ACORN COMPUTER Acorn Computers Limited, 4a Market Hill, Cambridge CB2 3NJ, England. Telephone 0223 312772

ACORN TECHNICAL MANUAL.

6502 Disk Operating System.

Disk Drive Module.

Disk Controller Card....200,004.

Issue 1 Sept 1980.

Intoduction

This manual describes the Acorn parts required to operate a disk based storage system for both programs and data. The Operating System software described is for use on 6502 based machines with the Tele-text Visual Display Unit interface and a parallel ASCII keyboard. The hardware described is that used on the System 3 and 4 and also on the 6809 systems.

The Disk Operating System software keeps catalogue information and manages the insertion and deletion of data on the disk. The software has to be personalised to the type of disk drive since it includes information about the timing of the drive's actions. Only matched pairs of disk drives can be used in dual drive systems. Although 6809 users may refer to this manual for hardware information the 6809 DOS (which is booted into RAM from disk at start up) is described elsewhere.

Each Disk Drive is contained within a 7 inch module which plugs into the Acorn Eurocard frames. A Eurocard size controller card is used to control the drive(s) and this can be fixed into the module if required.

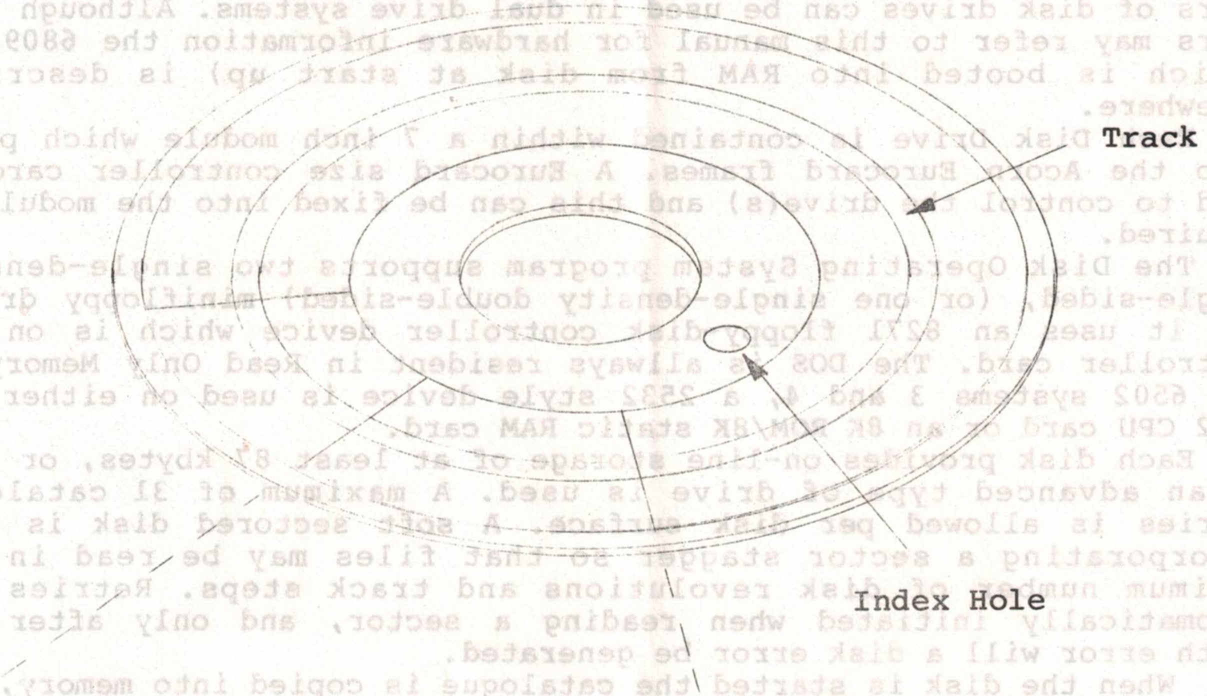
The Disk Operating System program supports two single-density single-sided, (or one single-density double-sided) minifloppy drives and it uses an 8271 floppy-disk controller device which is on the controller card. The DOS is allways resident in Read Only Memory in the 6502 systems 3 and 4, a 2532 style device is used on either the 6502 CPU card or an 8K ROM/8K static RAM card.

Each disk provides on-line storage of at least 87 kbytes, or more if an advanced type of drive is used. A maximum of 31 catalogue entries is allowed per disk surface. A soft sectored disk is used incorporating a sector stagger so that files may be read in the minimum number of disk revolutions and track steps. Retries are automatically initiated when reading a sector, and only after the tenth error will a disk error be generated.

