding the CRTL key while the A key (cursor left) or S key (cursor right) is pressed.

2.3.2 Character Correction

Characters can be corrected by positioning the cursor (see 2.3.1 above) and pressing the character key to be overwritten.

2.3.3 Line Deletion (CRTL-X)

A complete line can be deleted by pressing and holding the CTRL key while the X key is pressed.

2.3.4 Entering A Correct Line (RETURN Key)

When a command or line is correct it is finally entered by pressing the RETURN key.

2.4 LOADING A DISC

Until we deal with formatting in para 2.7 you should use the utilities disc (supplied with your System 5).

Carefully remove your disc from its white protective envelope.

Do not try to remove the disc from its black cover. Open the disc retaining door on the front of the disc drive. If you have two disc drives use the left hand one (Drive *0).* Carefully insert the disc, complete with black protective cover, label outermost, Write Protect Notch uppermost, into the slot in the disc drive. When the disc is fully inserted, close the disc retaining door. The disc is now ready to be read.

2.5 READING A CATALOGUE

In para 2.4 the utilities disc was loaded into Drive 0.

You can find out what files are on a disc by reading its catalogue.

To read a Catalogue in Drive 0 key in:

C A T RETURN

The disc drive lamp which comes on for a short time while the System is accessing the disc confirms that the disc drive is working.

You should now have the Catalogue of the Drive *0* disc on the screen. The Catalogue will depend on the utilities you have been supplied with, but will be something like:

```
Utilities Disc drive 0 qual option 0
: #COMPACT      #COPY
  #COPYF        #DUTY
  #FORM80       #INFALL
*_
```

File Names are listed alphabetically in two columns.

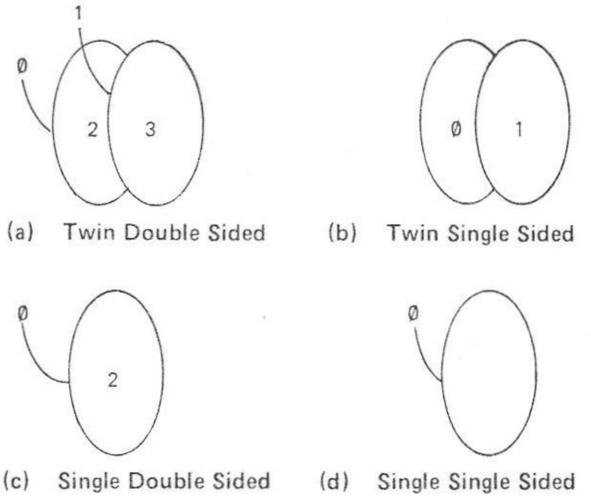
The screen tells us that:

- — The disc TITLE is "Utilities Disc"
- — The disc is located in Drive 0
- — Qualifier space is currently being used *(we* will deal with this later).
- — The disc is set to OPTION *0* (we will deal with this later).
- — No characters to the left of the ":" de-limiters indicate that qualifier space has been allocated to all of the files (again, we will deal with this later).
- — Six Files exist with the file names, INFALL, COPY, COPYF, COMPACT, DUTY and FORM80.

- All six Files are LOCKed (Write Protected by software — more of this later) indicated by the # symbol.

At the end of the CAT listing the DOS* and flashing cursor indicates that the DOS is waiting for a new entry.

The DOS recognises the disc surfaces available to you as drive numbers. The drive numbers available depend on the hardware environment, i.e. the number and type (single or double sided) of disc drives that are fitted. The DOS allocates a drive number to each available surface. So for a system configuration with two double sided disc drives, the left hand disc position has two surfaces number *0* and 2, while the right hand disc position has two surfaces numbered 1 and 3. Figure 2(a) shows the numbering for this configuration. If we have two single sided disc drives fitted, the left hand one is numbered 0 and the right hand one is numbered 1. Figure 2(b) shows the numbering for this configuration. If only one double sided disc drive is fitted, the two surfaces are numbered 0 and 2, see Figure 2(c). One single sided disc drive provides one surface numbered 0, see Figure 2(d).



(a) Twin Double Sided (b) Twin Single Sided

(c) Single Double Sided (d) Single Single Sided

Figure 2. Drive Numbering

Mistakes in numbering can be made when specifying drive numbers in DOS commands. For example, in a twin drive double sided system, COPYing from Drive 0 to 2 would result in transcribing a file from one surface to the other on the same disc instead of, as intended, from disc to disc. It is a good idea to label the drives for the particular configuration so that mistakes of this sort do not happen.

In para 2.5 when *we* read the catalogue, we did not specify from which drive we were to read the Catalogue. Whenever the drive is not specified the current drive (set to 0 after Break) Catalogue is read. We can specify from which drive DOS is to read the Catalogue by the entries CAT0, CAT1, CAT2 or CAT3. When the DOS assumes a parameter through omission it is said that the DOS "defaults" to an assumed parameter. Some other DOS commands default through omission and will be described when they occur.

If the drive number is specified in the CAT command, it sets the current drive to that specified (the DRIVE command can also be used to set the current drive).

2.6.1 Using Drives

When we first entered DOS the current drive was set to Drive 0. This allows file access to Drive 0. To access a drive other than 0 the DRIVE command is used with the format:

```
D R I V E 1 RETURN
```

where "1" is the new current drive.

An alternative way of changing the current drive is with the CAT command. In para 2.5 we read the catalogue of the current drive by not specifying the drive. If we, however, use the CAT command specifying a drive other than the current drive, the current drive is changed to that specified. For example, if the current drive is *0* and we enter:

```
C A T RETURN
```

the catalogue of Drive 1 is displayed and the current drive is changed to Drive 1.

Where a file is loaded and run using the file name alone, that file can be accessed from Drive *0* only.

2.7 FORMATTING A NEW DISC

Before you can use a new disc it must first be formatted. It is recommended that whenever a new box of discs is opened the discs are all immediately formatted.

When you read the catalogue of the utilities disc (in para 2.5), the listing included either FORM40 or FORM80. The version will depend on whether 40 or 80 track disc drives are fitted to your system. The procedure is the same for both FORM40 and FORM80.

To format a new disc, insert the utilities disc into drive 0, check using the CAT command that the file FORM40 or FORM80 is present. Type in the file name given in the Catalogue:

(either FORM40 or FORM 80)

followed by RETURN .

DOS now loads the formatting program into memory at the location given in the File Specification and runs the program.

The message:

```
Do you really want to format drive 0 ?
```

will appear on the screen.

The utilities disc in drive 0 must now be removed and replaced by the new disc to be formatted.

Once the disc to be formatted is in position, the YES response is keyed in.

The formatting starts immediately and the message:

```
Formatting drive 0
```

is displayed. As each track is formatted and verified, the track number is displayed as follows:

```
00 01 02 03 04 05 06 07 08 09
0A 0B 0C 0D 0E 0F 10 11 12 13
14 15 16 17 18 19 IA 1B IC ID
IE IF 20 21 22 23 24 25 26 27
disc formatted
```

If formatting is unsuccessful because the disc is Write Protected the message:

```
Crunch
```

will be displayed,

If formatting is unsuccessful for other reasons the message:

```
YY Failure at XX
```

will be displayed, which means "disc error YY at sector XX on the disc".

By replacing the new formatted disc with an unformatted disc, and typing GO RETURN a series of discs can be formatted.

To format side 2 type DRIVE2 then GO.

With twin drive installations (either single or double sided), the utilities disc can be left in drive 0 and the disc to he formatted may be mounted in drive 1 (single sided)) or drive 1/3 (double sided). In this case, before typing F0RM40 or F0RM80 , type in DRIVE (*drive number*), selecting the drive to be formatted. Whenever a drive other than 0 is selected, any subsequent file execute call will search for that file in drive 0 instead of the drive selected, unless RUN is used.

2.8 LOADING, SAVING AND DELETING FILES

2.8.1 Introduction

The LOAD and SAVE DOS commands are similar in function to the BASIC's LOAD and SAVE command but they are NOT the same. The DOS LOAD and SAVE commands are sometimes known as "Star Load" and "Star Save" to avoid confusion. The DOS DELETE command is unrelated to the Delete key function.

Before you start loading and saving files you need to know what the terms File Name, Start Address, End Address and Execution Address mean.

FILE NAME
The File Name:

- Must be from 0 to 7 characters long.
- Must not include spaces unless File Name is enclosed in quotes (" ").
- Must not contain an odd number of quotes (").
- Trailing spaces are ignored.

You must allocate a File Name to anything you save. You must also give the File Name when you load it.

START ADDRESS
The Start Address is the (start) address in memory to which the file is loaded, or, the (start) address in memory from which the file is saved. We will see in para 2.8.2 that we can either load a file to the same start address from which it was originally saved, or we can load the file to a new start address.

Programs, text or data entered from the keyboard normally have a start address of #2800. (# means hexadecimal).

END ADDRESS
When a file is saved the DOS needs to know the last memory address to be saved. This last memory address saved is known as the End Address.

EXECUTION ADDRESS
The Execution Address is the address in memory at which the program is entered when it is Run. If you fail to give an Execution Address, the program will begin to Run at the Start Address.

2.8.2 Loading a File

In para 2.7 we used a file called FORM40 (or FORM80) simply by entering it's name, followed by a RETURN. This is the method generally used to Load and Run a machine code program from Drive 0.

