CUMANA

DISK DRIVE GUIDE

ELECTRON

CUMANA®

The Cumana Disk Drive Guide for the Acorn Electron
(for use with the Cumana Floppy Disk System)

The Cumana Disk Drive Guide for the Acorn Electron

First Edition

A Cumana Publication

Published by:-
Cumana Limited,
The Pines Trading Estate,
Broad Street, Guildford,
Surrey, GU3 3BH

Printed by:
A3 Litho Limited, Guildford, Surrey, England.

Contents

Introduction

The Cumana Floppy Disk system for the Electron microcomputer consists of interface electronics and software in a plug-in cartridge, dual or single disk drive with mains power supply and utilities diskette.

The cartridge is intended to be plugged into the Acorn Electron Plus 1 Expansion unit.

The utilities diskette and the cartridge may also be supplied separately from the disk drives. Contact Cumana for the list of suitable alternative disk drives.

If any of the items mentioned above are missing from the package supplied, contact the supplier.

Please fill in the warranty registration card enclosed with the package.

Warning: This Equipment must be earthed.

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

GREEN AND YELLOW..... EARTH
BLUE..... NEUTRAL
BROWN..... LIVE

SEE DIAGRAM OVERLEAF.

The colours of the wires in the mains lead may not correspond with the colours or markings identifying the terminals in your plug, therefore proceed as follows:

The wire coloured green and yellow must be connected to the terminal marked with either the letter E (or the earth symbol) and/or coloured green (or green and yellow) .

The wire coloured blue must be connected to the terminal marked with either the letter N, and/or coloured black (or blue).

The wire coloured brown must be connected to the terminal marked with either the letter L, and/or coloured red (or Brown).

If the socket outlet available to you is not suitable for the plug supplied, the plug should be cut cleanly off and disposed of immediately. The exposed mains wires cause a potential shock hazard if the moulded plug were to be plugged in.

The moulded plug must only be used with the fuse and fuse carrier firmly in place. Plugs and fuse carriers from other manufacturers are not interchangable. If the plug or fuse carrier should become damaged, or the fuse carrier lost, the plug should be replaced in accordance with the directions above. The plug MUST NOT be used without a suitable fuse carrier. Should the fuse blow, the fault must first be remedied and the fuse replaced with a 3 AMP fuse that is ASTA approved to BS1362.

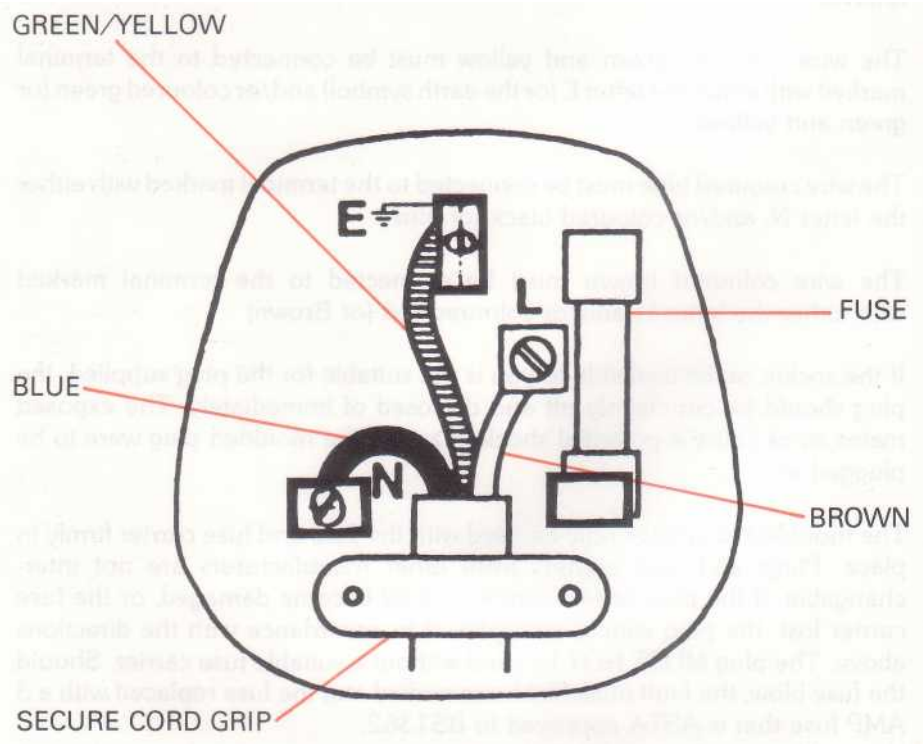
If in doubt consult a qualified electrician.

Warning

There are no user servicable parts inside a CUMANA disk drive. Before attempting to remove the lid be sure to isolate the unit by switching off at the mains and removing the plug from the main supply.

Red/Brown	To	Live
Black/Blue	To	Neutral
Green/Green&Yellow	To	Earth

FIT WITH A 3 AMP FUSE To BS 1362



Under no circumstances is the earth to be removed as damage could result to both computer and floppy disk drive, as well as the possible danger of electrocution.

If in doubt please consult a fully qualified electrician.

--

Chapter 1

Data Storage

There are various devices in which data can be stored, and in general they are referred to as memory. Memory size is normally measured in Kilobytes of information, where one byte is eight bits.

The Acorn Electron has 32 Kilobytes of user memory known as RAM or Random Access Memory, and as the name suggests, it can be used in a number of different ways. The RAM normally holds the current program and its variables. When a new program is needed it can be loaded into RAM that was previously occupied by the old program, thus overwriting it.

RAM can only hold information whilst the power is switched on. When the computer is switched off any program or data contained in its RAM will be lost.

The Electron also contains a certain amount of ROM or Read Only Memory. This memory contains the program or operating system that the Electron needs to communicate and interface with the outside world. It is normally known as firmware, and unlike RAM, it cannot be overwritten and it will not be lost when the machine is turned off.

Other types of memory associated with computers are those known as media storage peripherals: disk drives, magnetic tape drives, punched tape and card machines, etc. These devices are normally used for long-term data retention. These days, the most popular devices use magnetic media for data storage.

Data stored within a computer can be saved onto magnetic media devices before the machine is turned off or before another program is run. Similarly, data stored on a peripheral can be loaded into a computer for manipulation.

Floppy disk drives are a popular type of magnetic media peripheral. Floppy disk drives use diskettes, also known as floppy disks, as the data storage media. Floppy disk storage has the following advantages over cassette recorder storage:

a) Greatly increased speed of loading or saving data.
b) All programs contained on a diskette are catalogued automatically.
c) A program stored on a diskette can be accessed by specifying its name, without having to read the programs stored on the diskette before it:
d) Some advanced facilities are possible only with floppy disk storage.

Chapter 2

40/80 - Track Theory

A floppy disk is a thin circular piece of magnetic media contained within a flexible cardboard or plastic jacket. When inserted in a disk drive it rotates at 300 rpm. With a stationary read/write head this would produce a circular track around the diskette. However, the head inside a disk drive is able to move in steps, either towards the centre of the disk, or away from the centre, thereby producing a number of tracks, typically 40 or 80. Since the total distance covered by the head is the same for both 40- and 80-track systems, the head inside the 80-track drive must step twice as many times to go the same distance as the head inside the 40-track drive. This distance is measured in Tracks Per Inch or TPI. A 40-track drive has a pitch of 48 TPI and an 80-track drive a pitch of 96 TPI.

Fig. 2.1. below shows the basic outline of a 51/4" floppy disk. Its features are explained below.

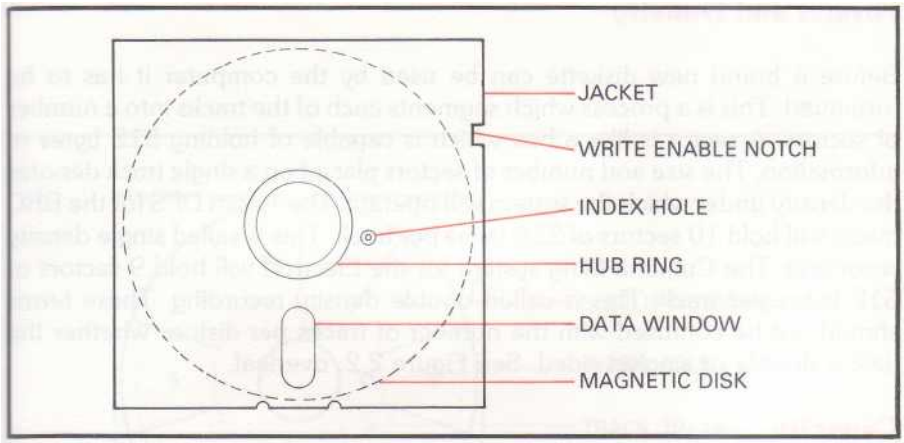


Fig. 2.1. A 5¼" floppy disk.

Index Hole

This is a small hole in the jacket of the diskette. When optically aligned with a second hole in the diskette itself it will produce a short pulse. This is referred to as a soft sector index and is used by the computer to time a single revolution of the disk.

Data Window

This is an elongated hole in the jacket which allows the read/write head to make contact with the floppy disk. The window appears on both sides of the disk as 40- or 80-track drives can have double heads for recording on both sides of a disk.

Write Enable Notch

This small notch on one side of the floppy disk is used to enable the disk drive to write on a diskette. When this notch is covered with a write protect tab, the disk drive will be prevented from writing data to the diskette. Useful for preventing accidental erasure of programs and data.

Hub Ring

This exposed portion of the floppy disk is mechanically clamped between the disk drive's clutch and drive hub, which spins at 300 rpm. The clutch engages as the disk drive door is closed.

Format and Density

Before a brand new diskette can be used by the computer it has to be formatted. This is a process which segments each of the tracks into a number of sectors. A sector is like a box which is capable of holding 512 bytes of information. The size and number of sectors placed on a single track denotes the density under which the system will operate. The Acorn DFS for the BBC micro will hold 10 sectors of 256 bytes per track. This is called single density recording. The Cumana filing system for the Electron will hold 9 sectors of 512 bytes per track. This is called double density recording. These terms should not be confused with the number of tracks per disk or whether the disk is double or singled sided. See Figure 2.2. overleaf.