When the disk is started the catalogue is copied into memory, and this catalogue is assumed valid while the disk is still 'ready'; thus, reading successive files requires as little head movement as possible. Changes made to the catalogue cause it to be written out to the disk. The catalogue and file buffers are stored in Random Access Memory at locations #2000 to #27FF and this RAM must be present in the system if the DOS is to work.

Use of the Diskettes

The five and a quarter inch diameter mini-floppy diskette consists of a metal oxide coated flexible mylar disk contained within a protective jacket with openings for the drive hub, index sensor and head. Data is stored on a number of circular tracks, the outermost track being track zero. The total number of tracks depends on the disk drive itself, common values are 35 and 40, with a denser 80 track drive beginning to appear. Under software control the tracks are each divided into ten sectors each of which contains 256 bytes of data together with identification information for the sector:-



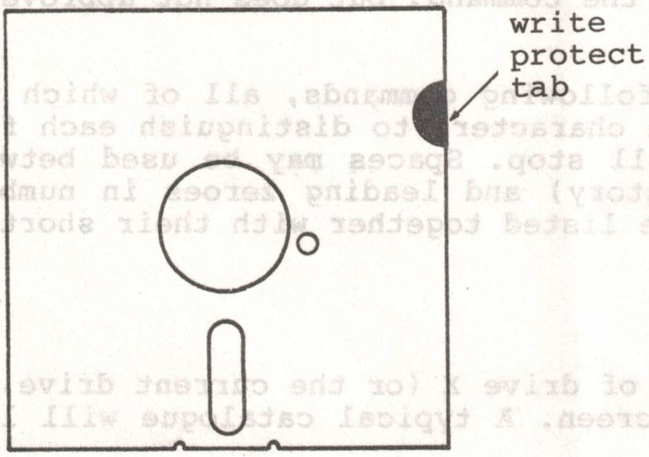
sector id field	post id gap	sector data field	post data gap	sector id field	post id gap
7 bytes	17 bytes	259 bytes	27 bytes		

The id field bytes contain an id address mark, the track address, the head address, the sector address, the sector length and two bytes of error checking code. The data field bytes contain a data address mark, 256 bytes of data and two bytes of error checking. Therefore there are 2560 bytes of user data per track, from the possible total of 3125 bytes per track. The sequence of data might include something that looks like an id field, so it is necessary for a physical mark from the disk to signal the start of the ten sectors. This mark is called the index mark and for this 'soft sectored' format it must only occur once per revolution. For a different type of system, termed 'hard sectored', an index mark indicates the start of each sector; this type of diskette cannot be used. The process of initialising the disk so that all the marking data is present is called formatting, and all new disks require formatting before they can be used for data storage. Every time any information is read from the diskette a check value is calculated and compared with the value held on the diskette; if the two values do not agree then the controller signals an error, but software will retry the particular operation up to ten times before notifying the user that an error has occured.

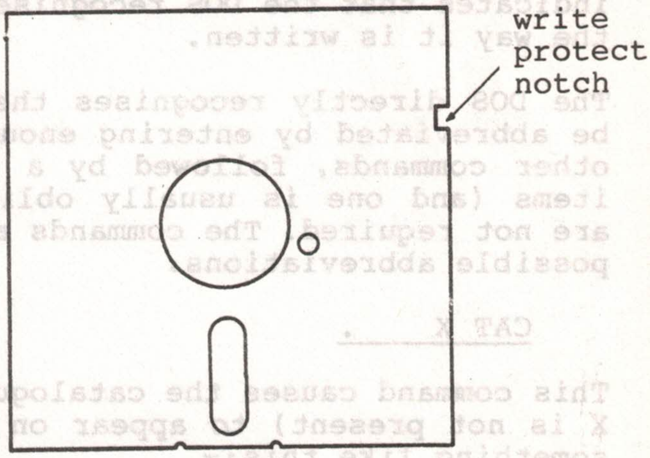
It is important that the diskette is handled and stored properly so that the integrity of the recorded data is maintained. A damaged diskette can impair or prevent recording of the data and can result in damage to the drive's head together with loss of information. The following points should be noted:-