If you have BASIC and you want to Load and Run it, you should mount the BASIC disc in Drive 0 (the left hand drive, if you have a twin drive System 5), and key in BASIC followed by RETURN from your keyboard. Different versions of BASIC may have different names, so it's a good idea to check the name of your version by first using the CAT command (as described in para 2.5).

To Load and Run a machine code program from a drive other than Drive 0 or a BASIC Program listing, separate Load and Run DOS commands must be used after setting the current drive (see para 2.6.1).

LOAD (File Name) followed by RETURN, loads the named file from disc to the memory. The file is loaded to the same memory location that it was originally saved from.

If you wish to load a file to a memory address other than that from which it was originally saved, you must specify the start address, i.e. LOAD (File Name) (Start Address).

2.8.3 Saving Files

Files are saved using the SAVE (File Name) (Start Address) (End Address + 1) DOS command.

Some program files may require to be executed from some point within the program. In this case the command SAVE (File Name) (Start Address) (End Address + 1) (Execution Address) is used.

2.8.4 Deleting Files

Files may be deleted using the DOS command DELETE (File Name).

When files are deleted, gaps occur, often leading to inefficient disc usage. It is recommended that periodic housekeeping is carried out using the COMPACT utility (described later in Section 5).

2.9 USiNG BASIC

2.9.1 Introduction

We have already mentioned that files can consist of programs, text or data (or a combination of all three). When we speak of programs we mean machine code programs. Your BASIC programs are handled by the DOS as text files. This may seem strange at first, but in Section 3 we will see that the DOS recognises only DOS commands, and not BASIC commands. BASIC, however, is a machine code program and can be loaded and run from Drive 0 by simply entering the file name BASIC (in the same way that we loaded and ran the FORM40 utility in para 2.7).

2.9.2 Loading and Running BASIC

To load and run BASIC you must first insert your BASIC disc in Drive 0 (as described in para 2.4). Check the BASIC File Name using the CAT0 command (as described in para 2.5). You must now load and run BASIC by keying:

```
B A S I C RETURN
```

Your screen will now show the > BASIC prompt (with the flashing cursor) indicating that you can now key in your BASIC Program. Your BASIC Manual tells you how to program in BASIC.

2.9.3 Saving a BASIC Program

To save your BASIC program you use the BASIC SAVE command. This is slightly different than the DOS SAVE command in that the file name must be enclosed in quotes and no space is required after SAVE.

Your BASIC disc should be Write Protected, so remove it and replace it with a formatted disc in Drive 0. If you are going to call your BASIC program "FRED", key in:

```
S A V E " F R E D " RETURN
```

If you have a twin drive system, you can save on to Drive 1 by keying *DRIVE 1 before saving as above.

2.9.4 Loading a BASIC Program

A BASIC program can only be loaded after BASIC itself has been loaded and run (see para 2.9.2).

To load a BASIC program insert your program disc in Drive 0 and key in:

```
L O A D " F R E D " RETURN
```

You can check that it has been loaded by keying in:

```
L I S T RETURN
```

and you can run your BASIC program by keying in:

```
R U N RETURN
```

2.9.5 Returning to DOS and Back

When you are in BASIC you can return to DOS by pressing the BREAK key.

You can subsequently return to BASIC by keying in:

GO C2B2RETURN

Unfortunately, when you re-enter BASIC in this way, you lose the BASIC program that you are using.

BASIC however, has a method of using all the DOS commands rather like subroutines. This method simply uses the DOS command preceded by a "*" in a BASIC program line, e.g.

```
40 *CAT
```

will when running BASIC (at line 40), print the catalogue of the current drive. if you wish to use DOS while entering a BASIC program you omit the BASIC line number.

2.9.6 BASIC Commands Associated with DOS

BASIC has a number of commands which allow you to create and use files for data, text and programs. We will look at these commands and how they are used later.

FOUT

This command creates and opens a file for output and gives you a File Handle which you subsequently use for output. The command is used in the form:

```
30 D=FOUT"FRED"
```

FRED is the File Name, "D" is the File Handle. You will subsequently use "D" in your BASIC program to identify that file in output operations.

if the file "FRED" already exists, DOS will use that file. If the file "FRED" does not exist, DOS will create that file.

FIN

This function opens a file for input and update. The function is used in the form:

```
70 D=FIN"FRED"
```

FRED is the File Name. "D" is the File Handle. You will subsequently use "D" to identify that file in input operations.

PUT

This statement sends a four byte word to a file. The statement is used in the form:

```
40 PUT D,563
```

where "D" is the File Handle (see FOUT above) and 563 is a number to be output to the file.

BPUT

This statement sends a single byte to a file. The statement is used in the form:

```
30 BPUT D,23
```

"D" is the File Handle (previously obtained using the FOUT function). 23 in this case is the value of the byte to be output (any number between 0 and 255 can be represented in a single byte).

SPUT

This statement sends a string to a file. The statement is used in the form:

```
50 SPUT D,A
```

"D" is the File Handle (previously obtained using the FOUT function). "A" is the pointer to the start of the string.

The use of "A" rather than "$A" requires a few words of explanation at this point. Acorn BASIC requires the allocation of a space for a string anywhere safe (remember we said that BASIC programs are really text). This is done using a BASIC system variable called TOP which points to the next free location above your BASIC program text. If we wish to input a string $A with a length of up to (say) twelve characters we could allocate space by the BASIC program line:

```
10 A=TOP; B=T0P+ 13
```

This allows twelve characters plus an "end of string marker".

The BASIC line 50 shown above will send out characters to the file (pointed to by "D"), starting at A and finishing at the "end of string marker". This means that within the limits set by BASIC line 10, whatever length strings we send out to disc, no file space is wasted by gaps when strings are different lengths.

GET

This function reads a four byte word from a file and reads it's value. The function takes the form:

```
80 J=GET D
  OR.
80 PRINT"THE FIRST NO.FROM FRED IS"GET D
```

where "D" is the File Handle. J in this program line points to the value read from the file. Before the GET function is used in a program the file must be opened for input and "D" determined by the FIN function in the form:

```
70 D=FIN"FRED"
```

The GET function is the usual way to input numbers from a data file.

BGET
This function returns a single byte from a file. The function takes the form:

```
80 K=BGET D
OR
80 PRINT"THE FIRST BYTE FROM FRED IS"BGET D
```

where "D" is the File Handle previously determined by the FIN function (see BASIC line 70 above).

SGET
This statement reads a string from a file. The statement is used in the form:

```
90 SGET D,A
```

where "D" is the File Handle (obtained by FIN). "A" is the pointer to the beginning of SA (initialized as described in SPUT).

EXT
This function returns the extent (length) of a file in bytes. The function is used in the form:

```
60 R=EXT D
```

where "D" is the File Handle (obtained by FIN or FOUT) and R is the pointer to the number of bytes in that file.

PTR
As we output to or input from a file DOS keeps a pointer to our position within that file. We can read this position using the PTR function for any file that is open for output, or open for input (FOUT and FIN respectively). The function is used in the form:

```
75 PRINT PTR D
```

where "D" is the File Handle (given by FIN). PTR gives our position in bytes.

In random file operation this DOS pointer can be modified through BASIC using the form:

```
40 PTR D=PTR D+23
```

where "D" is the File Handle obtained by the preceding FIN or FOUT statement. This BASIC program line will move the DOS pointer on by 23 bytes. The next file access will then be 23 bytes further on.

The pointer returns to the start of the file when the file is shut and subsequently re-opened (by FIN or FOUT) .

SHUT
This statement closes input or output files. We can shut individual files by using the form:

```
100 SHUT D
```

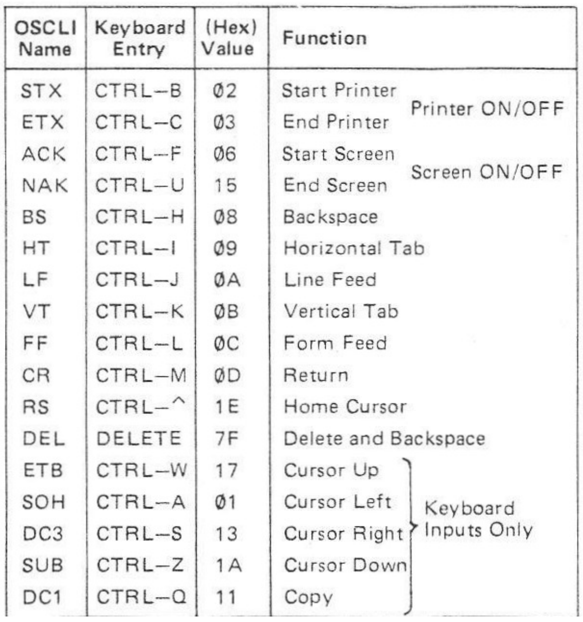
where "D" is the pointer to the file we wish to close.

We can shut all files by using the form:

```
100 SHUT 0
```

2.9.7 Control Codes and BASIC

The Operating System (OS) within DOS has a number of printer and screen control codes as shown in Table 1. All of the OS control codes except COPY and the Cursor controls can be used within BASIC Programs.