Capacity

By knowing the number of tracks on a disk and how many sectors per track there are, the storage capacity of a disk can be calculated. For example, in the Cumana filing system there are 9 sectors per track with 512 bytes per sector. This is a storage space of 4,608 bytes per track. Thus

40-track single sided drive holds 184,320 bytes,
40-track double sided drive holds 368,640 bytes,
80-track single sided drive holds 368,640 bytes and
80-track double sided drive holds 737,280 bytes.

The disk drive capacity is usually quoted in rounded form:
180 Kbytes, 360 Kbytes and 720 Kbytes respectively.

Catalogue

The catalogue of a floppy disk is an area of diskette that contains all the necessary information needed for the computer to locate, and thus load, any data contained on the diskette. The Cumana filing system uses the first five sectors (sectors zero to four) on the first track (track zero) for the catalogue. Note: computers always count from zero. Sectors and tracks are therefore numbered from zero upwards.

When a program is saved to the disk, the disk filing system automatically makes a catalogue entry. The entry contains the length of the file, the file name and the number of the sector which describes the position of the file.

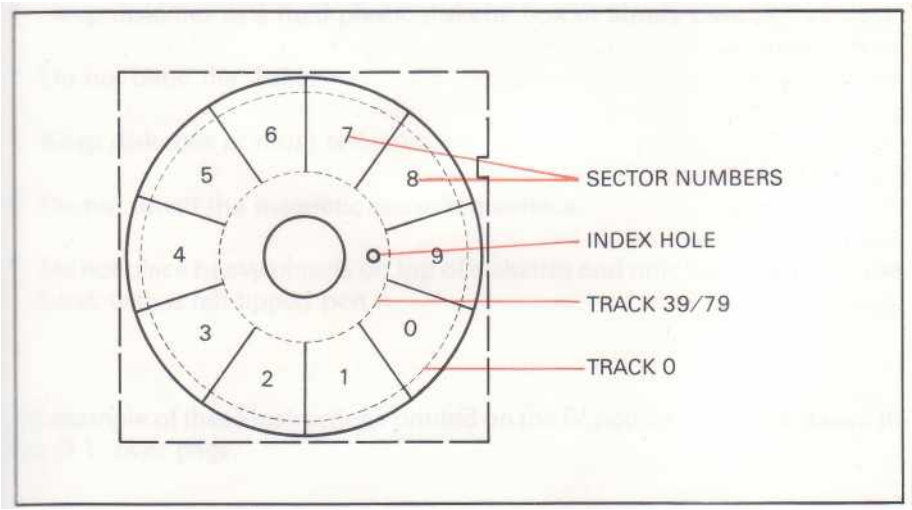


Fig. 2.2. Showing the 10 sector single density format (the sectors are laid out during the format process).

Chapter 3

Diskette Handling Precautions

To prevent damage to diskettes follow instructions printed on the floppy disk sleeve. In particular:

* Do not place diskettes within a magnetic field.

  Magnetic fields are generated by loudspeakers, transformers, electric motors, television sets, monitors and many other electric devices.

  For instance, leaving a diskette on top of an Electron is inadvisable. The Electron contains a power supply and a loudspeaker.

* Place diskettes in their protective sleeves when not in use.

  An unprotected diskette can pick up small particles of dust that can cause damage to the read/write head.

* Keep diskettes in a rigid plastic diskette box or library case.

* Do not bend the diskettes.

* Keep diskettes at room temperature.

* Do not touch the magnetic recording surface.

* Do not place heavy objects on top of diskettes and only write lightly on the label, with a felt-tipped pen.

An example of these instructions printed on the floppy disk sleeve is shown in Fig. 3.1. over page.

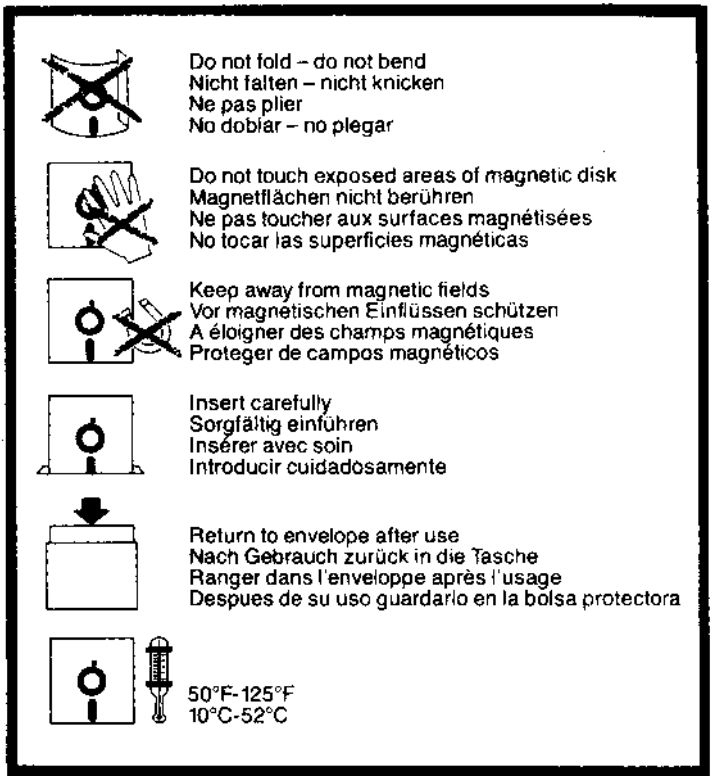
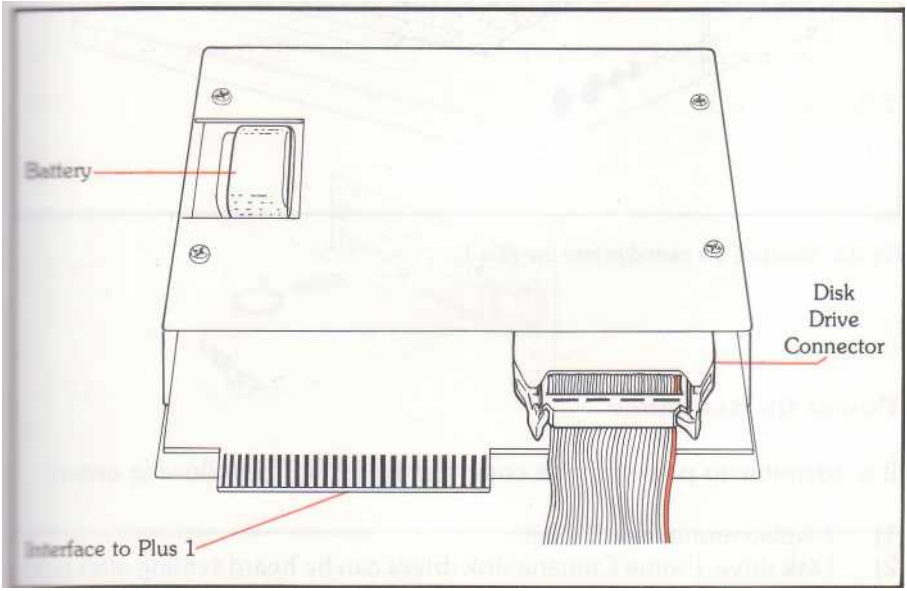


Fig. 3.1. Some disk sleeve warnings.

Chapter 4

Connecting Up

- Make sure that the Electron Plus 1 is installed and working as described in :he Electron Plus 1 User Guide supplied with the Plus 1.
- Make sure that the Electron and the disk drive are switched off and unplugged from the mains.
- Position the disk drive next to the micro.
- Firmly insert the ribbon cable protruding from the rear of the disk drive into the socket header in the cartridge. The cable should only go in one way, with the blue or red coloured stripe on the cable closest to the edge of the cartridge. See Fig. 4.1. below. Do not attempt to force the cable socket into the header. If the socket refuses to go in, check for possible bent pins in the header or incorrect orientation of the socket.



* Insert the cartridge into the Plus 1 as shown in Fig. 4.2. below preferably into the rear of the two sockets provided, so that the first is available for additional cartridges.

Remember to ensure that the Electron is switched off whenever you insert a cartridge into the Plus 1.

* Attach the lead with the moulded plug to the mains power outlet.

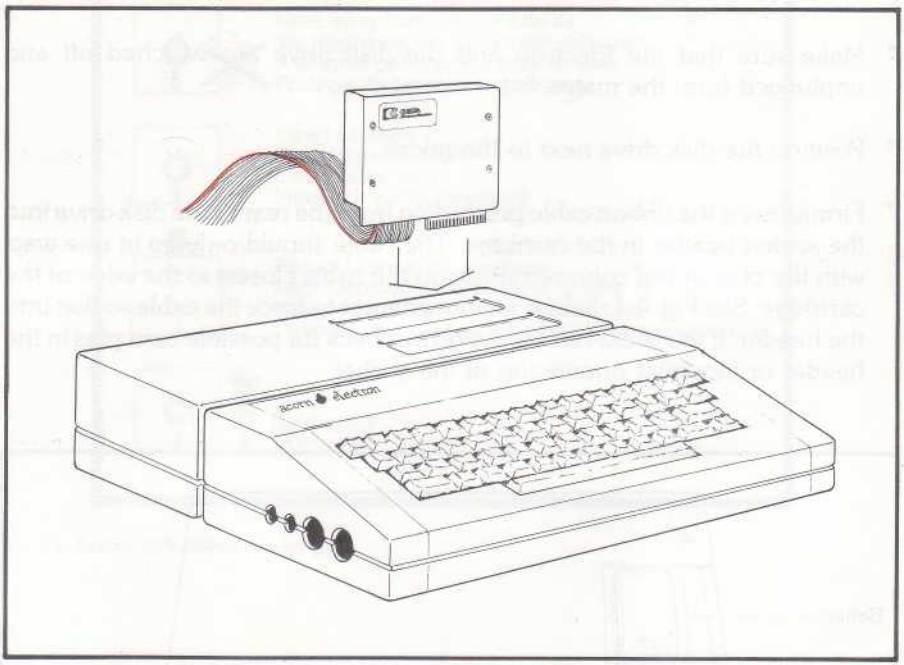


Fig. 4.2. Inserting the cartridge into the Plus 1

Power tip Sequence

It is advisable to power up the computer system in the following order:

1) Display monitor or TV set.
2) Disk drive. (Some Cumana disk drives can be heard settling after being switched on.)
3) Electron microcomputer.