(a) Do not touch the mylar disk surface with anything, especially fingers or hard objects
(b) Insert the diskette carefully into the drive until the backstop is reached. Do not shut the drive door until the diskette is fully inserted
(c) Open the drive door and adjust the position of the diskette if it rotates noisily
(d) Avoid damage to the centre hole which locates the disk onto the drive hub
(e) Return the diskette to it's paper storage jacket when not in use, keeping the head slot inside the jacket
(f) Keep the diskette away from magnetic fields, e.g. power supply transformers and cathode ray tube scan coils
(g) Do not bend or fold the diskette, keep and use the diskettes in room temperature i.e. between 50 and 100 degrees Fahrenheit
(h) Keep diskettes out of direct sunlight to avoid warping
(i) Write carefully on the diskette label with a fibre pen NOT a biro or pencil which may mark the disk inside

A notch on the side of the diskette envelope can be covered with a self adhesive tab, which will prevent any attempt by the drive to write on the disk:-



Write Protected



Un Protected

Operating System Commands

The Disk Operating System is a 4K byte program, resident in read only memory. It provides support for other programs, e.g. high level languages, by dealing with peripheral devices and allowing data to be filed on 5.25" mini-floppy disks. The standard devices which it uses are Acorn's Teletext Visual Display Interface, a parallel ASCII keyboard and a Centronics parallel printer interface, contained on Acorn's Versatile Interface Board.

Enter the DOS by pressing the delete (DEL) key on the keyboard (and hold it, if your keyboard is not Acorn's) and the reset (BREAK) key. The system will display on the screen:-

```
        Acorn Dos

        *_
```

The * is the DOS prompt, indicating that it is waiting for the user to type in a command. To the right of this is the flashing cursor at which characters will appear. After a character has been typed on the keyboard, it will be displayed on the screen and the cursor will move one space to the right. Incorrect characters may be corrected by typing a delete key, in which case the character to the left of the cursor is erased and the cursor moved back one space. An entire line may be thrown away by typing control x (hold the CTRL key down and type x or X), in which case a new line will be started on the screen. When you think the line of text is in a satisfactory form for the DOS, typing RETURN will present it to the DOS for approval. The command will be executed (if possible) and the * prompt will usually reappear, prompting for more commands. If the DOS detects an error in the command, a complaining message will appear, e.g. :-

```
        Syntax ?
```

indicates that the DOS recognises the command, but does not approve of the way it is written.

The DOS directly recognises the following commands, all of which can be abbreviated by entering enough characters to distinguish each from other commands, followed by a full stop. Spaces may be used between items (and one is usually obligatory) and leading zeroes in numbers are not required. The commands are listed together with their shortest possible abbreviations.

CAT X   .

This command causes the catalogue of drive X (or the current drive, if X is not present) to appear on screen. A typical catalogue will look something like this:-

```
      *CAT0
      Basic disk vl drive 0 qual s opt 0
       :    #BASIC        #LISP
      s:     ZOMBY
```

The title of the disk is Basic disk vl; we are currently using drive 0 and qualifier s and the disk 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 by qualifier and file name when it is output. The character X can be omitted or it must be 0 or 1. If not, then

Drive ?

will appear.

DIR X D.

This command causes the catalogue of the drive specified as in CAT to be loaded into memory at hex address 2000. The command is often used to wait until completion of the previous operation.

DRIVE x DR.

This command sets the current drive to x, where x can be either 0 or 1 or omitted completely (for compatibility with CAT and DIR). If x is neither of these the error

Drive ?

will appear, if x is multi-character a

Syntax ?

will appear. Drive 0 is set on reset.

SETY

This command sets the current qualifier to Y, where Y can be any character. All following file access will use only the Y portion of the catalogue. Qualifier space is set on reset.

USEY

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

Definition <s>

The symbol <s> will stand for a string of characters. If the required string does not contain any spaces and does not begin with a " quote, it may be typed directly. If not, it must be enclosed in " quotes, with " quotes in the required string typed as "".

Examples

Required string	<s> form
FRED	FRED or "FRED"
"FRED	"""FRED"
"FRED"	"""FRED"""
"	""""
hello	hello or "hello"
a b c	"a b c"
	" "

A valid <s> form can be surrounded by spaces, and has an even number of " quotes in it. When <s> is used as a filename, the string must be fewer than 8 characters in length, otherwise a

Name ?

message is produced.

TITLE <s>   T.

This command sets the title of the disk in the current drive to the first 13 characters in <s>, filling with spaces if there are fewer than 13. It is often useful 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 disk is protected, a

```
        Disk prot
```

message is produced.

OPTION X   O.