OSCLI Name	Keyboard Entry	(Hex) Value	Function	
STX	CTRL–B	Ø2	Start Printer	Printer ON/OFF
ETX	CTRL–C	Ø3	End Printer	
ACK	CTRL–F	Ø6	Start Screen	Screen ON/OFF
NAK	CTRL–U	15	End Screen	
BS	CTRL–H	Ø8	Backspace	
HT	CTRL–I	Ø9	Horizontal Tab	
LF	CTRL–J	ØA	Line Feed	
VT	CTRL–K	ØB	Vertical Tab	
FF	CTRL–L	ØC	Form Feed	
CR	CTRL–M	ØD	Return	
RS	CTRL–^	1E	Home Cursor	
DEL	DELETE	7F	Delete and Backspace	
ETB	CTRL–W	17	Cursor Up	Keyboard Inputs Only
SOH	CTRL–A	Ø1	Cursor Left	
DC3	CTRL–S	13	Cursor Right	
SUB	CTRL–Z	1A	Cursor Down	
DC1	CTRL–Q	11	Copy	

Table 1. Operating System Control Codes

All of the Control Codes can be entered from the keyboard by pressing and holding the CTRL key while the appropriate character key is pressed.

To include a Control Code within a BASIC program the PRINT $ command is used. The usual form is:

```
40 PRINT $#C
    OR
40 PRINT $12
```

where #C is the hexadecimal number corresponding to Form Feed (Clear Screen and Home Cursor) and 12 is its decimal equivalent. These BASIC lines perform the Formfeed (clear the screen, moving the cursor to the top left hand position).

2.10 DISC SECURITY

Programs, text and data stored on disc may represent weeks, or even months of work, and care must be taken not to lose them. DOS has special facilities which prevent accidential overwriting or deletion.

2.10.1 Write Protection Notch

Each disc has a square Write Protection Notch on one edge, see Figure 1. This may be covered to prevent accidental erasure of the disc. Every box of discs is supplied with a number of self-adhesive tabs which, when wrapped around the notch will prevent the disc being overwritten. All utilities should be write protected.

2.10.2 Software Lock

DOS provides you with a command which will prevent accidental erasure of individual files.

The command:

```
LOCK FRED RETURN
```

will write protect the file "FRED". All utilities and programs should be locked.

You can unlock an individual file by the unlock command:

```
UNLOCK FRED RETURN
```

which will remove the write protection of the file " FRED".

2.10.3 Back-Up Discs

It is important to keep back-up discs of all utilities, important programs and data files.

2.10.4 Transaction File Organization

Data Processing Centres using large main frame computers usually use a disc or tape management system called Transaction File Organization. This can sometimes be used in System 5 applications to add a measure of security to large data files which are subject to frequent change. A typical application is in a stock control system where items are issued or received continually, bui where re-ordering occurs only periodically (say once a day).

The principle is that the large database file (stock, re-order codes, suppliers, minimum stock levels etc) is kept locked during normal transaction periods. All issues and receipts of stock are handled by a simple (and fast) program which creates and adds to a file (on a separate disc) containing only changes or transactions.

When issues and receipts of stock cease (say lunchtime), another program is entered which uses the transaction file to update the large database file and, subsequently, to provide re-ordering printouts etc.

The large stock database file (which may take days or weeks of physical stocktaking to re-create) is thus protected for much of the time.

A secondary advantage of Transaction File Organization is that the lengthy program searches and sorts associated with re-ordering are avoided allowing the user immediate access for input of issue and receipt data.

2.10.5 Ancestral File System

A disadvantage of the Transaction File Organization outlined in para 2.10.4 is that if a system failure occurs while the stock database is being updated, the database could be corrupted or lost. The Ancestral File System is a method of minimising the effect of such a catastrophe.

In principle, the Ancestral File System uses three database and three transaction files, all on separate discs, labelled Grandfather, Father and Son. At each update period, the Son database and transaction files are used to create a new database file on the (old) Grandfather disc, Thus the (old) Grandfather becomes the (new) Son, the (old) Son becomes the (new) Father and the (old) Father becomes the (new) Grandfather. The transaction and database files (now Fathers) used to create the Son database file are kept together in a safe place and, should a catastrophe occur, can be used to re-create the Son database file.

3. DOS FEATURES

3.1 TRACKS, SECTORS AND BYTES

Information is written onto the disc in concentric circles, called tracks. Your System 5 will be equipped with either 40 or 80 track Floppy Disc Drives. Each track is divided into 10 sectors and each sector is divided into 256 bytes. Each byte corresponds to one character. Floppy Disc Drives may be single or double sided types.

3.2 FORMATS

3.2.1 Disc Format

Because the user allocates file names as character strings, and the computer recognises tracks and sectors by numeric representation of the file's physical location, an area on the disc is allocated for an index or Catalogue seen by typing CAT. This area is always located in Sectors 0 and 1 on disc. It supplies the DOS with a catalogue to locate a file quickly. The user may allocate a TITLE to a particular disc surface and the Title is stored in the Catalogue.

Each File Name in the Catalogue has it's position on disc stored. Other file parameters, some allocated by the user, and some determined by the DOS, are entered against each file in the Catalogue. Among the user defined parameters stored against each file in the Catalogue are:

- LOCK, (#), which is a software write protect facility.

- Qualifier, which is associated with the SET *(qualifier)* DOS command. This allows, in association with the USE *(qualifier)* DOS command, division of a TITLE (disc surface) into a number of groups of files.

- Start Address. This allows the DOS command LOAD (*file name)* to specify loading to memory location specified by the file. This parameter is originated by the SAVE *(file name) (start address) (end address)* DOS command.

- Execution Address. This parameter originated by the SAVE *(file name) (start address) (execution address)* DOS command is mainly applicable to program files used in association with the OPTION 2 facility and for command files, e.g. BASIC.

The length parameter is worked out by the DOS itself when the DOS SAVE command is executed. This parameter gives the number of bytes contained within the file.

3.2.2 Catalogue Format

When the disc is started the Catalogue is copied into memory, and this Catalogue is assumed valid while the disc is rotating, Care must be taken with some types of drive (which spin when a disc is inserted) that the disc has stopped before using it. Reading successive files therefore requires as little head movement as possible. Changes made to the Catalogue cause it to be written out to the disc. The Catalogue and file buffers are stored in Random Access Memory at locations 2000 to 27FF (hex) and this RAM must be present in the system.

3.3 THE COMMAND LINE INTERPRETER

It is useful, though not essential, to understand how the OS and DOS work. The two main inputs to the OS are from the keyboard and the disc. The input strings (of characters and codes) from the keyboard are put, first of all, into a buffer store (part of the computer's memory). Whenever a string ends with a Carriage Return code a special OS program called the Command Line interpreter is automatically run. The Command Line interpreter or OSCLI for short, does exactly what its name says — it looks for and interprets a command line. But how does the OSCLI know that a string in the buffer store is a command?

The OSCLI, when it looks at the string in the buffer store, compares it with a Command List (or table) in the computer memory (DOS ROM). When comparison is found, an entry against that command code vectors the program execution to the subroutine in OS (or DOS) that executes the command. If the string is not found in the DOS ROM, the DOS looks it up on disc (and will load and run the file so named). if the string is still not found an error message occurs. Figure 3 illustrates the basic principles of operation of a Command Line Interpreter.

When the system is powered up or reset (using the BREAK key), the vectors are loaded from ROM into RAM at start address 0200 (hex).

All keyboard entries to the OS are buffered at RAM address 0100 (hex). BASIC, however, may put them elsewhere. The Operating System subroutine OSCLI, on recognising and identifying a valid OS command in the buffer, vectors the program execution to the appropriate subroutines to implement the command keyed in. The commands must be terminated by a RETURN (CR) code, 0D (hex).

The System 5 Disc Operating System (DOS) extends the Command List to provide the disc and file handling facilities that are available. Many of the utility programs available use the Command Line Interpreter in their operation.

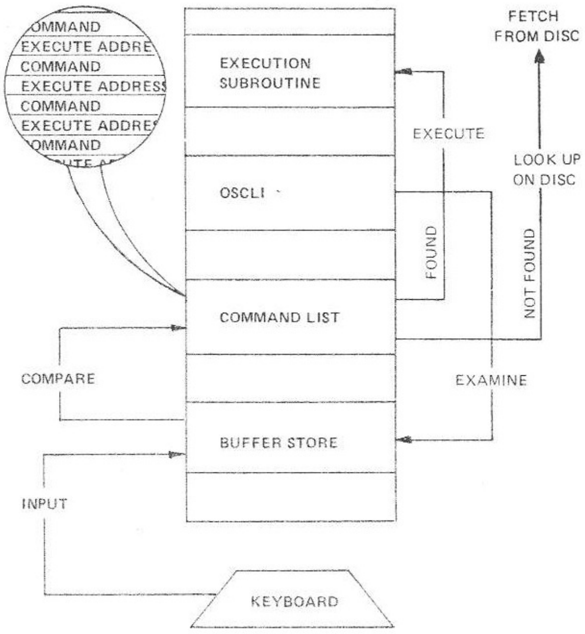


Figure 3. Command Line Interpreter

3.4 KEYBOARD CONTROL FUNCTIONS

The following facilities are available from the keyboard or from BASIC programs (see Table 1).

3.4.1 Start Printer, STX [CTRL-B, 02 (hex)]

This code, which is not sent to the printer, starts the printer output stream, All further output is sent to the printer as well as the screen until receipt of an ETX (End Printer) code.