If the system is connected correctly the following message will appear on the screen:

Acorn Electron #

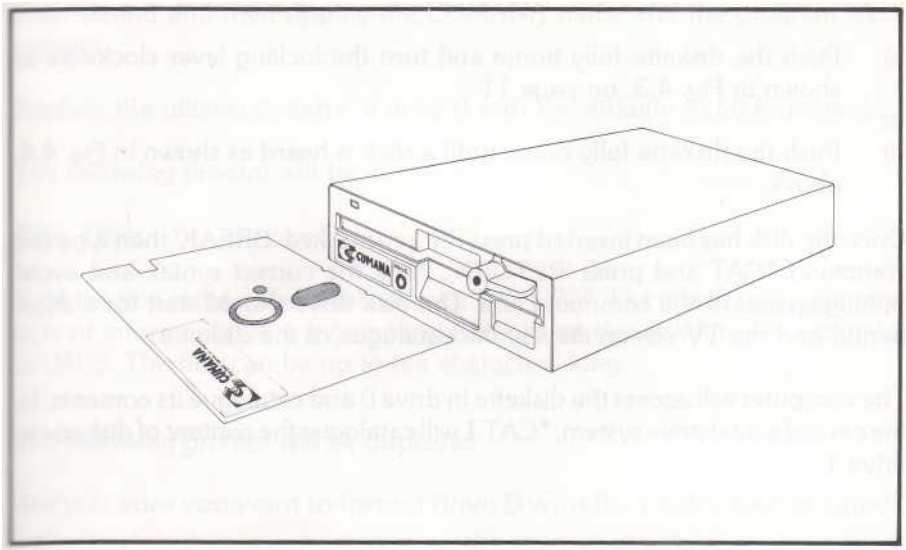
Cumana Disk System - 22 Nov 1984

BASIC

>

Other names may also appear if you have specialist ROM software fitted. the If the date is incorrect, this can be adjusted as described in Chapter 9 (Page 31).

Place the utility diskette into the disk drive marked '0' or the disk drive selected as drive 0 or master and close the door. Use Fig. 4.3. below or Fig. 4.4. overleaf for guidance.



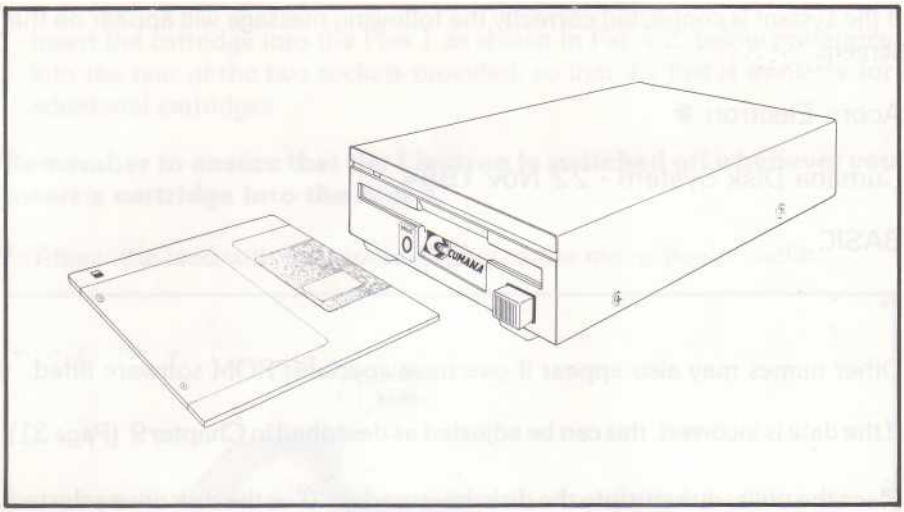


Fig. 4.4. Inserting a 3½" disk into the drive

Insertion

Depending on the type of drive you have purchased either:-

a) Push the diskette fully home and turn the locking lever clockwise as shown in Fig. 4.3. on page 11

or

b) Push the diskette fully home until a click is heard as shown in Fig. 4.4. above.

Once the disk has been inserted press the key marked 'BREAK' then type the command *CAT and press 'RETURN'. Note the correct syntax and avoid putting spaces in the command line. The disk drive should start for a short period and the TV screen display the catalogue of the diskette.

The computer will access the diskette in drive 0 and catalogue its contents. In the case of a dual drive system, *CAT 1 will catalogue the content of diskette in drive 1.

Chapter 5

Formatting

Diskettes must be formatted before they are used for data storage in a disk system.

The process of formatting erases any data that may have been stored on a floppy disk.

Two formatting programs are supplied on the utilities diskette, FORMAT (for Cumana DFS diskettes) and SFORMAT (for Acorn DFS diskettes).

To format a blank floppy disk that will be used in the Cumana disk system, insert the utilities diskette into drive 0 and type:-

*RUN FORMAT and then press 'RETURN'

The utility disk will be accessed, following which the screen will go blank for a short period and then display the CUMANA name and the program issue -lumber.

Replace the utilities diskette in drive 0 with the diskette to be formatted.

The following prompt will be displayed:-

Disk Title:

Type the title of the diskette followed by 'RETURN'. The title should reflect the type of information that is going to be stored on the diskette. For example, GAMES. The title can be up to ten characters long.

The following prompt will be displayed:-

Are you sure you want to format drive 0 with 80 tracks double sided?

or

Are you sure you want to format drive 0 with 40 tracks single sided?

Type 'Y' to format the diskette to the stated dimensions. Type 'N' to abort the formatting process.

The full syntax of the command is:-

***RUN FORMAT <DRIVE><TRACKS> S/D**

where <drive> specifies the drive containing the diskette to be formatted (`0' or '1'), <tracks> should be 40 or 80 and S/D means single or double sided. Thus, to format a doubled sided 40-track diskette in drive 1 type:-

***RUN FORMAT 1 40 D and press 'RETURN'**

and to format a single sided 80 track diskette in drive 0 type:-

***RUN FORMAT 0 80 S and press 'RETURN'**

The formatting process must not be interrupted. The following message is displayed while the diskette is being formatted:-

Formatting Track ..... nn

`nn' will be updated as each track is formatted. When the formatting has finished, each track is checked for correct format.

The following message is displayed while the diskette is being checked:-

Verifying Track ..... nn

`nn' will be updated as each track is verified. If an error is detected in the diskette format, a message Disc Error XX will be displayed and the verifying stopped.

Only diskettes that verified with no errors should be used for data storage.

Note that the Cumana disk system uses double density recording on diskettes. These diskettes are not interchangeable with diskettes formatted with single density recording, such as the standard Acorn DFS. To convert diskettes between the Acorn DFS format and Cumana DFS format use the utility programs described in Chapter 12 (Page 45).

Chapter 6

Making a Back-up

Copies of all important information should be kept on different diskettes. The ` Grandfather, Father, Son' principle of copying information is now a standard routine. Assuming the information is held on a master disk the following sequence is used:-

Day 1, master is copied to Grandfather
Day 2, master is copied to Father
Day 3, master is copied to Son

This system allows a record of changing information to be kept and recovered if accidentally erased from the master. On day 4 the cycle is repeated and the master copied to the Grandfather.

There are two different BASIC programs supplied on the utilities diskette, called BACKUP and COPY, which allow diskettes to be copied from one to another.

BACKUP

This program takes an exact copy of the floppy disk being copied. Each sector is read individually and written to the second disk. This sort of copying is called a physical backup.

To perform a physical backup ensure that the utilities disk is inserted in the drive. Make sure that BASIC is selected on the Electron and type:-

CHAI N "BACKUP" and press 'RETURN'

The Cumana name will be displayed together with the version number of the program. The following prompt will be displayed:-

Source Drive ?

Type the drive number for the source diskette. The following prompt will be displayed:-

Destination Drive ?

Type the drive number for the destination diskette. The following message will be displayed:-

Insert Source Disk in Drive 'n'
and Press Space to Continue

'n' is the source drive selected. Ensure that the source diskette is in the correct drive and then press the 'SPACE' bar. The diskette will then be read and its size printed on the screen, e.g.

80 Tracks Double Sided.

The following message will be displayed:-

Insert Destination Disk in Drive 'n'
and Press Space to Continue

'n' is the destination drive selected. Ensure that the destination diskette is in the correct drive. If the source and destination drive are the same, remove the source diskette from the drive before inserting the destination diskette. From now on the ' Insert Source Disk' and 'Insert Destination Disk' prompts will only appear if both source and destination drives are the same. When the backup has finished, the following message will be printed:-

Backup Complete

COPY

The copy program allows the copying of selected files from one diskette to another. It may also help to speed up access to the files. A diskette that has been used for some time may have the files spread over a large proportion of its area. When the files are copied, they will all be placed at the beginning of the diskette.

The diskette that contains the file to be copied is called source diskette. The diskette to which the file is to be copied is called destination diskette.

Using the program:-

- Insert the utility diskette into drive 0 and type CHAIN "COPY" and press ` RETURN'.

— For a dual drive system, insert the destination diskette into drive 1. For a single drive system, it is necessary to swap source and destination diskettes in drive 0 when requested by the program, which will also request the drive numbers for the source and destination diskettes.

— All files in a given directory or all files on the source diskette can be copied.

— As each file is found, the information about it is displayed:-

Files found:
FILE1
FILE2

'la' stands for load address, 'ea' stands for exec address and len' stands for file length.

— To select the files to be copied, the program prompts with each file name in turn and the files to be copied are selected by typing 'Y'. To skip a file, type `N'.

— The program indicates its progress while copying by displaying:-

Reading FILE1
Reading FILE2
Writing FILE1
Writing FILE2

— Before writing each file, if a file with the same name is already on the destination diskette a new name can be specified, so that the existing file is not overwritten.

At this point, it is wise to take a copy of the utilities diskette for safe keeping. The utility disk is not of a standard format and therefore the BACKUP program cannot be used. The COPY program however, will work as normal.

Chapter 7

File Specifications

Programs or files are normally given names to identify them. This name or <fsp> (filespec) can consist of numbers and letters to a maximum of ten. There are, however, some characters reserved for special purposes. These are #:.*! and they should not be included as part of the filespec. When a file is saved on diskette it is given a name. This name is recorded in the catalogue together with other information.