This command sets the option of the disk in the current drive to the number X. If the entire disk is protected, a

```
        Disk prot
```

message is produced. The option enables automatic use of the file BOOT in qualifier space on drive 0 when the system is reset. The automatic start may be totally defeated by pressing DELETE while the system is reset, and is enabled by pressing space. 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 produced on reset if BOOT does not exist.

MON   M.

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

NOMON   N.

This command disables messages.

LOAD <s> XXXX   L.

This command loads the file <s> on the disk in the current drive from the current qualifier into memory starting at address XXXX. The address XXXX may be omitted when the file's own address is used. If the file is not found a

```
        File ?
```

message is produced.

Examples

```
        LOAD                  file name is space
        LOAD FRED
        LOAD "FRED"
        LOAD FRED 1000
        LOAD "FRED" 1000
```

RUN <s>1 <s>2   R.

This command loads the file <s>1 on the disk in the current drive from the current qualifier into memory at the address for the file. The <s>2 is turned back into the original string form and stored in memory from hex address 0140 upwards, terminated by a carriage return.

Examples

```
        RUN FRED
        RUN "FRED"
        RUN FRED jim
        RUN "FRED" jim
        RUN "FRED" "jim 1"
```

SAVE <s> XXXX YYYY ZZZZ

This command saves the block of memory between XXXX (start address) and YYYY (end address plus 1) to the file <s> in the current qualifier of the directory. If entire disk is protected a

    Disk prot

is produced, and if <s> is locked a

    File prot

message is produced. If <s> exists and is not locked, it is deleted. Starting at the extreme outside of the disk (track 0), a gap large enough to contain the block is searched for; if it cannot be found a

    Disk full

message is produced, if there are already 31 files in the catalogue a

    Full

message is produced. The ZZZZ address is the execution address which defaults to XXXX if not supplied.

DELETE <s>   DE.

This command deletes the file <s> in the current qualifier from the current disk's catalogue. If entire disk is protected a

    Disk prot

message is produced, if the file is not found a

    File ?

message is produced, if the file is protected a

    File prot

message is produced.

GO XXXX

This command causes the machine code subroutine at XXXX to be entered. If XXXX is not given, the last known execution address is used. Warning : th% execution !ddress is destroyed by CAT and INFO does not set the execution address.

LOCK <s>   LO.

This command locks the file <s> in the current qualifier on the current disk. If entire disk is protected a

    Disk prot

message is produced, if the file is not found a

    File ?

message is produced.

UNLOCK <s>   U.

This command unlocks the file <s> in the current qualifier on the current disk. If entire disk is protected a

    Disk prot

message is produced, if the file is not found a

    File ?

message is produced.

INFO <s>   I.

This command produces information about the file <s> in the current qualifier on the current disk. If the file is not found a

        File ?

message is produced. The information is in the following form:-

```
current       file              load      execution  length     start
qualifier  :  lock  FILNAME     address   address    in bytes   sector
```

For example, the information on the files on the example catalogue could be

```
 : #BASIC    C000 C2B2   01000 002
 : #LISP     2800 2800   02000 012
s:  ZOMBY    3000 C2B2   00312 032
```

EXEC <s>   E.

This command reads the bytes from the file <s> in the current qualifier on the current disk as if they came from the keyboard. If the file is not found a

        File ?

message is produced. The file is automatically closed after all the bytes in it have been read. EXEC uses calls to OSFIND, OSSHUT and OSBGET.

If the command is not one of the above, then it is treated as a RUN command file name in qualifier space of the disk in drive 0. This is when the <s>2 string is the most useful; assuming the existence of EDIT, EDIT "fred" is a valid command. If the command cannot be found a

        Command ?

is produced. With the example catalogue, valid commands are BASIC or LISP, which will be loaded and executed.

Error messages

Disk error 08 (clock error)

During a disk read operation a clock bit was missing.

Disk error 0A (late DMA)

During a disk transfer the processor did not respond fast enough, most likely due to a faulty disk interface card.

Disk error 0C (ID field CRC error)

The cyclic redundancy check derived from the data read back, differed from that stored on the disk.

Disk error 0E (data CRC error)

The cyclic redundancy check derived from the data read back, differed from that stored on the disk during a disk read.

Disk error 10 (drive not ready)

During a transfer the disk stopped rotating. Often a badly-inserted disk.

Disk error 14 (track zero not found)

Controller failed to find track zero. Often an unformatted disk.