3.4.2 End Printer, ETX [CTRL-C, 03 (hex)]

This code ends the printer output stream.

3.4.3 Start Screen, ACK [CTRL-F, 06 (hex)]

This code starts the output stream to the screen.

3.4.4 Backspace, BS [CTRL-H, 08 (hex)]

This code moves the cursor back one position.

3.4.5 Horizontal Tab, HT [CTRL-I, 09 (hex)]

This code moves the cursor forward by one position.

3.4.6 Line Feed, LE [CTRL.-J, 0A (hex)]

This code moves the cursor down one line.

3.4.7 Vertical Tab, VT [CTRL-K, 08 (hex)]

This code moves the cursor up one position.

3.4.8 Form Feed, FE [CTRL-L, 0C (hex)]

This code clears the screen and moves the cursor to the top left hand corner of the screen.

3.4.9 Return, CR [CTRL-M, 0D (hex)]

This code moves the cursor to the start of the current line.

3.4.10 End Screen, NAK [CTRL-U, 15 (hex)]

This code ends the output stream, to the screen. The only code recognised in this condition is ACK.

3.4.11 Home Cursor, RS [CTRL-^,1E(hex)]

This code moves the cursor to the top left hand corner of the screen.

3.4.12 Copy, DC1 [CTRL-Q, 11 (hex)]

This code copies the characters above the cursor to a new line.

3.5 ABBREVIATED ENTRIES

DOS recognises abbreviated entries for many DOS commands. Section 4 lists, against each command, the abbreviated entry, if any.

3.6 PRINTER, SCREEN AND KEYBOARD STREAMS

DOS sees the keyboard as an input stream and the screen and printer as output streams.

3.6.1 SPOOL Command

The SPOOL command opens a file for output and copies all output stream onto that file. The file is closed by the SHUT command. The SPOOL command format is:

```
SPOOL FRED RETURN
```

where "FRED" is the user allocated file name to the SPOOL file.

3.6.2 EXEC Command

The EXEC command reads the named file and displays the file as characters. If the stream contains valid DOS commands they are executed. If BASIC is entered via a DOS command in the EXEC file, subsequent BASIC commands are executed. This allows a series of BASIC programs, and/or data files to be handled consecutively,

The format of the EXEC command is:

```
EXEC FRED RETURN
```

where "FRED" is the name of the file to be executed.

3.7 AUTO-START (OPTION AND BOOT)

In para 2.5 when we read the catalogue of a disc, one of the parameters associated with the disc was Option. The DOS Option feature allows a series of files to be loaded and/or run sequentially by simply pressing and holding the SPACE bar while pressing the BREAK key. This feature is useful where BASIC programs are being used, especially where the BASIC programs require access to data or text files. The Boot file (para 3.7.2) must exist on Drive 0 to be used on auto-start,

3.7.1 OPTION Command

The Option of a disc in the current drive is set using the OPTION command format:

```
OPTION 3RETURN
```

where "3" is the Option to which the disc in the current drive is set.

The options available are as follows:

- Option 0 : Do not do anything
- Option 1 : Load the file named BOOT
- Option 2 : Run the file named BOOT
- Option 3 : Execute the file named BOOT

3.7.2 BOOT File

The Boot file provides the user with a facility which may be used in many applications. Essentially it allows you, in association with Option, to execute a series of DOS commands by simply pressing and holding the SPACE bar while pressing the BREAK key. If BASIC is present on disc, it can be loaded and run via Boot, and a series of BASIC Programs loaded and run via BASIC commands within the Boot file.

CREATING A BOOT FILE
To create a Boot file you will need (after loading and running BASIC) to enter and run a simple BASIC program as follows:

```
19 A=FOUT "BOOT"
20 INPUT $TOP
30 IF $TOP = "END" GOTO 60
40 SPUT A $TOP
50 GOTO 20
60 SHUT A
70 END
SAVE "INBOOT"
RUN
```

Line 10 opens a file named "BOOT" for output and allocates its pointer "A". Line 20 allows you to input a command which is stored immediately above your BASIC program "INBOOT". Line 30 allows you to terminate Boot file creation by typing "END". Line 40 puts the command (from $TOP) into the Boot file (pointed to by "A"). Line 60 closes the Boot file. The " SAVE" line saves the Boot creation program with the file name "INBOOT" so you can use it again.

You will now have the "?" prompt for command input on screen. A series of commands can now be entered from the keyboard, each terminated by a RETURN. When your Boot file is complete typing "END" will close the Boot file.

The Boot file:

```
? LOAD BASIC
? RUN BASIC
? LOAD FRED
? RUN FRED
```

will not work but the following Boot file will:

```
? LOAD BASIC
? RUN BASIC
? LOAD "FRED"
? RUN
```

Once we run BASIC only commands in BASIC syntax will be executed.

As an example of the use of BOOT, we will write two simple programs called FRED and HARRY and create a Boot file to run them successively via the BREAK key. First of all press and hold the DELETE key while the BREAK key is pressed to enter DOS and then enter:

```
BASIC
10 PRINT ' "THIS SHOWS THAT FRED HAS RUN" '
20 END
SAVE "FRED"
10 PRINT ' "THIS SHOWS THAT HARRY HAS RUN" '
20 END
SAVE "HARRY"
```

You have now loaded and run BASIC and created the files FRED and HARRY. If you have not already done so enter the INBOOT program and load and run it to create the Boot file by entering:

```
LOAD BASIC
RUN BASIC
LOAD "FRED"
RUN
LOAD "HARRY"
RUN
PRINT ' "BOOT FILE HAS NOW BEEN EXECUTED" '
END
```

By pressing and holding the DELETE key while the BREAK key is pressed you will re-enter DOS, and we must now set the Option to 3 by keying:

```
OPTION 3
```

Now, by pressing and holding the SPACE bar while the BREAK key is pressed at any time, the Boot file will be executed, and with our example, the screen will display:

```
* LOAD BASIC
    BASIC C000 C2B2 01000 003
* RUN BASIC
    BASIC C000 C2B2 01000 003
>LOAD "FRED"
    FRED 2800 C2B2 00032 002
> RUN
  THIS SHOWS THAT FRED AS RUN
> LOAD "HARRY"
    HARRY 2800 C2B2 00033 013
>RUN
  THIS SHOWS THAT HARRY HAS RUN
>PRINT ' "BOOT FILE HAS NOW BEEN EXECUTED" '
BOOT FILE HAS NOW BEEN EXECUTED
```

Should you wish to Break without executing Boot, you must press and hold the DELETE key while the BREAK key is pressed.

To disable the auto-start, re-enter DOS and enter OPTION 0. Alternatively, if you are in BASIC, enter *OPTION 0.

3.8 EXEC COMMAND

Any file created in the same way as Boot (para 3.7.2) can be executed by the DOS command EXEC. If you still have your Boot file on disc set Option to 0 (see para 3.7.1) and enter EXEC BOOT (from DOS) or (from BASIC) enter *EXEC BOOT. The Boot file will now run with the same screen display as in para 3.7. 2. The EXEC format is given in para 3.6.2.

3.9 QUALIFIERS

In para 2.5 the CAT command display showed that qualifier "space" was currently being used, and that all files had a space to the left of the colon (:) indicating that all files were allocated the qualifier "space".

The qualifier facility is a useful aid to file handling. It is implemented by means of the SET and USE DOS commands.

The SET command sets the current qualifier to any character until either the BREAK key is pressed, or the SET command is again used. Pressing the BREAK key sets the current qualifier to "space". Files are saved with the current qualifier (except when a file is saved immediately after a USE command). Note that no space is allowed between SET and the character, e.g. SETA is acceptable while SET A is not.

The USE command allows the next file operation to use other than the current qualifier. After the next file operation the qualifier reverts to the current qualifier.

The most common use of qualifiers is to separate files according to their content, e.g. qualifier B for BASIC programs and qualifier D for data files.

There may be two (or more) separate files with the same file name providing they are in separate qualifiers. Hence with the Assembler you can have a source file called, say, UADE01 in qualifier "space" which you will assemble into an object file with the same name providing it has a different qualifier.

3.10 SPOOLING

The SPOOL command will open a file for copying. To demonstrate you should enter:

```
SPOOL SPFILE
```

or if you are in BASIC:

```
*SPOOL SPFILE
```

For demonstration purposes, to put something in the file named SPFILE you could EXEC BOOT, see para 3.8 (or *EXEC BOOT if you are in BASIC).

To disable the SPOOL facility (closing the file) you must enter SHUT (or *SHUT SPFILE from BASIC).

You will need a small BASIC program to read your SPFILE, so re-enter BASIC and key in this program:

```
10 D=FIN "SPFILE"
20 C=EXTD
30a IF C=PTRD GOTO b
40 A=BGET D
50 IF A=10 PRINT $13
60 PRINT $A
70 GOTO a
80b SHUT D
90 END
```

Line 20 uses the EXT command to determine the length of the file. Line 30 checks to see if the end of file has been reached before reading in a character from the file. Line 50 tidies up the screen format, adding a Return to each Line Feed. Saving this file with the file name "SPREAD" will allow you to use it again. By changing the file name in Line 10 you can read any file.