Example

SAVE ":0.$.TEST"

':0.' specifies drive 0
'$.' specifies the directory $
'TEST' is the name of the file.

In practice the <fsp> can be shortened thus:-

SAVE "TEST"

This is because unspecified drive and directory are set by default to 0 and $ respectively.

Within a catalogue, files may be placed in various directories. This allows the use of the same file name for different programs provided that each one is placed in a different directory. In the above example the directory was `$' (the default directory assumed by the computer at power on). To change the directory the following command is used:-

***DIR <dir>**

Any alphabetical character can be used but reserved characters #:.*! should again be avoided. To change the default drive number, the command:-

***DRIVE <drive>**

is used where <drive> is either '0' or '1'.

Wildcards

By using wildcards a group of files with similar filespecs can be processed to achieve a common end. Two of the special characters `#' and '*' are used. The symbol `#' is used to replace a single character and the symbol '*' is used to replace a group of characters. A filespec which includes a wildcard is called an alternative filespec or <afsp>.

For example:-

*INFO ABC2 command will give information about the file ABC2 only. Whereas
*INFO ABC# command will give information about all files beginning with ABC and having one more character making up the filename e.g. ABC1, ABC2, ABCD etc.
*INFO *.B* command will give information about all files in any directory beginning with 'B'. If information on all the files is required, *INFO *.* can be used.

Chapter 8

Filing System Keywords and Utilities

All Cumana disk system keywords begin with the character `*'.

*ACCESS <afsp> L

This command will lock or unlock a file. Locking a file will prevent it from being overwritten. To lock a file use:-

```
*ACCESS <afsp> L
```

To unlock a file use:-

```
*ACCESS <afsp>
```

Wildcards can be used. To lock all files on a diskette, use the command:-

```
*ACCESS *.* L
```

*BOOT <string>

This command will set an option which will be executed on pressing 'BREAK' and starting the Cumana filing system. The <string> can be quoted ("<string>") or not quoted. To include quotes in a quoted string, double quotes must be used.

Control characters in <string> must be preceeded by '|'. For example, control-M would be represented as '|M', which performs the same function as pressing ' RETURN'. To make the Electron print "HELLO" on start-up use:-

```
*BOOT "PRINT ""HELL0""|M"
```

To use the character '|' type '||'
To set up a function key to print "HELLO", use:-

```
*BOOT"*KEY1 PRINT ""HELLO""||M"
```

To turn the boot option off, type *BOOT and press 'RETURN'.

*BUILD <fsp>

This command will create a command file that can be directly executed by the command *EXEC <fsp>. As each line is typed, it will be given a line number. To save this file, press 'ESCAPE'. The command can also be used to create the auto-boot function. Pressing `SHIFT' and holding it while pressing 'BREAK' will automatically execute the !BOOT file. See *OPT for more details. Useful boot files are those that do repetitive tasks, e.g. a boot file that CHAINS a menu program to assist easy program selection, or a boot file that sets the function keys for specific tasks, such as word processing.

*CAT <drive>

This command produces on the screen the catalogue of files on a diskette. It will also show the current directory, the diskette title, whether the file is locked or unlocked, open or closed (see Chapter 10), the date and time each file was last updated and the current library. The <drive> is optional; if included, it can be '0' or '1'.

*CLOSE

This command will close any currently open files. This is similar to the CLOSE#0 facility in BASIC. However, it can be used even if BASIC is not selected.

*DELETE <afsp>

This command can be used to delete a single file or a number of files. If no wildcards are used the specified file is removed. If wildcards are used, each file matching the <afsp> is listed together with a prompt 'Y/N : '. Type 'Y' to delete the file. Type 'N' to leave it on the diskette. Open or locked files cannot be deleted. These are listed for information only.

For example, to delete file DATA, type:-

***DELETE DATA**

To delete files DATA1, DATA2 and DATA3A, but leave DATA and DATAFILE on the diskette type:-

DELETE DATA

and answer the prompts as follows:-

DATA	**Y/N : N**
DATA1	**Y/N : Y**
DATA2	**Y/N : Y**
DATA3A	**Y/N : Y**
DATAFILE	**Y/N : N**

***DIR ;,drive>.<dir>**

This command is used to set the current drive and the current directory. It can also be used to set only the current directory e.g. *DIR B, where is the directory to be selected.

If the directory is omitted from <fsp>, the current directory will be used. Similarly if the drive number is omitted from the <fsp>, the current drive will be used. For example, assuming the current directory is `A' and the current drive is '1',

LOAD "PROGNAME" is equivalent to LOAD "i1.A.PROGNAME".

After pressing 'CONTROL' and `BREAK' together or powering on, drive 0 will be selected and the directory will default to '$'.

***DRIVE <drive>**

This command is used to change the current drive. The drive number can be either 0 or 1. After pressing 'CONTROL' and 'BREAK' together or powering on, drive 0 will be selected.

***DUMP <fsp>**

This command will display a hexadecimal listing of the file with ASCII equivalent characters on the right hand side of the screen.

***EXEC <fsp>**

This command executes files created with the *BUILD command.
The content of these files is treated as equivalent to the keyboard input. Thus, to save repititive typing EXEC file can be used to auto-run *BUILD files.

*FREE <drive>

This command returns the hexadecimal number of used directory entries and the number of free sectors on the diskette.

*HELP <keys>

This command is used to gain information about the computer and the ROM software that is fitted into it, including the Cumana filing system ROM. Typing *HELP on its own will list the ROMs installed in the computer together with a list of keywords to which they will respond. Typing *HELP DFS will display the correct syntax for all the Cumana filing system commands. For example. type:-

***HELP DFS and press 'RETURN'**

The following will be displayed:-

Cumana Disk System Version 1.00
ACCESS <afsp> (L)
BOOT <string>
DELETE <afsp> DIR (
<directory>) DRIVE
<drive> FREE <drive>
INFO <afsp>
LIB <directory>
PBOOT
RENAME <old fsp> <new fsp>
TITLE <string>

Typing *HELP UTILS will produce a list of the general utilities available within the Cumana disk system. For example, type:-

***HELP UTILS and press 'RETURN'**

The following will be displayed:-

Cumana Filing System Version 1.00
BUILD <fsp>
CLOSE
DISC
DISK
DATE
DUMP <fsp>
TYPE <fsp>

***INFO <afsp>**

This command displays information about files on the diskette. The information is displayed in the form:-

<fsp> (LO) <load address> <exec address> <length> <date>

Where <fsp> is the full name of the file, (LO) is an abbreviation for locked or open, <load address> is the address to which the file would be loaded if a *LOAD command was used, <exec address> is the address at which execution would start if the file was *RUN, <length> is the length of the file and <date> is the date on which the file was last updated. The <load address>, <exec address> and <length> are all in hexadecimal. For example:-

```
*INFO DATA

$.DATA          L     00003000 00003000 001234 24 Nov 1984 10:20

*INFO *

$.DATA          L     00003000 00003000 001234 24 Nov 1984 10:20
$.DATA1         L     FFFF2000 FFFF1000 000100 24 Nov 1984 10:22
$.DATA2               FFFF2000 FFFF1000 000200 23 Nov 1984 23:10
$.DATAFILE      0     00003000 00002000 002000 23 Nov 1984 23:30
$.MYPROG              FFFF0E00 FFFF8023 001034 19 Oct 1984 13:21
```

***LIB R cirive>.<dir>**

This command is used to set the library drive and directory. Typing *LIB :1.C will set the drive to '1' and the directory to 'C'. After pressing 'CONTROL' and `BREAK' together or on powering on, the library is reset, the drive becomes '0' the directory '$' . These are the default values.

The correct use of the *LIB command is to redirect machine operating system (MOS) calls to the selected library drive and directory. For example:-

Typing *TEST or *RUN TEST and then pressing 'RETURN' will make the Electron search for the command 'TEST', initially in the MOS ROM. If it is not found there, the following sequence of searches will automatically ensue: sideways ROMs, current drive and directory. If this command is not found in any of the above, the current library drive and directory is searched, in this case drive 1 directory C. If the command is not found anywhere an error message 'Bad Command' will result.

*LOAD <fsp> <address>

This command loads a file into memory. The starting address is specified by <address>. If <address> is omitted the default load address is used from the diskette. For example:-

*LOAD M.CODE 2000 will load the file M.CODE into memory starting at address location &2000 (HEX).

*OPT N n

There are two OPT commands that concern the use of disk drives. They are:-

*OPT 1 n

Information about a file is not normally displayed when it is accessed. *OPT 1 1 causes the information about the file to be displayed every time a file is accessed. *OPT 1 0 turns this option off.

*OPT 4 n

Where (n) is a value in the range 0 to 3. The OPT 4 n command is displayed in the catalogue just above the library and to the right of the screen. The OPT 4 n command specifies the auto-boot function of the computer.

*OPT 4 0	do nothing, the auto-boot function is off
*OPT 4 1	automatically *LOADs the file !BOOT.
*OPT 4 2	automatically *RUNs the file !BOOT.
*OPT 4 3	automatically *EXECs the file !BOOT.

If the file !BOOT is not on the diskette the 'File not found' error message will be produced.

When setting the option, the *OPT 4 n values are stored on the diskette. A write protected diskette will generate the error 'Disk Read Only'.

*PBOOT

This command prints out the boot option set by the *BOOT command. If no

option is set, the reply will be:-

No option

***RENAME <old fsp> <new fsp>**

This command is used to rename a file, but can also be used to transfer files from one directory to another. For example:-

***RENAME A.MYCODE B.MYCODE**

This has transferred the file MYCODE from the A. directory to the B. directory.

***RUN <fsp> or *<fsp> or */<fsp>**

This command is used to load the file <fsp> into the memory at the saved load address and start the execution at the saved execution address

(see *SAVE).
If the execution address of the file is -1 (&FFFFFFFF), the file will be EXECuted as if *EXEC <fsp> had been typed.

The alternatives are provided in order that files with the same name as disk system commands can be used. For example, if a file is called 'CAT', *CAT would generate a catalogue of the disk. Type */CAT or *RUN CAT instead.