Disk error 18 (sector not found)

Controller failed to find required sector. Either a corrupted or an unformatted disk.

Utility Programs Disk

A diskette with utility programs is available for use on the 6502 systems. These programs may change with different issues of the DOS ROMs or with the type of drive and so they may not transfer from one system to another. If a disk 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 disk will have a BOOT file which will cause a description of the disks programs to appear on the screen at system start up. Currentley this screen is as follows:-

```
Hello, this is an Acorn System 3.

This message was automatically loaded.

This disk has 4 usefull programs:-

INFALL info on all files
COPY copy from drive 0 to 1
COMPACT garbage collect a disk
FORMAT initialise a disk

These programs are used by typing
their name. INFALL works immediately.
COPY and COMPACT wait until a key is
pressed. FORMAT requires the word YES
to be typed.
```

In a single drive system the disk with the utilities on will need to be removed and the disk to be operated upon inserted, the programs contain a waiting state for this operation where appropriate. Dual drive systems will usually have the utilities on a disk in drive 0 and the user disk in drive 1. Select drive 1 using the CAT 1 or DRIVE 1 commands before running the utility. The original utility disk supplied with a system is best kept write protected so that it can not be accidentally formatted or copied on to.

INFALL

Infall uses the INFO command in the DOS to give load address, run address anddisk sector information on all the files on a disk.

COPY

Copy is for use on dual drive systems, it is loaded on entering the word COPY and the system then stops. After placing the scource disk in drive 0 and the destination disk in drive 1 pressing the space bar will cause a complete copy to occur.

COMPACT

After saving and then deleting files on a disk unused sectors will appear where a file was deleted and no files of the same lengh have since been saved. Compact copies files one after the other into RAM and then re-saves them with no gaps between them.

FORMAT

Format initialises new disks with the track and sector format. The disk requiring formatting should be inserted and checked with CAT. Insert the standard disk and type FORMAT. The program prompts with a message, which will give time to remove the standard disk and insert the disk to be formatted. The characters Y, E and S, must then be typed with no mistakes, before the formatter will operate. The program initialises the entire disk and clears the catalogue, then verifies the entire disk. If an error occurs during verification, the formatter should be tried a few more times. A protected disk will produce a

Disk prot

message.

The Disk Drive

The drive will be similar to the one illustrated on the next page. It consists of a cast metal body with a plastic face plate. The main drive motor rotates the diskette at 300 r.p.m. and a stroboscopic ring allows the speed to be checked. The diskette should be inserted with the label at the top right, the label being the last thing to enter the drive. Shutting the door centres the diskette and clamps the disk onto the driving hub. Reading and writing data is done by magnetic heads which must be in close contact with the disk. On some drives the head is 'loaded' when the door is shut, while on other drives a solenoid loads the head when the drive is in use.

A stepping motor moves the head radially across the disk, its position is controlled by single step pulses issued by the controller together with a physical reference signal from the drive when the head is at track zero. An optical system senses the position of the index hole in the diskette. The L.E.D. on the front panel of the drive will be lit when the drive is in use. One or more printed circuit boards on the drive carry the control and interface circuitry which can stop and start the drive motor interface between digital signal levels and the signals to and from the head. External connections to the drive are by a 34 way ribbon cable carrying control and interface signals and a 4 way power plug.