3.11 DOS POINTER

When accessing files DOS maintains a pointer to the current access position within the file. The pointer can be read or changed by the BASIC PTR function. We used PTR in the SPREAD program (Line 30) in para 3. 10 to exit from a loop when the end of file was reached. The pointer is set to "0" whenever a file is opened (FIN, FOUT' SPOOL, etc). We will look at the DOS Pointer again in para 6.1.

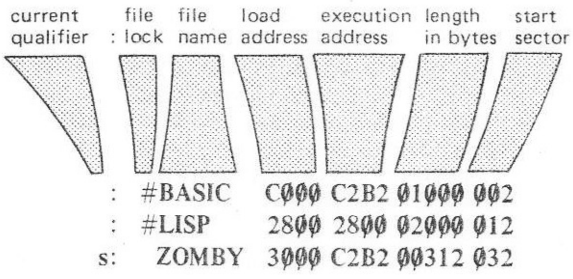
3.12 DISC TITLES

We can give the disc in the current drive a title using the command TITLE followed by a space and the disc name we are allocating (in BASIC the command is preceded by a *). Up to 13 characters may be used for a title.

3.13 INFORMATION ON FILES

3.13.1 INFO Command

We can read information on any file in the current drive using the INFO command (followed by a space and the file name). The information displayed is in the following form:



```
  :  #BASIC   CØØØ C2B2 Ø1ØØØ ØØ2
  :  #LISP    28ØØ 28ØØ Ø2ØØØ Ø12
s:    ZOMBY   3ØØØ C2B2 ØØ312 Ø32
```

3.13.2 MON and NOMON Commands

The MON command turns on a message system which displays a file's information at each file access.

The NOMON command turns off the message system which was originally turned on by the MON command.

3.14 MACHINE CODE PROGRAM EXECUTION

To execute a machine code program the GO command may be used (followed by a space and the execution address). We used this command in para 2. 9.5 to reenter BASIC at C2B2 from DOS.

If the execution address is omitted the last known execution address will be used. Note that the current execution address is destroyed by CAT, and INFO does not set the execution address.

If a machine code program is on Drive 0 you can use the file name directly (as we called FORM40 and BASIC).

4. DOS CO#MMANDS AND ERROR MESSAGES

In the following commands, error and other messages, text, data and programs are said to be " displayed". They can, in fact be displayed on screen or printed or both, subject to the preceding Operating System control codes entered, see Table 1 in para 2. 9.7.

File names and disc titles must be enclosed in quotes unless no spaces exist in file name. Only even numbers of quotes (or non at all) are allowed in file names or disc titles. Up to 13 characters are allowed in disc titles, and up to 7 characters are allowed in file names.

Legal drive numbers are dependent on the disc drive configuration, see pare 2.6, but illegal drive numbers will simply cause the system to wait for the drive which is not there. Abbreviated command formats are shown following the full command formats.

4.1 DOS COMMANDS

CAT *(drive number)*

. *(drive number)*

This command causes the Catalogue of drive *(drive number),* (or the current drive if *(drive number)* is omitted) to appear on the screen.

A typical Catalogue will look like:

```
*CAT0
Basic disc v1 drive 0 qual s opt 0
:   #BASIC        #LISP
s:   ZOMBY
```

The title of the disc is Basic disc v1 ; we are currently using drive 0 and qualifier s. The disc option is 0 (no auto-start features). Two files have been saved in qualifier "space", both of which have been locked to prevent careless deletion. One file has been saved in qualifier "s" and this has been left unlocked. The Catalogue is sorted alphabetically by qualifier and file name when it is output. The *(drive number)* can be omitted, or it must be 0, 1, 2 or 3. If outside the range 0 to 3 then the message:

```
Drive ?
```

will he displayed.

If the *(drive number)* is specified, the default drive for subsequent commands is changed to that specified.

DELETE *(file name)*

DE. *(file name)*

This command deletes the *(file name)* in the current qualifier from the current disc's Catalogue. If the entire disc is Write Protected a

```
Disc prot
```

message is displayed. If the file is not found a

```
File ?
```

message is displayed. !f the file is Locked a

```
File prot
```

message is displayed.

DIR *(drive number)*

D. *(drive number)*

This command causes the Catalogue of the drive *(drive number)* to be loaded into memory at hex start address 2000. The command is often used to wait until completion of the previous operation. The drive number can be omitted, or it must be in the range 0 to 3. if outside the range then the message:

Drive ?

will be displayed.

DRIVE □ *(drive number)*

DR. □ *(drive number)*

This command sets the current drive to *(drive number)* where *(drive number)* is either in the range 0 to 3, or is omitted (for compatibility with CAT or DIR). If *(drive number)* is outside the range the message:

Drive ?

will be displayed. If *(drive number)* is entered as a multi-character then the message:

Syntax ?

will be displayed. Drive 0 is set on BREAK.

EXEC □ *(file name)*

E. □ *(file name)*

This command reads and displays as characters the bytes from *(file name)* in the current qualifier on the current disc. If *(file name)* is not found a

File ?

message is displayed. After all the bytes have been read, *(file name)* is automatically closed.

GO □ *(execution address)*

This command causes the machine code subroutine at *(execution address)* to be entered. If *(execution address)* is not given, the last known execution address is used. Note that the current execution address is destroyed by CAT, and INFO does not set the execution address.

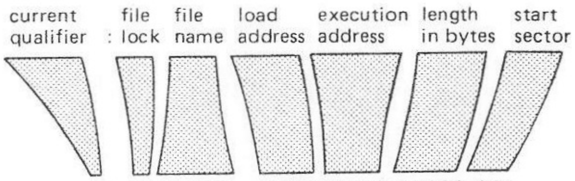
INFO □ *(file name)*

I. □ *(file name)*

This command produces information about the *(file name)* in the current qualifier on the current disc. If the file is not found a

File ?

message is displayed. The information displayed is in the following form:



```
   :  #BASIC   CØØØ C2B2 Ø1ØØØ ØØ2
   :  #LISP    28ØØ 28ØØ Ø2ØØØ Ø12
  s:   ZOMBY   3ØØØ C2B2 ØØ312 Ø32
```

LOAD *(file name)* □ *(start address)*

L. □ *(file name)* □ *(start address)*

This command loads the file *(file name)* on the disc in the current drive from the current qualifier into the memory at *(start address).* The *(start address)* may be omitted when the file's own *(start address)* is used. If the file is not found a

File ?

message is displayed.

LOCK □ *(file name)*

LO. □ *(file name)*

This command Write Protects an individual file.

The command locks the *(file name)* in the current qualifier on the current disc. If entire disc is protected a

Disc prot

message is displayed. If the file is not found a

File ?

message is displayed.

MON □

M. □

This command turns on a message system which displays a file's information at every file access.

NOMON □

N. □

This command turns off the message system which was originally turned on by the MON command.

OPTION □ *(option)*

O. □ *(option)*

This command sets the option of the disc in the current drive to the number *(option).* If the entire disc is protected, a

Disc prot

message is produced. The option enables auto-start use of the file BOOT in qualifier "space" on drive 0 when the system is reset by the BREAK key. The auto-start may be totally defeated by pressing the DELETE key while the BREAK key is pressed, and is subsequently enabled by pressing "space" while the BREAK key is pressed. The possible modes are:

— Option 0 : do not do anything
— Option 1 : load the file BOOT
— Option 2 : run the file BOOT
— Option 3 : exec the file BOOT

In option 0, the system will not mind if BOOT is not present, in the other modes, a

File ?

message will be displayed if BOOT does not exist when the BREAK key is pressed.

RUN□ *(file name 1)* □ *(file name 2)*
R.□ *(file name 1)* □ *(file name 2)*
(file name 1) *(file name 2)*

The *(file name 2)* is optional.

This command loads the file *(file name 1)* on the disc in the current drive from the current qualifier into memory at the *(start address)* of the file. The optional *(file name 2)* is turned back into the original string form and stored in memory at hex start address 0140, terminated by a RETURN (CR character).

SAVE□ *(fn)* □ *(sa)* □ *(ea+1)* □ *(xa)*

where *(fn)* = *(file name)*
(sa) = *(start address)*
(ea+1) = *(end address + 1)*
(xa) = *(execution address)*

The *(execution address)* is optional. When omitted it defaults to the start address.

This command saves the block of memory between *(start address)* and *(end address + 1)* to the file *(file name)* in the *(current qualifier)* of the Catalogue. If the entire disc is Write Protected a

Disc prot

message is displayed. If the *(file name)* is Locked a

File prot

message is displayed. If *(file name)* exists and is not Locked, or the disc is not Write Protected, the *(file name)* is deleted. Starting at the extreme outside edge of the disc (track 0), a gap is searched for which is large enough to contain the block to be saved. If a large enough block cannot be found a

Disc full

message is displayed. If there are already 31 files on the disc Catalogue a

Full

message is displayed.

SET *(current qualifier)*

This command sets the current qualifier to *(current qualifier),* where *(current qualifier)* can be any character. All following file access will use only the *(current qualifier)* portion of the Catalogue. Qualifier " space" is set on keying BREAK.

SHUT□

This command closes all files for input or output.

SPOOL□ *(file name)*

This command opens file and copies all characters going out onto it. The Spool facility is stopped by the SHUT command.