***SAVE <fsp> <start> <end> <exec> <reload>**

This command is used to save a portion of memory, typically machine code. <start> is the 32 bit (4 bytes) HEX location of the start address, <end> is the 32 bit HEX location of the end address, <exec> is the 32 bit HEX location of the execution address, and <reload> is the 32 bit relocation address. Both the execution address and the relocation address can be omitted, in which case they are assumed to be the same as the start address. For example:-

***SAVE PROG 2000 2800 2100 1900**

will save the portion of memory between &2000 and &2800. The address at which execution begins is &2100 and if the file is called up again it will load at address &1900, unless otherwise instructed.

The alternative syntax for *SAVE is:-

***SAVE <fsp> <start> + <length> <exec> <reload>**

Where <length> is a 32 bit hexadecimal value which must be preceeded by '+'. Again the execution and relocation addresses are optional. For example:-

***SAVE PROG 2000+0800**

will save a portion of memory from address &2000 to address &2800.

***SPOOL <fsp>**

This command will open a file <fsp> and then any ASCII text sent to the screen will be written into the open file. To close the file, type *SPOOL again. This utility is particularly useful for creating text files from BASIC programs. Once created they can be printed with the *TYPE command.

Another powerful feature of *SPOOL command is the ability to merge two BASIC programs together. For example, to make two BASIC programs, called "FIRST" and "SECOND", into one program, called "ONE", then "FIRST" must have lower line numbers than "SECOND" (the BASIC RENUMBER command should be used). The procedure is as follows:-

LOAD "FIRST"	
*SPOOL LISTING	(opens the file "LISTING")
LIST	(lists the file "FIRST")
*SPOOL	(closes the file "LISTING")

A spooled file, called "LISTING", has been created.

LOAD "SECOND"	(loads the second file)
*EXEC LISTING	(executes "LISTING" as fast keyboard entry.) (The two programs are now merged.)
SAVE "ONE"	(saves the merged program on diskette)

Note: when closing the spooled file, *SPOOL must be typed without a file name.

***TITLE <disk name>**

This command is used to give a diskette a name. The name can be up to ten characters long but must not contain any spaces. If spaces are required the whole title must be placed in quotes. For example:-

***TITLE "GAMES DISK"**

*TYPE <fsp>

This command will display a file as lines of ASCII text. The command is useful for displaying files created using *BUILD or *SPOOL. However, it is not intended for typing out programs created by the BASIC language. A BASIC program is stored in compressed or tokenized form and when displayed under *TYPE appears as rubbish. However, BASIC programs can be converted into ASCII text using *SPOOL.

ABBREVIATIONS FOR CUMANA FILING SYSTEM COMMANDS

The filing system commands may be abbreviated as follows:-

***ACCESS**	***A.**
***BOOT**	***BO.**
***BUILD**	***BU.**
***CAT**	***.**
***CLOSE**	***C.**
***DATE**	***DA.**
***DELETE**	***DE.**
***DIR**	***DIR**
***DISC**	***D.**
***DISK**	***D.**
***DRIVE**	***DR.**
***DUMP**	***DU.**
***EXEC**	***E.**
***FREE**	***FR.**
***HELP**	***H.**
***INFO**	***I.**
***LIB**	***LIB**
***LOAD**	***L.**
***OPT**	***0.**
***PBOOT**	***p.**
***RENAME**	***RE.**
***RUN**	***R.**
***SAVE**	***S.**
***SPOOL**	***SP.**
***TITLE**	***TI.**
***TYPE**	***TY.**

For example:-

***RE.B.MYCODE C.MYCODE is equivalent to *RENAME B.MYCODE C. MYCODE**

Chapter 9

Date and Time

The date and time are kept by the disk cartridge even when the computer is switched off and can be accessed by typing *DATE and pressing 'RETURN'. This will give an answer in the form:-

26 Nov 1984 10:31

The program SET_TIME is used to adjust the time and date (see Chapter 12). This program can also be used to provide a continuous display of time and date on the screen. Use *RUN SET_TIME and press 'RETURN'.

Advanced Programmer Interface

The program SET_TIME uses the OSWORD interface (see Electron User Guide) to access the date and time information. The OSWORD calls use A=&50 to read the time and A=&51 to set the time. The X and Y registers point to a control block with the following format:-

Entry address: &FFF1
Indirection address: &020C

Byte	00	Seconds (0-59)
Byte	01	Minutes (0-59)
Byte	02	Hours (0-23; 0 = 12 am, 23 = 11 pm)
Byte	03	Day of week (1-7;1 = Sunday)
Byte	04	Date of Month (1-31)
Byte	05	Month (1-12; 1 = January)
Byte	06	Year (0-99)

Chapter 10
Random Access Files

A disk drive can position its read/write head at any point on the diskette, and read or write just one piece of information.

When using text files the computer reserves an area of memory, called a buffer. A buffer is 512 bytes in length, which is the same size as one section on a floppy disk. When writing to diskette, the buffer is filled by the user program. Then it is written out to the diskette, freeing the buffer for more information. When a file is read, the buffer is filled with one sector's worth of information. As it is emptied by the users program, so it is filled again from diskette until the end of a file is reached.

There are certain BASIC keywords that aid manipulation of text files. They are:-

BGET#	(gets one byte from the buffer)
BPUT#	(puts one byte into the buffer)
EOF#	(used for detecting the end of file)
EXT#	(indicates the length of file)
INPUT#	(inputs a block of data from the buffer)
OPENIN#	(opens a file for data input only)
OPENUP#	(opens a file for data update - read or write)
OPENOUT#	(creates a file and opens it for data output)
PRINT#	(outputs a block of data to the buffer)
PTR#	(points to the current file position)

See Electron User Guide for detailed explanation of the keywords.

Example Program

To store a list of names on a diskette type:-

```
10 A = OPENOUT "MYFILE"
20 PRINT#A, "EUGENE"
30 PRINT#A, "JOHN"
40 PRINT#A, "SHIRLEY"
50 PRINT#A, "STEPHEN"
60 CLOSE#A
```

The keyword OPENOUT in this case, opens a file called "MYFILE". The "A" is the channel variable allocated to the file. A maximum of eight files can be open at any one time.

The PRINT# A outputs the name string to channel A. The CLOSE# A stores the incomplete buffer onto the disk and in so doing closes "MYFILE".

If the file "MYFILE" already exists on the diskette it will be deleted and the file name "MYFILE" re-entered in the diskette catalogue under the currently selected directory.

Using the BASIC function PRINT# causes the bytes of the string to be written in reverse order preceded by two bytes. The first byte is the type and the second is the length of the string. The three possible types in BASIC are:-

Integer numbers	- &40
Real numbers	- &FF
Strings	- &00

The result of PRINT#A, "EUGENE" would be:-

Byte	0	&00	(type for a string)
Byte	1	&06	(six letters in "EUGENE")
Byte	2	&45	(="E" in ASCII)
Byte	3	&4E	(="N" in ASCII)
Byte	4	&45	(="E" in ASCII)
Byte	5	&47	(="G" in ASCII)
Byte	6	&55	(="U" in ASCII)
Byte	7	&45	(="E" in ASCII)

Random access files have a pointer which points to the current position in the file. After the above print statement, the pointer will be eight. The pointer increments every time a byte is put into the file and can be interrogated using the BASIC PRINT function. At this stage, PRINT PTR# A would result in "8" being printed.

To read out all the names in the file use function OPENIN. Use the following:-

```
1 0 A = OPENIN "MYFILE"
20 INPUT#A, A$
30 PRINT A$
40 INPUT#A, A$
50 PRINT A$
```

```
60 INPUT# A, A$
70 PRINT A$
80 INPUT# A, A$
90 PRINT A$
100 CLOSE# A
```

Line 10 opens the file "MYFILE" for input and places the channel number in "A". Line 20 then reads the first name and line 30 prints it out. Line 40 reads in the next name and line 50 prints it out. This continues until line 100 is reached. This closes the file.

The value of the pointer can be written or read. Thus, the first name can be omitted by replacing lines 20 and 30 with PTR# A=8. The pointer was set to 8, because the first name contains 6 letters and two bytes are occupied by type and length.

If the number of names in the file is unknown, then a loop is required. This loop continually reads in a name and then prints it out. To find out when the end of the file is reached, the current pointer is compared with the length of the file:-

```
10 A = OPENIN "MYFILE"
20 REPEAT
30 INPUT#A, A$
40 PRINT A$
50 UNTIL PTR#A = EXT #A
60 CLOSE#A
```

Alternatively, line 50 can be replaced with **UNTIL EOF# A**.

When a file is opened for input, the data within it cannot be changed. To change the data, the file must be opened for update with OPENUP. To change the first name in the file to "RONALD" the following program is needed:-

```
10 A = OPENUP "MYFILE"
20 PRINT#A, "RONALD"
30 CLOSE#A
```

Note that the new name has to be the same length as the old one. This is to ensure that the format of the file is not changed and following INPUT# statements will execute correctly.

The bytes of a file can be read and written individually using BGET# and BPUT#. If the above file is again opened for update with the channel number in A, a "BGET# A" will return &00 (the type byte for a string). The next "BGET# A" will return &06 (the length of the string). If the next action is "PTR# A = 7" followed by "BPUT# A, ASC"D"" the "R" in "RONALD" will be replaced by a "D", creating the name " DONALD".

Note: The following commands open files: *EXEC, *SPOOL, *TYPE, *BUILD and *DUMP. These are counted as part of the maximum of eight open files allowed. Any file which is open will have an '0' after it's name in the catalogue.

Chapter 11

Using the Filing System from Assembler

This chapter assumes familiarity with the assembly language for the Electron micro, as described in the Electron *User Guide*.

All facilities described in previous chapters are also available from assembler. Additionally, some other facilities are only available from assembler. Both types are described in this chapter.

An area of memory is prepared as a control block which contains the information relevant to the command. The A, X and Y registers may also be relevant. Functions are performed by preparing the control block and the registers and then calling a particular address. All functions are indirected via a vector which can be redirected to a user supplied routine if a different function is required. The following list contains all routines, their entry address, indirection addresses and their function.

routine	entry	vector	in-direction	function (BASIC equivalent)
OSFIND	&FFCE	FINDV	&021C	Open or close a file (OPENIN, OPENOUT,OPENUP).
OSARGS	&FFDA	ARGV	&0214	Read or write information about an open file (PTR#,EXT#).
OSFILE	&FFDD	FILEV	&0212	Load or save a file (LOAD, SAVE).
OSBGET	&FFD7	BGETV	&0216	Read a byte from an open file (BGET#).
OSBPUT	&FFD4	BPUTV	&0218	Write a byte to an open file (BPUT#).
OSGBPB	&FFF1	GBPBV	&021A	Read or write a number of bytes from/to an open file.