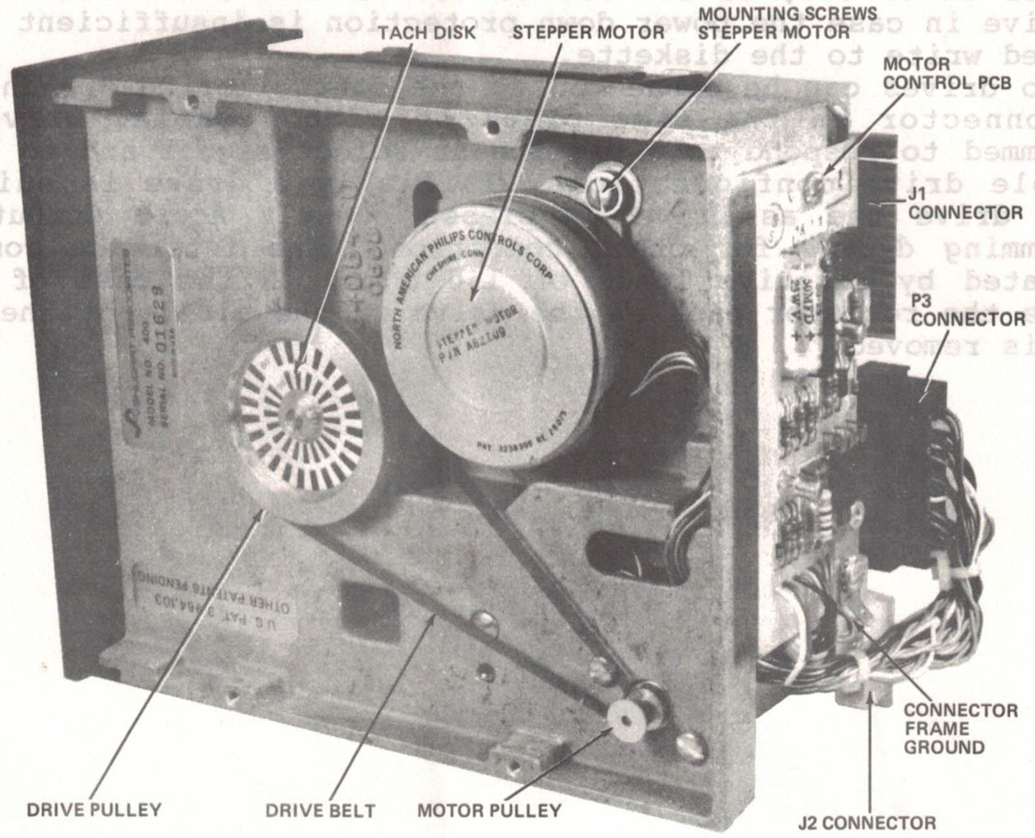
34 way interface connector

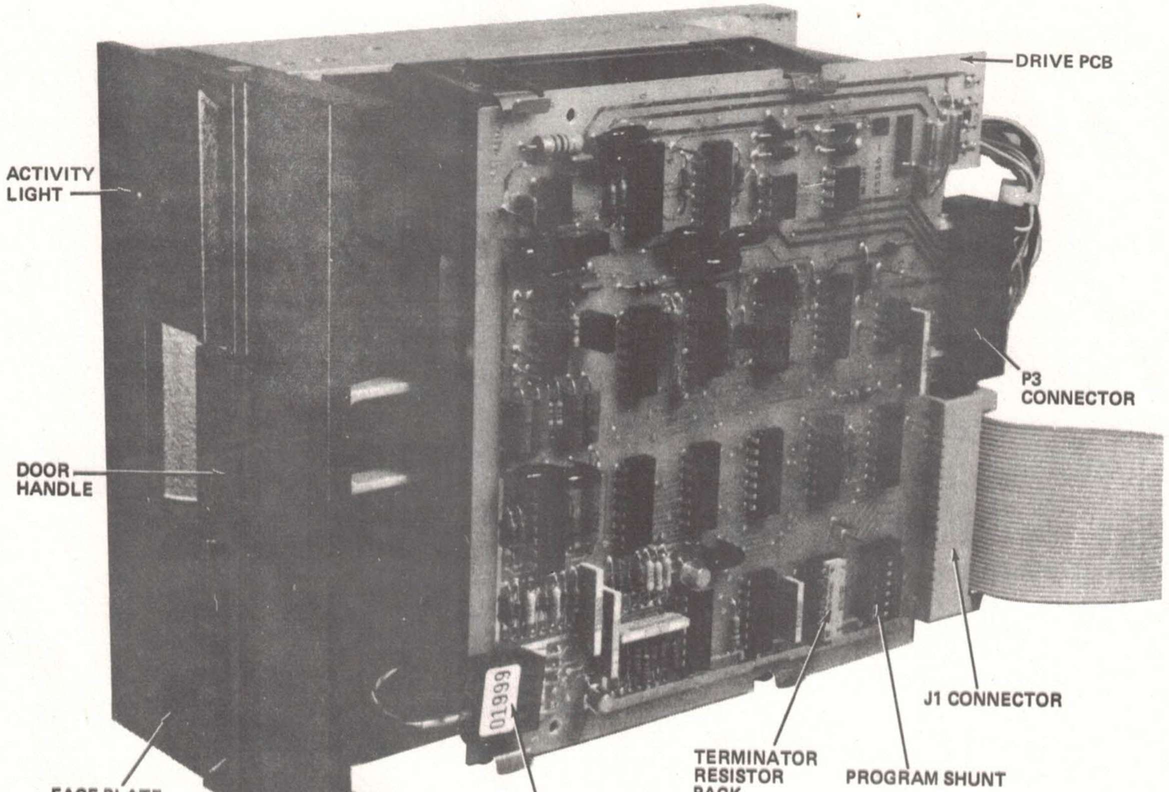
34	Not used	33	-All ground
32	Side select	31	
30	Read data	29	
28	Write protect	27	
26	Track zero	25	
24	Write enable	23	
22	Write data	21	
20	Step	19	
18	Step direction	17	
16	Motor on	15	
14	Select 2	13	
12	Select 1	11	
10	Select 0	9	
8	Index	7	
6	Select 3	5	
4	Spare	3	
2	Not used	1	