TITLE□ *(disc name)*
T.□ *(disc name)*

This command sets the title of the disc in the current drive to the first 13 characters in *(disc name),* filling with spaces if there are fewer than 13 characters. It is possible to include Form Feed (CTRL—L) at the start of a Title, so that Catalogues appear at the top of the screen. If the entire disc is Write Protected, a

Disc prot

message is displayed.

UNLOCK□ *(file name)*
U.□ *(file name)*

This command unlocks the *(file name)* in the current qualifier on the current disc. If entire disc is protected a

Disc prot

message is displayed. If the file is not found a

File ?

message is displayed.

USE *(qualifier)*

This command allows the next file operation to use the *(qualifier)* portion of the Catalogue. After the file operation is complete, the previous *(current qualifier)* will be made current again. If an error occurs in the file operation, the qualifier does not immediately revert so that the operation can be repeated. To force reversion after an error, use a MON or NOMON command. Two successive USE commands will result in the loss of the original *(current qualifier).*

4.2 ERROR MESSAGES

Crunch : An attempt has been made to format a Write Protected disc.

Disc Full : A block large enough to hold the file cannot be found. Periodic housekeeping with the COMPACT utility will minimise the occurrence of this error.

Disc Prot : A Write operation has been attempted to a Write Protected disc.

Drive ? : Drive number is incorrectly (or not) specified.

File ? : File name cannot be found.

File Prot : A Write operation has been attempted to a Write Protected file.

Full : 31 files alread-y exist on the current disc.

Syntax ? : Command is recognized but Syntax error exists.

Disc error 08 : During a disc Read operation a clock bit was missing.

Disc error 0A : During a disc transfer the processor did not respond fast enough (probably due to a faulty Floppy Disc Interface Board)

Disc error 0C : The CRC (cyclic redundancy check) derived from the ID data read back, differed from that which was originally loaded to the disc.

Disc error 0E : The CRC derived from the data read back on a disc read differed from that loaded to the disc.

Disc error 10 : During a transfer the disc stopped rotating (probably due to a badly inserted disc).

Disc error 14 : The Floppy Disc Controller Interface Board failed to find Track 0. This error usually results in attempting to access an unformatted disc.

Disc error 18 : The Floppy Disc Controller Interface Board failed to find the required sector. This error usually results from either using an unformatted disc, or the disc being corrupted by magnetic fields etc.

4.3 FILE SPECIFICATIONS

The full File Specification is shown in Figure 4.

4.3.1 File Name

The File Name:

- Must be from 0 to 7 characters long.
- Must not include spaces unless File Name is enclosed in quotes (" ").
- Must not contain an odd number of quotes (").
- Trailing spaces are ignored.

The File Name is specified by the user when a file is created. Files are c+reated by:

- The DOS SAVE command.
- Calling the OSFIND routine via assembler (see para 7.1 and 7.3) to perform an open-foroutput.
- Calling the OSFIND routine via the BASIC FOUT function (see para 6.1), e.g.

  X = FOUT "FRED" in 4K BASIC

The File Name can only be changed by execution of the DOS command LOAD *(old file name)* followed by the SAVE *(new file name)* command.

4.3.2 Qualifier

The Qualifier:

- Must be any single character (or space).

All files are saved with a particular qualifier. The qualifier serves to divide the disc Catalogue into sections. The most common use of the qualifier is to

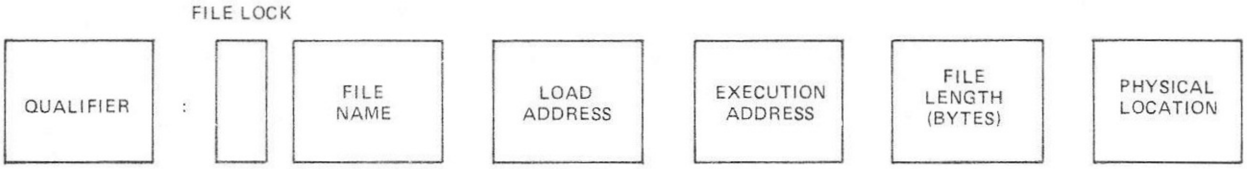


Figure 4. Full File Specification

separate the files according to their content, e.g. qualifier B for BASIC programs and qualifier D for data files.

There may be two (or more) separate files with the same File Name providing they are in separate qualifiers. Hence with the assembler the user may have a source file called UADE01 in qualifier "space" which will be assembled to give an object file with the same name providing it has a different qualifier.

All file operations use the currently selected qualifier. After a reset, the qualifier is set to "space". The DOS provides two commands to change the current qualifier:

SET(qualifier)	sets the qualifier to *(qualifier)* until another command is used to change it.
USE(qualifier)	sets the qualifier to *(qualifier)* for the next file operation only, after which it reverts to its previous value.

Note that there must be no space between the command and the *(qualifier).*

4.3.3 Load Address

The Load Address specifies the address in memory to which the file is loaded when the DOS command LOAD *(file name)* is executed. The Load Address is a four digit hexadecimal number.

The Load Address is defined when the DOS command SAVE *(file name) (load address)* is executed.

When the DOS command LOAD *(file name) (load address)* is executed, the *(load address)* in the File Specification defaults to the *(load address)* specified in the LOAD command, but the *(load address)* in the File Specification is not changed. Note that files created by OSFIND have a load address of 0000.

4.3.4 Execution Address

The Execution Address specifies the address at which execution of a program in a File will commence when the DOS command RUN *(file name) is* executed. The Execution Address is a four digit hexadecimal number.

The Execution Address is specified when the DOS commands SAVE *(file name) (start address) (end address + 1) (execution address)* is executed. if the *(execution address)* Field is omitted in the SAVE DOS command, the Execution Address in the File Specification defaults to the *(start address).*

4.3.5 File Length

The File Length specifies the number of bytes in the file and is a five digit hexadecimal number.

The File Length is calculated by the DOS when the command SAVE *(file name) (start address) (end address +1) is* executed.

4.3.6 Physical Location (Position on Disc)

The Physical Location is a three digit hexadecimal number specifying the physical location of the start of the file on disc, i.e. consecutive (through all tracks) sector number. The Physical Location is allocated by the DOS when the file is created.

The DOS allocates a position on disc when the SAVE DOS command is executed.

5 UTILITI ES

A Utility Mini-Floppy Disc will be supplied with your System. The programs may change with different system variants, so your Utility Disc may not be compatible with other systems.

If a disc with the utilities on it is in drive 0 typing just the utility name will cause the program to be loaded and run, thus the utilities appear to operate as additional DOS commands. The utilities disc may have a BOOT file which will cause a description of the disc programs to appear on the screen at system start up.

In a single drive system the disc with the utilities on will need to be removed and the disc to be operated upon inserted, the programs contain a waiting state for this operation where appropriate. Dual drive, single sided disc systems will usually have the utilities on a disc in drive 0 and the user disc in drive 1. Dual drive double sided disc systems will usually have the utilities on a disc in drive 0 and the user disc in drive 1/3. Select drive 1/3 using the CAT1 (or 3) or DRIVE1 (or 3) as appropriate before running the utility. The original utility disc supplied with a system is best kept Write Protected so that it cannot be accidentally formatted or copied on to.

5.1 INFALL

INFALL uses the INFO command in the DOS to give load address, run address and disc sector information on all the files on a disc.

5.2 COPY

COPY is for use on dual drive systems, it is loaded on entering COPY *(source drive) (destination drive)* and the system then stops. Pressing the space bar will cause a complete copy to occur.

5.3 COMPACT

After saving and then deleting files on a disc unused sectors will appear where a file was deleted and no files of the same length have since been saved. Copies one file after the other into RAM and then re-saves them with no gaps between them. Waits for the space bar to be pressed.

"SAVE"d files use the first available space on disc which is large enough to accommodate the file. Files created use the space occupied by a previous file of the same name (if it existed). If the spare space on disc becomes split up into small parts, attempting to create a large file will cause a "no room" error. To collect all the free space together, the COMPACT utility should be used.

5.4 COPYF

COPYF copies particular files from one drive to another. The source and destination drives are specified as in COPY.

5.5 DUTY

DUTY copies and compacts a disc from one drive to another.

5.6 FORM40, FORM80

FORM (formats) initializes new discs with the track and sector format. The disc requiring formatting should be inserted and checked with the CAT command. A description of Formatting procedure is given in para 2.7.

5.6.1 Soft Sectoring

TRACKS, SECTORS AND BYTES
Information is written onto the disc in concentric circles, called tracks. Your System 5 will be equipped with either 40 or 80 track Floppy Disc Drives. Each track is divided into 10 sectors and each sector is divided into 256 bytes. Each byte corresponds to one character. Floppy Disc Drives may be single or double sided types.

DISC FORMAT
Because the user allocates file names as character strings, and the computer recognises tracks and sectors by numeric representation of the file's physical location, an area on the disc is allocated for an index or Catalogue seen by typing CAT. This area is always located in Sectors 0 and 1 on disc. It supplies the DOS with a catalogue to locate a file quickly. The user may allocate a TITLE to a particular disc surface and the Title is stored in the Catalogue.