X and Y registers are used to point to an area of memory in most cases (X contains the low byte and Y the high byte of an address). For example, if the address of the area of memory is Hex &1234 then X must be &34 and Y must be &12.

OSFIND

Entry Address: &FFCE
Indirection address: &021C

This routine opens a file for input, output or update or will close a specific file or all files.

If A is non-zero, X,Y must point to the name of a file which is terminated with a carriage return (Hex &0D).

A=&40 the named file will be opened for input.
A=&80 the named file will be opened for output.
A=&C0 the named file will be opened for update.

On exit, if A=0, this means that the named file was not found. If A is non-zero, the named file is opened and the channel number is returned in A.

If A=0 on entry, then the action depends on Y. If Y=O, then all open files will be closed (equivalent to BASIC's CLOSE#0). If Y is non-zero, it must contain the channel number of an open file which will then be closed.

OSARGS

Entry address: &FFDA
Indirection address: &0212

This routine reads or writes the attributes (length and pointer) of an open file. It can also be used to read various filing system parameters.

If Y is non-zero, then it must contain an open file's channel number. A determines the effect of the routine. X contains the low byte of the control block address. The high byte of the address is assumed to be zero (i.e. the control block is in zero page).

A=0 means read the file's sequential pointer and place its value in zero page.

A=1 means write the file's sequential pointer from the value supplied in zero page.

A=2 means return the file's length in zero page.

A=&FF this is used to 'ensure' that a file is up to date. As explained in Chapter 10, an open file may have a partially updated block in memory. This call is used to make sure that such a block is written to disk.

If Y is zero, again the function is determined by A.

A=0 means return the type of the filing system in A. Current types of filing system on the Electron are:-

0	No file system
1	1200 baud cassette file system
2	300 baud cassette file system
3	ROM pack file system
4	Cumana disk file system
5	Econet file system

A=1 means return the address of the rest of the command line. This is used to obtain the address of any parameters that may have been passed when a file has been *RUN. The address of anything else that might have been typed in is returned. This will point to a carriage return (Hex &0D) if nothing else was typed.

A=&FF is used to 'ensure' all files are up to date on disk. This is similar to A=&FF, Y=channel number above, but ensures all open files are up to date.

X and Y are preserved, but all other registers may be altered.

OSFILE

Entry address: &FFDD
Indirection address: &0212

This routine is used for loading and saving a file, or for updating file attributes. On entry, X,Y should point to a control block in the following format:-

Byte	00	Address of the file name terminated by &OD.
	01	Least significant byte first.
Byte	02	Load address of file. Least significant byte first.
	03	
	04	
	05	
Byte	06	Execution address of file. Least significant
	07	byte first.
	08	
	09	
Byte	10	Start address of data for write operations.
	11	Length of file for read operations.
	12	Least significant byte first.
	13	
Byte	14	End address of data, i.e. the address of the
	15	byte after the last byte to be written or
	16	file attributes.
	17	Least significant byte first.

A determines the type of operation to be performed:-

A=0 save a section of memory as the named file. The catalogue information is also written. This is equivalent to *SAVE.

A=1 write a file's load address, execute address and attributes (locked or unlocked and the creation date).

A=2 write the load address of the named file.

A=3 write the execution address of the named file.

A=4 write the attributes of a file.

A=5 read file load address, execution address, length and the attributes.

A=6 delete the named file. This is equivalent to *DELETE

A=&FF load the named file. The file is loaded into the position determined by the first byte of the execution address supplied in the control

block (byte 6). If this byte is zero, the file will be loaded at the load address supplied in the control block. If the byte was non-zero, the file's own load address is used.

The attributes consist of the date the file was last updated and the access to the object:-

Byte	14	bits 7 to 3: year (relative to 1983) bits 2 to 0 and
Byte	15	bit 7: month (1-12, 1 = January) bits 6 to 2: date (1-31) bits 1 to 0 and
Byte	16	bits 7 to 5: hour (0-23, 0 = 12 a.m., 23 = 11 P.m.) bits 4 to 0 and
Byte	17	bit 7: minutes (0-59) bits 6 to 1 are user definable. bit 0 is access (0=unlocked, 1=locked)

For example, if the attributes are as follows:- byte 14 = &OD, byte 15 = &El, byte 16 = &4B and byte 17 = &81,
(in binary byte 14 = 00001101,
byte 15 = 11100001, byte 16 = 01001011 and byte 17 = 10000001) then

Year = 00001 = 1 = 1984
Month = 1011 = 11 = November
Date = 11000 = 24
Hour = 01010 = 10 am
Minutes = 010111 = 23
Access = 1 = Locked.

The X and Y registers are preserved.
On exit A=1 if file was found and A=& if file was not found.

OSBGET

Entry address: &FFD7
Indirection address: &0216

This routine reads a single byte from the current pointer position of the file (the file's channel number is in Y). The byte is returned in A. If the end of the file has been reached, the carry flag is set and the value &FE is returned. The carry is clear if the transfer was successful. X and Y registers are preserved.

OSBPUT

Entry address: &FFD4
Indirection address: &0218

This routine writes a single byte at the current pointer position of the file (the file's channel number is in Y). The byte written is supplied in A. X and Y registers are preserved.

OSGBPR

Entry address: &FFD1
Indirection address: &021A

This routine reads or writes a multiple number of bytes from or to a file. The function of the routine depends on A. It can also be used to read information about the current catalogue. On entry X, Y must point to a control block containing the following information:

Byte	00	Channel number
Byte	01 02 03 04	Pointer to the data. Least significant byte first.
Byte	05 06 07 08	Number of bytes to read from or write to a diskette. Least significant byte first.
Byte	09 10 11 12	PTR# value. Should be set before the transfer (if used). Least significant byte first.

A determines the function of the routinei-

A=1 Set PTR# pointer to the value supplied (bytes 9 to 12).
Write the specified number of bytes to the file from the given address.

A=2 Write the specified number of bytes to the file from the given address. Ignore PTR# value supplied (bytes 9 to 12).

A=3 Set PTR# pointer to the value supplied.
Read the specified number of bytes from the file to the given address.

A=4 Read the specified number of bytes from the file to the given address.
Ignore PTR# value supplied.

If the value of A is greater than four, the routine returns information on the current catalogue. The control block is identical to that above, but only the address field is used.

A=5 The title of the currently selected drive and the current boot option are returned to the address supplied in the control block.
The data returned is in the format:-

single byte giving the length of the title (i.e. 10)
the title in ASCII characters
single byte boot option

0 = No option
1 = *LOAD the !BOOT file
2 = *RUN the !BOOT file
3 = *EXEC the !BOOT file.

A=6 Read the currently selected directory and drive number. The data returned is in the following format:-

single byte giving the size of the drive number (i.e. 1) the drive number in ASCII
single byte giving size of the directory name. (i.e. 1) the currently selected directory name.

A=7 Read the currently selected library directory and the library drive number. They are returned in the same format as A=6 above.

Values of A greater than seven are used to return the file names in the catalogue. The format of the control block is altered:-

Byte	00	Sequence number of the catalogue.
Byte	01 02 03 04	Pointer to memory area for transferring the file names to.

Byte	05 06 07 08	Number of file names to be read.
Byte	09 10 11 12	Number of file names to be skipped before the file names are transferred.

A can have the following values:-

A=8 This reads file names from the currently selected directory. The file names are returned in the form:-

length of file namel
file namel
length of file name2
file name2
etc.

A=9 This reads file names, from the beginning of the catalogue regardless of selected directory. File names are held in alphabetical order. The names are returned in the following format:-

length of directory + length of file namel
directory of file namel
file namel
length of directory + length of file name2
directory of file name2
file name2
etc.

After any of these calls, the address is updated to point to the end of the data transferred. The number field is updated to show the number of file names remaining to be transferred (normally zero unless the end of the file or catalogue has been reached). The pointer value or file number to be skipped is set so that it points to the next byte or file name. If the end of the catalogue or file is reached before the call is completed the carry flag is set. A, X and Y are preserved.

Chapter 12

Utilities

MENU

The menu allows a rapid and easy selection of the utility programs on the utility disc.

Using the program:-

- Insert the utility disc and press and release the BREAK key while holding down the SHIFT key. Do not release the SHIFT key before the BREAK key has been released.

– Each utility program name is shown on a separate line, preceded by the letter to be typed to select it and followed by a summary of the program's use. Hit the key corresponding to the program you wish to run.

– Some of the programs need parameters specified before running. For example, the FORMAT utility needs to know the drive number of the disc to be formatted. If any parameters are required, each will be requested by the MENU program in turn, followed by the possible options in brackets. Type the first letter of the option required, or simply press RETURN for the default (the default is always the first option shown in the brackets).

– Finally, the command that could have been typed to run the utility directly is displayed, and the program is loaded and run automatically.

The following utilities are supplied:-
BACKUP
COPY
D_TO_S
DISK_EDIT
FORMAT
SET_TIME
SFORMAT
ST 0_D D
SVERIFY
VERIFY
A_TO_C

BACKUP and COPY programs are described in Chapter 6.
FORMAT program is described in Chapter 5.

D_TO_S

This program copies files from double density Cumana DFS system, onto a single density Acorn DFS diskette that can be used on the BBC microcomputer.

Using the program:-

- Insert the utility diskette into drive 0 and type CHAIN "D_TO_S" then press ` RETURN'
- The program requests the number of the destination drive: reply with '0' if the files are to be copied onto the top surface of the destination diskette or `2' if they are to be copied on the bottom surface.
- For a dual drive system, insert the double density source diskette into drive 1. For a single drive system, it is necessary to swap source and destination diskette in drive 0 when requested by the program. which will also request the drive number of the source diskette.
- All files in a given directory or all files on source diskette can be copied.
- As each file is found the information about it is displayed:-

  Files found:

FILE1	**laE00**	**ea8023**	**1en212**
FILE2	**la 1200**	**ea1200**	**len300**

- If there is not enough space left on the diskette or not enough space in the diskette catalogue to hold all the files to be copied, the program will ask for a selection of files to be copied.
- The program indicates its progress while copying by displaying:-

Reading FILE1
Reading FILE2
Writing FILE1
Writing FILE2

— Before writing each file, if a file with the same name is already on the destination diskette a new name can be specified, so that the existing file is not overwritten.