4 way power connector

1	+12 V D.C.
2	0 V
3	0 V
4	+5 V D.C.

A Disk Drive


TACH DISK
STEPPER MOTOR
MOUNTING SCREWS
STEPPER MOTOR
MOTOR
CONTROL PCB
J1
CONNECTOR
P3
CONNECTOR
CONNECTOR
FRAME
GROUND
DRIVE PULLEY
DRIVE BELT
MOTOR PULLEY
J2 CONNECTOR


DRIVE PCB
ACTIVITY
LIGHT
P3
CONNECTOR
DOOR
HANDLE
J1 CONNECTOR
TERMINATOR
RESISTOR
PACK
PROGRAM SHUNT
FACE PLATE
P4 CONNECTOR

Power requirements vary for difderent drives, but in general 5V @ 600mA and 12V @ 900mA regulated within 5% is required per drive. The 12V supply should be capable of supplying up to 1800mA for a short time when a main drive motor is started. When turning the power supplies on or off, it is wise to not to have a diskette inserted in the drive in case the power down protection is insufficient to stop an unwanted write to the diskette.

Two drives can be connected to the controller card each with it's own connector on the same piece of ribbon cable. Drives can be programmed to respond to a particular select signal and to operate in multiple drive configurations. Initially a drive is suitable for single drive use as drive 0; please consult Acorn Computers about programming drives for other conditions. The interface connector is terminated by resitive loads of 150 ohms, in the case of two drive systems the resistor pack on the drive in the middle of the interface cable is removed.

The Controller Card

The Acorn Disk Controller card uses an Intel 8271 integrated circuit to minimize both hardware and software overheads involved in using mini-floppy disk drives. The controller is a Eurocard, that is 100 by 160 mm, and it plugs on to the standard Acorn bus via a 64 way connector. A 34 way ribbon cable plugs on to the card in order to connect to the drive(s).