Each File Name in the Catalogue has it's position on disc stored. Other file parameters, some allocated by the user, and some determined by the DOS, are entered against each file in the Catalogue. Among the user defined parameters stored against each file in the Catalogue are:

- LOCK (#), which is a software write protect facility.
- Qualifier, which is associated with the SET *(quainter)* DOS command. This allows, in association with the USER *(qualifier)* DOS command, division of a TITLE (disc surface) into a number of groups of files.
- Start Address. This allows the DOS command LOAD *(file name)* to specify loading to memory location specified by the file. This parameter is originated by the SAVE *(file name) (start address) (end address)* DOS command.
- Execution Address. This parameter, originated by the SAVE *(file name) (start address) (end address +1) (execution address)* DOS command is mainly applicable to program files used in association with the OPTION 2 facility, and for command files, e.g. BASIC.

The length parameter is worked out by the DOS itself when the DOS SAVE command is executed. This parameter gives the number of bytes contained within the file.

The DOS uses these above mentioned file parameters in its' various operations. Because the Catalogue has a fixed field format, the number of files on each disc surface is limited to 31. Similarly, File Names are a maximum of 7 characters and Titles are truncated to 13 characters. We will later see how these parameters can be displayed by the DOS commands INFO and MON.

To allow the computer to access the various tracks and sectors on the disc, after determining the relationship between the File Name and the position on disc, each track must be divided into sectors, identified, and the beginning and end of the sector marked. Figure 5 shows the format of each sector.

7 Bytes	17 Bytes	259 Bytes	27 Bytes
SECTOR IDENTIFICATION (ID)	POST ID GAP	SECTOR DATA FIELD	POST DATA GAP

Figure 5. Sector format.

Before a disc is used for the first time it must be formatted. This is done with a utility called FORMat. If your System 5 is provided with 40 track disc drives you will have been supplied with the FORM40 version, bit if you have 80 track disc drives you will have the FORM80 version. Use of these utilities is described in para 2.7.

The FORM80 and FORM40 utilities structure the Catalogue portion of the disc (Sectors 0 and 1), and records the sector IDs in all sectors. In subsequent operations, the DOS provides random File access by looking up the File Name in the Catalogue to determine the physical location and it can then immediately access that particular file without having to serially search until the file is found.

6. SEQUENTIAL FILE HANDLING

In para 3.11 we described the DOS file pointer. The pointer is used extensively within user programs for handling sequential files.

6.1 POINTER OPERATION

Whenever a file is opened for input or output DOS creates a pointer to the current access position (in bytes) to the file, and initializes this pointer to 0. The BASIC line

```
20 D= OUT "NED"
```

will open the file "NED" and initialize the pointer for " NED" to *0* which could be read by the BASIC program line

```
90 PRINT PTRD
```

Add the program line

```
100 END and RUN
```

and the pointer will be displayed on your screen. if we add a few program lines to output 4 bytes at a time we can show the pointer at each file access.

```
30 INPUT "ENTER" B
40 PUT D,B
RUN
```

In response to the "?" key in any number (or character) and we will have the pointer value 4 displayed. This is because the BASIC PUT statement outputs four bytes at a time. If we add one more line we can output more numbers and see the pointer moving through the file.

```
95 GOTO 30 and RUN
```

Each time you enter a number (and RETURN) you will see the pointer increase by 4. Try keying instead of a single number or character, a number between 0 and 33, 554, 419 (we need four bytes to represent this magnitude number). The PUT statement is the most economic way of storing numbers on disc.

If we change line 40 to

```
40 BPUT B and RUN
```

we will see the pointer increment by 1 for each entry. Using BPUT is the most economic way of storing single characters.

If we now modify the program we can see how the file pointer behaves with strings, so we will change lines 30 and 40, and add lines 10, 15, 35 and 99:

```
10 A= TOP
15 B=A+ 51
30 INPUT "ENTER" $A
35 IF $A="%" GOTO 99
40 SPUTD, A
99 SHUTD
```

We have now tidied up our program, providing an exit at Line 35 and shutting the "NED" file at Line 99. Note that Lines 10 and 15 define the maximum length of any string to 50 characters.

If we now run our program and enter a four character string *we* will see that the pointer moves from 0 to 5. This is because DOS inserts an "end of string marker'. This allows DOS, when retrieving strings from the file using SGET to establish how many characters to read for each string. So the pointer moves on by the number of characters +1 for each string access to file. Enter a few more strings into your "NED" file (so we can read it later) before exiting using "%". Save this program -- you may want to use it later, We will now enter a program for reading strings from " NED" sequentially, printing the File pointer after each string:

```
10 A=TOP
20 B=A+51
30 C=FIN"NED"
40 D=EXTC
50 IF D<=PTRC GOTO a
60 SGETC, A
70 PRINT '$A'
80 PRINT ' "THE POINTER IS" PTRC
90 INPUT $B
100 GOTO 50
110a PRINT ' "END OF FILE" '
120 SHUTC
130 END
```

When this program is run the first string is printed together with the pointer. Pressing RETURN will sequence through the file, printing the string and pointer at each access. When the end of file is reached a message is displayed and we return to BASIC.

By now you should be able to write a short program to read files using the GET (for 4 bytes at a time) and the BGET (one byte at *a* time) commands.

7. ASSEMBLY CODE ACCESS TO DOS

7.1 INTRODUCTION

The DOS file handling routines can be used with assembly language. These routines are entered via Jump Indirect instructions stored between FF00 and FFFF (hex). These Jump Indirect instructions, resident in ROM, refer to the vectors stored in RAM between 0200 and 023F. For example the routine to open or shut a file, OSFIND, is entered at FFCE which indirectly calls via FNDVEC at 0218. 0218 and 0219 contain the address of OSFIND (F97A).

The file handling routines, addresses, calls and call addresses and entry points are summarized in Table 2.

On NMI, any operating system interrupts are serviced, otherwise:

```
PHA
IMP (NMIVEC)
```

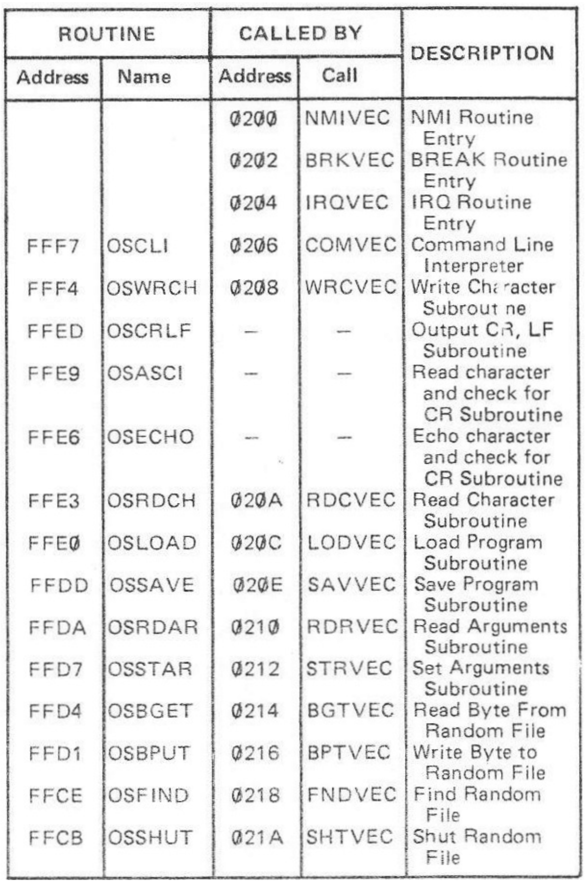
is executed.

On IRQ or BREAK (BRK):

```
STA 00FF
PLA
PHA
AND #10
BNE BRK
LDA 00FF
PHA
JMP (IRQVEC)
LDA 00FF
PLP
PHP
JMP (BRKVEC)
```

is executed. On power up or restart, the OS intializes the vectors from 0200 to 021A to its own handling routines.

Acorn Operating systems use memory in Zero Page starting from location $00AC up to location $00AD for their own purposes. Location $00FE contains the code of an ASCII character which is filtered from the output to the printer, this is initialized to $0A a line feed. Location $00FF is used for IRQ/BRK service as above.



ROUTINE		CALLED BY		DESCRIPTION
Address	Name	Address	Call	
		Ø2ØØ	NMIVEC	NMI Routine Entry
		Ø2Ø2	BRKVEC	BREAK Routine Entry
		Ø2Ø4	IRQVEC	IRQ Routine Entry
FFF7	OSCLI	Ø2Ø6	COMVEC	Command Line Interpreter
FFF4	OSWRCH	Ø2Ø8	WRCVEC	Write Character Subroutine
FFED	OSCRLF	–	–	Output CR, LF Subroutine
FFE9	OSASCI	–	–	Read character and check for CR Subroutine
FFE6	OSECHO	–	–	Echo character and check for CR Subroutine
FFE3	OSRDCH	Ø2ØA	RDCVEC	Read Character Subroutine
FFEØ	OSLOAD	Ø2ØC	LODVEC	Load Program Subroutine
FFDD	OSSAVE	Ø2ØE	SAVVEC	Save Program Subroutine
FFDA	OSRDAR	Ø21Ø	RDRVEC	Read Arguments Subroutine
FFD7	OSSTAR	Ø212	STRVEC	Set Arguments Subroutine
FFD4	OSBGET	Ø214	BGTVEC	Read Byte From Random File
FFD1	OSBPUT	Ø216	BPTVEC	Write Byte to Random File
FFCE	OSFIND	Ø218	FNDVEC	Find Random File
FFCB	OSSHUT	Ø21A	SHTVEC	Shut Random File

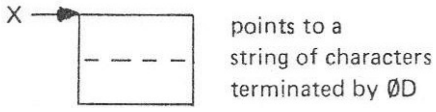
Table 2. Summary of Routines and Calls

7.2 OS SUBROUTINES

The OS routines are described below:

7.2.1 OSFIND

This routine returns, in the A Register, a file handle for the file whose name is pointed to by:



The handle is zero if the file does not exist otherwise it is a byte of value 1 to 255. If the carry flag is set on input the file must already exist and is opened for reading and updating. If the carry flag is clear on input the file need not exist, and will be used for output. The sequential pointer is set to zero.