If a file name on the source diskette is more than 7 characters long, it must be shortened.

DISK_EDIT

This program reads and writes diskettes directly. It can be extremely useful, but should be used with care and is intended for advanced users. To experiment with this program, format a diskette and save some test files. **Never experiment with the utilities or other valuable diskettes in the drive.**

Using the program:-

— Insert the utilities diskette and type CHAIN "DISK_EDIT" and then press ` RETURN'

— Remove th utilities diskette and insert the diskette to be read or altered.

— The program operates in Cumana format double density double sided mode on the current default drive. To change drive number and/or drive parameters use the procedure described in Changing Drive Option below.

— To exit the program press 'BREAK'.

— Press 'E' to edit a sector of the diskette. Enter the number of the sector to be editted. Cumana format diskettes have 512 byte sectors and 9 sectors per track. Sector 9, for example, is therefore the first sector of the second track.

— Each of the lower 256 bytes of the 512 byte sector is displayed as ASCII equivalents preceded by a space. If there is no ASCII equivalent, the byte is displayed as two hexadecimal digits.

— To obtain a pure hexadecimal display, press `FUNC' and 'H' together.

— To alter a byte, overtype the byte in the appropriate place. Each byte occupies two positions. If the cursor starts in the left-most position, the program expects two hex digits. If the cursor starts in the right-most position, the program expects a single ASCII character to be entered.

— To write out the altered sector to diskette press `FUNC' and 'W' together.

— To move to the upper part of the same 512 byte sector or to the next sector press `FUN' and 'N' together.
To move to the lower part of the same sector or to the previous sector press ` FUNC' and 'P' together.

— Press 'ESCAPE' to finish editing.

Changing Drive Option

— Type 'D' to change drive parameters.

— Diskettes formatted in different ways can be read by selecting appropriate drive parameters.
The program prompts with:-

Double density (No = single density Acorn DFS format)
512 byte sectors (No = 256 byte sectors, 16 sectors/track)
Cylinder access (No = surface access)
Drive number

— Cylinder access is only possible with double sided drives. With cylinder access and 512 byte sectors, sectors 0-8 are read from track 0 of the top surface of the disc, sectors 9-17 are read from track 0 of the bottom of the disc, sectors 18-26 are on track 1 of the top surface etc.

— If single density is selected, 256 byte sectors with surface access will be assumed.

— The drive number permitted is 0 or 1 for cylinder access. and 0 to 3 for surface access. For surface access, drives 2 and 3 select the lower surfaces of the drives which have top surfaces 0 and 1 respectively.

— 40-track drives may be used. However, it is possible in this case to enter sector numbers which are not on the diskette, resulting in disk errors being displayed and possible damage to the drive.

SET_TIME

This program sets the date and time.

Using the program:-

— Insert the utility diskette and type *SET_TIME and press 'RETURN'

– The date and time will be read from the cartridge and displayed. If any part of the date read is obviously incorrect, question marks will be displayed. The reason for this error is usually reduced power in the battery. The battery will recharge during use.

– The time will be updated continuously until any key is pressed. Thus, this program may be used to show a continuous time and date display on the computer.

  To change the date or time, move the flashing cursor under the digit to be altered by using cursor left and right keys.

– Make the new setting by overtyping with the correct value. The program will ignore obviously incorrect entries.

– The display will not always alter by just one digit for every keystroke. The program may 'guess' to save typing and ensure that entries are not only part-made. For example, the month may be Jan and you enter 'S' for September. The month is automatically updated to Sep.

– The cursor moves to the next position that may be altered. For example, the month is May and it is to be changed to Jul. Enter J and the month changes to Jan with the cursor under the 'a', enter U and the month is Jun with the cursor under the 'n', finally enter L and the month changes to Jul with the cursor under the next alterable year digit.

– Enter year digits of 83 or above for 20th century dates, and below 83 for 21st century dates.

– The new date and time will only be saved following 'RETURN'. If incorrect date for the month is entered e.g. 31 Sep, the computer will beep, when `RETURN' is pressed. Press ESCAPE to exit the program without writing the new date and time.

SFORMAT

This is the single density formatter. It allows diskettes to be formatted as Acorn DFS diskettes. This utility together with the D_TO_S utility allows files to be copied onto Acorn DFS diskettes and therefore files to be transferred from Electron to BBC micro. The single density formatter works in the same way as the double density formatter except that it produces Acorn DFS diskettes. For example, to format an 80-track disk on drive 2 type:-

***RUN SFORMAT 2 80 and press 'RETURN'**

or to format a 40-track disk on drive 1 type:-

***RUN SFORMAT 1 40 and press 'RETURN'**

Type in a diskette title (up to 13 characters). The following prompt appears:-Are you sure you want to format drive 1 with 40 tracks?

Note: the process of formatting destroys any data previously recorded on the diskette.

Type 'Y' to format (type 'N' to exit the program). The diskette is then formatted. During this process the track number being formatted is displayed. All sectors are then checked to ensure they were correctly formatted. The diskette is now suitable for use in an Acorn DFS system. Cumana DFS files may be copied onto Acorn DFS diskettes using D_TO_S utility.

S_TO_D

This program copies the files from single density Acorn DFS diskette onto double density Cumana DFS diskette.

Using the program:-

- Insert the utility diskette and type CHAIN "S TOE" and press 'RETURN'
- When the program is running, remove the utility diskette and insert the single density source diskette into drive 0.
- The program requires the drive number of the source diskette: type '0' if the files are on the top surface of the diskette or '2' if they are on the bottom surface.
- For a dual drive system, insert the double density destination diskette into drive 1. For a single drive system, it is necessary to swap source and destination diskettes in drive 0 when requested by the program, which will also request the drive number of the destination diskette.
- The catalogue is read from the source diskette and as each file is found the information about it is displayed:-

  Files found:

FILE1	**laE00**	**ea8023**	**len212**
FILE2	**lal200**	**la1200**	**len300**

— The program prompts with 'Copy all files? (Y/N)'. Type 'Y' if all files are to be copied. Type 'N' for selective copying. Then the program will prompt with each file name in turn and only the files to which 'Y' is typed will be copied.

— The program indicates its progress while copying by displaying:-

Reading FILE1
Reading FILE2
Writing FILE 1
Writing FILE2

SVERIFY

This utility verifies all sectors on Acorn DFS diskettes (single density). To check a diskette for accumulated errors, type:-

***RUN SVERIFY and press 'RETURN'**

The program will attempt to read in every sector from the diskette in drive 0. Errors and their track position are reported. To verify drive 3, type:-

***RUN SVERIFY 3 and press 'RETURN'**

VERIFY

This utility verifies all sectors on Cumana DFS diskettes (double density). The program operation is identical to operation of SVERIFY.

A_TO_C

This program copies files from an Acorn Electron ADFS format diskette onto a double density Cumana format diskette.

Specifying the source directory:-

- The program is very similar as the S_TO_D utility, with the exception that the files from only one specified directory are copied. Therefore there is an additional first stage to determine the ADFS directory holding the files to be copied. The program scans the root directory, and prompts for each further directory found there. If you respond that a directory is in the pathname of the files, this new directory is scanned as for the root. This process is

continued, until the point where there are no further directories in the path specified, and at that point all the filenames found in the final directory are listed.

— The destination directory will be the directory selected before the program was started.

— The program will now continue as for S_TO_D.

Chapter 13

Transfer from Tape to Disk

The Cumana Disk System does not use Electron user RAM. PAGE remains at Hex &0E00 and therefore all programs written for the TAPE Filing System should run on the Cumana Disk System. Transferring a file from TAPE to DISK is straightforward:-

For a BASIC program type:-

*TAPE and press 'RETURN'
LOAD "NAME" and press 'RETURN' where "NAME" is the file name. Remember diskette file names can only contain a maximum of ten characters. Thus, some cassette file names will have to be shortened. Type:-

***DISK or *DISC and press 'RETURN'** followed by
SAVE "NAME" 'RETURN'

The file is now present on diskette.

When transferring machine code files to diskette, certain additional information is required. Type:-

*TAPE and press 'RETURN'	then
*OPT 1 2 and press 'RETURN'	this will give information about the file after it has been loaded.

For example:-

*LOAD NAME 2000 'RETURN'	the file is forced to load at &2000.

The following will be displayed:-

Loading

NAME 04 0500 00000E00 00000E3C

Where "NAME" is the file name, '04' is the number of blocks, '0500' is the length in bytes (Hex), '00000E00' is the loading address (Hex), and `00000E3C is the execution address (Hex).

This information is used in saving the file to the disk drive.

***DISK 'RETURN'**
***SAVE "NAME" 2000+<length> <exec address> <load address> ' RETURN'**

Where "NAME" is the file name required, <length> is the length of the file given above (`0500'), <exec address> is the execution address (`00000E3C) and <load address> is the reload address (`00000E00'). The file is now stored on the diskette and can be *RUN in the same way as from tape.

Appendix A

Technical Information

Sector numbering

Sectors are 512 bytes long and there are nine sectors per track. Sector 0 is sector 0 of track 0, sector 1 is sector 1 of track 0 etc. Sector 9 is either sector 0 of track 1 for a single sided diskette, or sector 0 of track 0 side 1 for a double sided diskette. In this way, a double sided diskette is treated as a single sided diskette.

Files

To avoid the error 'Can't Extend' which is very common in other filing systems, each file has a single sector allocated to it which describes the position of all the other sectors of the file. The descriptors are organized in pairs (sector number, length). The sector number is the first sector in a series of contiguous sectors. The length is the number of contiguous sectors making up that part of the file. The block is therefore organized as follows-

Byte	00 01	Sector number (low byte) Sector number (high byte)
Byte	02 03	Length (low byte) Length (high byte)
Byte	04 05	Sector number (low byte) Sector number (high byte)
Byte	06 07	Length (low byte) Length (high byte)
Byte	508 509	
Byte	510 511	Sector number (low byte) Sector number (high byte)
		Length (low byte) Length (high byte)

Sector number zero is used to denote the end of descriptors if a file requires less than 128 (this is equivalent to one sector). In the worst case each entry describes just one sector. Thus, the minimum file size before a 'Can't Extend' error message is generated is 64 Kbytes (128*512 bytes). In practice, this situation is very difficult to achieve.

Free Space

A single sector holds a bit for every sector on the diskette. If this bit is one, the sector is free. If this bit is zero, the sector is in use. This sector is called the 'bit map'.

When space is required, the bit map is scanned from the beginning until the first free sector is found. The number of this sector is then recorded in a descriptor sector with the number of consecutive free sectors. If this was not enough for the file, the bit map is scanned until the next free sectors are found and the process is repeated. If a file is being extended and therefore already has some sectors allocated to it, the search for free sectors begins with the last sector already allocated to the file.

Catalogue

The catalogue contains the size of the diskette, location of the bit map, number of directory entries used, free space available, diskette title and the directory entries. The catalogue is kept on track 0 of the diskette in the first 5 sectors. Sector 0 contains the following general information:-

Bytes	00 to 09	The disk title
Byte	10	The sequence number
Byte	11	Current boot option
Byte	12	Number of used directory entries
Byte	13	Free space (low byte)
Byte	14	Bits 0-2 free space (high order) Bit 6 number of sides (0=1, 1=2) Bit 7 number of tracks (0=40,1=80)
Byte	15	Address of bit map (low order)

High order address of bit map is always zero.

Each directory entry contains the following information:-

Byte	00	Directory of entry
Bytes	01 to 11	Name of entry
Byte	12	Bits 0-2: sector number (high order) Bits 4-7: size (high order)
Bytes	13 to 16	Load address (low order first)
Bytes	17 to 20	Execution address (low order first)
Byte	21 22	Size (low order) Size (middle order)
Bytes	23 to 25	Date
Byte	26	Bit 7: top bit of date Bit 0: access

Catalogue entries start immediately after the header information. Only complete entries are allowed in each sector, but as many as possible are fitted. When the end of one sector is reached, the next entry is at the beginning of the next sector. The bit map is in the sector immediately after the catalogue. Therefore, the sector number of the bit map is also the number of sectors making up the catalogue.

Direct Disk Access

To write programs which by-pass the filing system, or which format the diskette or use single density diskettes, an OSWORD call is provided (see Electron User Guide for details of OSWORD calls). Cumana Disk Filing System uses Western Digital WD1793 equivalent Floppy Disk Controller chip. The OSWORD is entered with A=&72 and with the X and Y registers pointing to a control block in the following form:-

Entry Address: &FFF1
Indirection Address &020C

Byte	00	Control byte: bit 0: Side select (0=1, 1=2) bit 1: Drive 0 select bit 2: Drive 1 select bit 3: Double density select (0=double,1=single)
Byte	01	Command register (see below)
Byte	02	Track register
Byte	03	Sector register
Byte	04	Data register
Byte	05	Address (low order)
	06	Address
	07	Address
	08	Address (high order)
Byte	09	Return code

If the command register is set to &FF, the drive running state is toggled i.e. if the drive was spinning it will stop; if it was stopped, it would start spinning. If the track, sector or data registers are &FF, they will not be written to the relevant WD1793 register. Other commands are:-

&80 Read sector
&A0 Write sector

Track and sector registers within WD1793 must be set up prior to issuing a read sector or a write sector command.

&04 Restore (move heads to track zero)
&14 Seek (desired track number in data register)
&50 Step-in (one track)
&70 Step-out (one track)

The bottom two bits (0 & 1) of any command which moves the heads, hold the head stepping rate.

The stepping rates available are:-

0 0	6 ms
0 1	12 ms
1 0	20 ms
1 1	30 ms

On completion of the command, byte 2 will contain the new value of the track register, byte 3 will contain the new value of the sector register and byte 9 will contain a copy of the status register. For read sector, the status register value should be masked with &1C. If the result is zero, the command has completed with no errors. For write sector, the status register value should be masked with &5C. For head moving commands, the status register value should be masked with &18.

The address supplied is a full four byte address for use with a 2nd processor, if fitted. This can only be used with single sector read or write, because a sector is buffered in the filing system's own internal memory. For multiple sector read or write and read or write track, the data cannot be buffered and therefore the address has to be in the I/O processor. The Electron also has to be in a low resolution mode, such as Mode 6. This is because of the way the display is generated on the Electron. For a high resolution display, the processor's bandwidth to the memory is too limited for the time critical regions of code in the filing system. Reading and writing across the tube is also too slow.

Sector Written To

The maximum number of sectors on an 80-track double sided diskette is 1,440. This requires 1,440/8 = 180 bytes for the bit map.

The bit map is therefore split in two halves and the second half is used to record if the sector has been written to. This is particularly useful when a large random access file is required which is all initialized to zero. The file is opened for output and the pointer set to the relevant size. The bits in the written to map remain set. Whenever a request is received to read in a sector, the written to map is checked, and if the bit is found to be one, the sector is not read in. In this case, an area of store is set to zero. As soon as any data is written to the sector, the written to map is adjusted and written to diskette.

Appendix B

Using Different Drives

The default head stepping rate for the Cumana disk system is 6ms. If a different head stepping rate is required, OSBYTE with A=255 should be used. This call alters some operating system values, including the screen mode which is selected on 'BREAK', the head stepping rate and a bit which reverses the action of `SHIFT'+`BREAK'. The call address of OSBYTE is &FFF4 and it functions as follows:-

<new value> = (<old value> AND Y) FOR X

Bits 4 and 5 are used to set the head stepping rate and should be

Bit	5	4	
	0	0	6ms
	0	1	12ms
	1	0	20ms
	1	1	30ms

The disk system records this value every time the disk is accessed and saves it in its own non-volatile memory. The value is then reset whenever 'BREAK' is pressed or on power-up. In order to ensure the value has been stored, the disk must be accessed before the power is switched off.

The disk system records the value of the entire byte and resets it on 'BREAK', so that the machine will power-up with options set.

The other bits in the byte have the following meaning:-

Bits	0 to 2	Screen mode.
Bit	3	Reverse `SHIFT'+`BREAK' action.
Bits	4 and 5	Disk stepping rate, as described above.
Bits	6 and 7	Reserved for future expansion.

Instead of using the assembly language entry point, *FX can be used. This is explained in the Electron User Guide, but some examples are given here:-

*FX 255,14	Sets 6ms, Mode 6, normal <SHIFT>+<BREAK> action.
*FX 255,30	Sets 12ms, Mode 6, normal <SHIFT>+<BREAK> action.
*FX 255,46	Sets 20ms, Mode 6, normal <SHIFT>+<BREAK> action.
*FX 255, 62	Sets 30ms, Mode 6, normal <SHIFT>+<BREAK> action.
*FX 255,2	Sets 6ms, Mode 2, reverse <SHIFT>+<BREAK> action.

Appendix C

Fitting Additional ROM

The Cumana cartridge has space for an extra ROM (word processor, spread sheet, graphics extension etc).

The cartridge can be returned to Cumana for the fitting or the following procedure may be adopted:-

1 Disconnect the unit from the mains.
2 Remove the cartridge from Plus 1.
3 Unscrew the four screws on the back of the cartridge.
The back can then be removed, exposing the empty ROM socket inside.
4 Holding the cartridge with the components facing you and the cable socket to your right, pin 1 of the ROM is at the bottom left corner of the spare socket. Carefully insert the ROM the correct way round into the empty socket, taking care to ensure all the pins of the ROM are in place. Do not touch any of the other components, as some are CMOS and could be damaged.
5 Replace the back of the cartridge reversing the procedure in 3.
6 Reconnect the cartridge as described in Chapter 4.

Appendix D

Replacing the Battery

The battery used is a Mempac 3.6V NiCad rechargeable battery and should last a considerable time, however when the battery requires replacing, a warning message will be printed on the Electron screen on power-up:-

WARNING! – Battery Broken

At this stage the battery should be replaced. Forward the complete cartridge to:

Service Division
Cumana Ltd
Daimler Close
Royal Oak Industrial Estate
Daventry, Northants.
NN11 5QJ
Telephone: (03272 79494)

NOTE: This battery must not be forced open or disposed of by fire.

Appendix E

Disk System Error Messages

Already Exists &CC
is returned when an attempt is made to rename a file to the name of an existing file.

Bad attribute &CF
is returned when an illegal character follows the file name for *ACCESS command.

Bad Command &FE
is returned when an unrecognized disk system command or utility is given.

Bad directory &CE
is returned if more than one letter is entered for the *DIR or *LIB command.

Bad drive &CD
is returned if the number entered is not '0' or '1'.

Bad filename &CC
is returned if the file name used is less than one or more than ten characters in length or contains illegal characters.

Bad Option &CB
is returned if the value given in *OFF is not '1' or '4'.

Bad String &FD
is returned if the string specified in a *BOOT command is invalid.

Bad Title &D7
is returned if invalid characters or too many characters are used in a *TITLE command.

Can't Extend &BF
is returned on the very rare occasion of the free space being so scattered on the diskette that it cannot be described in one sector.

Catalogue Full &BE
is returned when an attempt is made to save more than 90 files to a diskette.

Channel &DE
is returned when an invalid channel number is specified in a random access operation.

Disk changed &C8

is returned when a new diskette is put in the drive whilst files were open on the old diskette.

Disk error &C7

is returned when a corrupt, damaged or incorrect type of diskette is used.

Disk Full &C6

is returned when an attempt is made to write to a diskette that has insufficient room to allocate to the file.

Disk read only &C9

is returned when a write command is issued to a diskette that is write protected.

EOF &DF

is returned when an attempt is made to read off the end of a random access file.

File locked &C3

is returned when an attempt is made to overwrite or delete a locked file.

Not found &D6

is returned when the named file does not exist or is not on the currently selected drive and directory.

Open &C2

is returned when an attempt is made to delete or open an already open file.

Read only &C1

is returned when an attempt is made to write to a read only file opened with OPENIN.

Too many open files &C0

is returned when an attempt is made to open the ninth file. Only eight files can be open at the same time.

If the "Disk Fault" message ever occurs, there will be an extension to the error message. For example:-

Disk Fault xx at yyy

Where xx is the fault code and yyy is the sector number at which the fault occurred.

The most common fault is 10. This means that the diskette is corrupt or unformatted.

Fault code 08 is a CRC error. This means that the diskette is probably getting old and the data should be transferred to a new diskette as soon as possible.

Index