7.2.2 OSSHUT

This routine closes the file whose file handle is in the Y Register. This involves writing out any buffers that contain data that has been changed, and updating the main disc Catalogue to show the length of the file. A zero file handle in the Y Register will cause all sequential files to be closed. After a file has been closed, the same file handle may be eventually used for a different file.

7.2.3 OSCLI

This routine interprets a string of characters held at 0100, terminated by a Carriage Return (0D), as an operating system command. All processor registers are used and detected errors are met with a BRK.

7.2.4 OSWRCH

This routine sends the byte in A Register down the output channel. This channel output is usually treated as ASCII data and special action may be taken on ASCII control characters. No processor registers are destroyed.

7.2.5 OSCRLF

This routine generates a Line Feed and then a Carriage Return using the OSWRCH routine. A Register will contain 0D on exit.

7.2.6 OSASCI

This routine uses the OSWRCH routine to output an ASCII character except that the Carriage Return character wil be output as Line Feed and Carriage Return, using the OSCRLF routine.

7.2.7 OSECO

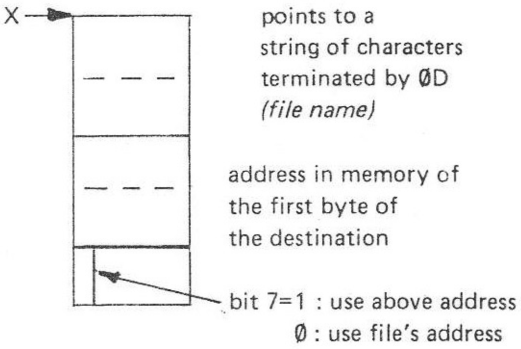
This routine fetches a byte using the OSRDCH routine and then writes it out using the OSASCI routine.

7.2.8 OSRDCH

This subroutine fetches a byte from the input channel into the A Register. The state of N, Z and C flags is unknown, X and Y Registers are preserved.

7.2.9 OSLOAD

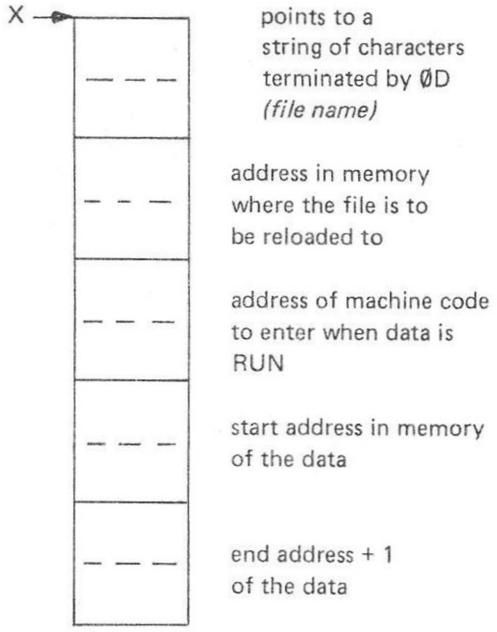
This routine loads all of a file into a specified area of memory. On entry the X Register must point to the following data in Zero Page:



This data is copied by the operating system and is not lost. All processor registers are used. If the processor's carry flag was set on input, a wait until completion is performed by interrupt or DMA driven systems. A BRK will occur if there is an error.

7.2.10 OSSAVE

This routine saves all of an area of memory to a specified file. On entry, the X Register must point to the following data in Zero Page:



This data is copied by the operating system and is not lost. All processor registers are used. If the carry flag was set on input, a wait until completion is performed by interrupt or DMA driven systems. A BRK will occur on an error.

7.2.11 OSSTAR

This routine sets the value of a file's pointer. On entry, the X Register points to Zero Page locations containing the value and the Y Register contains the file handle. The X and Y Register contents are not lost.

7.2.12 OSBGET

This routine returns the next byte from a random file. On entry the Y Register contains the file handle. The X and Y Register contents are not lost and the byte is returned in the A Register. The file's sequential pointer is incremented after the byte is read. Errors are met with a BRK.

7.2.13 OSBPUT

This routine adds the byte in the A Register to a random file. On entry, the Y Register contains the file handle. The A, X and Y Register contents are retained and the file's sequential pointer is incremented after the byte is added. Errors are met with a BRK.

7.2.14 OSRDAR

This subroutine returns the value of a random file's arguments. On entry, the X Register points to locations in Zero Page where the result is to be stored, and the Y Register contains the file handle, and the A Register specifies the argument, where:

A=0 : the file's sequential pointer in bytes
A=1 : the extent (length) of the file
A=2 : the region of the file

The data, typically 3 bytes is placed at X, X+1, X+2. The X and Y Register contents are retained.

7.3 RANDOM FILE OPERATION

7.3.1 Introduction

The DOS entry points involved with random files are:

OSFIND	prepare file for random access
OSSHUT	close file, release buffer, tidy up
OSRDAR	read parameters of some open file
OSSTAR	update parameters of some open file
OSBGET	read byte from file
OSBPUT	write byte to file

At any one time there may be up to five random files active. These active files will each have a one byte internal name referred to as the "file handle". File handles are allocated by OSFIND, cancelled by OSSHUT and passed as arguments to all other routines. Valid file handles are all non-zero, The use of zero as a file handle cause some of the routines to perform special functions.

An open file has various status information associated with it, including:

The sequential pointer	P (called PTR by BASIC)
The file extent	E (called EXT by BASIC)
The file region	R

The file is viewed as a row of bytes labelled 0, 1, 2, 3 etc. The sequential pointer holds the number of the next byte to be read or updated. As OSBGET and OSBPUT access successive bytes of the file, they increment P, which is a three byte value. The file extent E is another three byte value which holds the number of characters stored in the file. E=0 indicates an empty file and when E=P an attempt to go further onwards will return an end of file marker and subsequently cause a 'BRK'. The region R is used when output is sent to a file.

When a new file is created a region of disc is set aside for it. The new file will have an extent of 0, and R will show the size of the disc block allocated. As bytes are written to the file, E is incremented and when E=R the file is full and no further bytes may be added. R is always a multiple of the disc sector size (256 bytes) and cannot subsequently be changed (files cannot be extended). When a file is SHUT any unused sectors are released. It will always be true that:

0 <= P <= E <= R

and R is a multiple of 256.

7.3.2 OSRDAR and OSSTAR

OSRDAR and OSSTAR provide a means of interrogating P, E and R, and updating P. The ability to change P gives the user random access and the update capability for sequential files. If P is set beyond the extent of a file using OSSTAR, the space in the file from its old length to its new will be filled with bytes of value FF.

7.3.3 OSBPUT

OSBPUT writes bytes to a file. On entry, the A Register holds the byte to be written and the Y Register holds the file handle. If P=R, the file is full, and OSBPUT closes the file and executes a 'BRK'. Otherwise byte P of the file is updated and then, if P=E, both E and P are incremented (or P only is incremented). In the normal case when bytes are being added to the end of the file P=E.

7.3.4 OSBGET

OSBGET reads bytes from a file. On entry, the Y Register will hold the file handle. If P=E there are no more bytes in the file, so OSBGET sets the carry flag and returns the value FF. A second attempt to read at end of file causes a 'BRK'. Otherwise, OSBGET puts byte P of the file into the A Register, increments P and returns with the carry flag clear.

7.3.5 OSFIND

OSFIND opens files for input or output. Before calling OSFIND, it is necessary to provide a block of store containing the file name terminated by 0D, (Carriage Return). Two bytes in Zero Page point to this block, and the X Register contains the address of the pointer. If the carry flag is set the named file must already exist and E and R will be set to its actual size. If the file is not present on disc then OSFIND will return 0, this gives the user a way to detect whether a file exists or not.

If the carry flag is clear and the named file already exists, the old file region will be used, but with E set to zero initially. The result of this will be that the data in the old file cannot be accessed, but the region of the new file will he the same as the old, i.e. the old file is effectively deleted. If the old file was protected, OSFIND will fail. If no old file existed a new file is created with E set to zero and R given the default value of 4096 bytes. If there is not enough room on disc, then a 'BRK' is taken. If the user needs to control the size allowed for files (for instance requiring more than the default size), then the files should be pre-allocated by using SAVE so that OSFIND does not create them. Note that file names in OSFIND are effectively modified by the current drive and qualifier.

7.4 FILE BUFFERS AND CONTROL BLOCKS

The region of store from 2200 to 27FF is used for file buffers and control blocks. To reduce chance of disc corruption, the software maintains checksums on this memory, causing a 'BRK' if a check fails. In this event the safest thing to do is start from hardware reset, but in most cases it should he safe to shut files first.