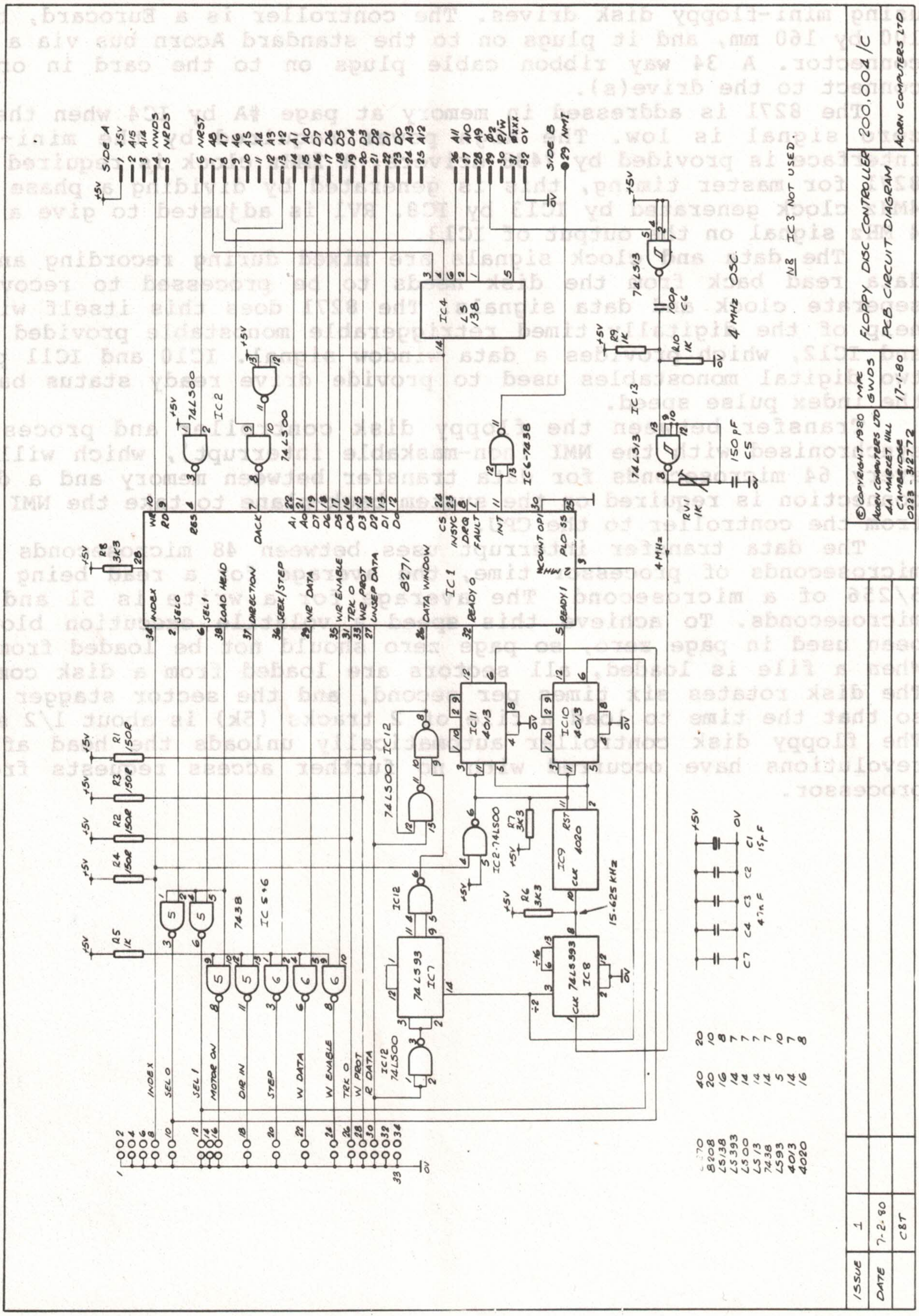
The 8271 is addressed in memory at page #A by IC4 when the block zero signal is low. The high power required by the mini-floppy interface is provided by 7438 drivers. A 2MHz clock is required by the 8271 for master timing, this is generated by dividing a phase locked 4MHz clock generated by IC13 by IC8. RV1 is adjusted to give a stable 4 MHz signal on the output of IC13.

The data and clock signals are mixed during recording and thus data read back from the disk needs to be processed to recover the seperate clock and data signals. The 8271 does this itself with the help of the digitally timed retriggerable monostable provided by IC7 and IC12, which provides a data window signal. IC10 and IC11 provide two digital monostables used to provide drive ready status based on the index pulse speed.

Transfer between the floppy disk controller and processor is synchronised with the NMI (non-maskable interrupt), which will occur every 64 microseconds for data transfer between memory and a disk. A connection is required on the system back-plane to take the NMI signal from the controller to the CPU.

The data transfer interrupt uses between 48 microseconds and 56 microseconds of processor time, the average for a read being 48 and 5/256 of a microsecond. The average for a write is 51 and 5/256 microseconds. To achieve this speed a volatile execution block has been used in page zero, so page zero should not be loaded from disk. When a file is loaded, all sectors are loaded from a disk complete. The disk rotates six times per second, and the sector stagger is +3, so that the time to load a file of 2 tracks (5k) is about 1/2 second. The floppy disk controller automatically unloads the head after 10 revolutions have occurred with no further access requests from the processor.

Controller Card Circuit Diagram



Controller Card Parts List

PCB	Acorn Computers 200,004	
IC1	8271	disk controller
IC2	74LS00	TTL nand gate
IC3	Component reference not used	
IC4	74LS138	TTL decoder
IC5	7438	TTl nand gate
IC6	7438	TTL nand gate
IC7	74LS93	TTL counter
IC8	74LS393	TTL counter
IC9	4020B	CMOS counter
IC10	4013B	CMOS flip-flop
IC11	4013B	CMOS flip-flop
IC12	74LS00	TTL nand gate
IC13	74LS13	TTL schmitt trigger
C1	15 uF electrolytic capacitor	
C2,3&4	47 nF capacitors	
C5	180 or 220 pF capacitor	
C6	100 pF capacitor	
C7	47 nF capacitor	
R1-4	150 ohm resistors	
R5	1 Kohm resistor	
R6,7&8	3.3 Kohm resistors	
R9&10	1 Kohm resistors	
RV1	1 Kohm preset resistor	

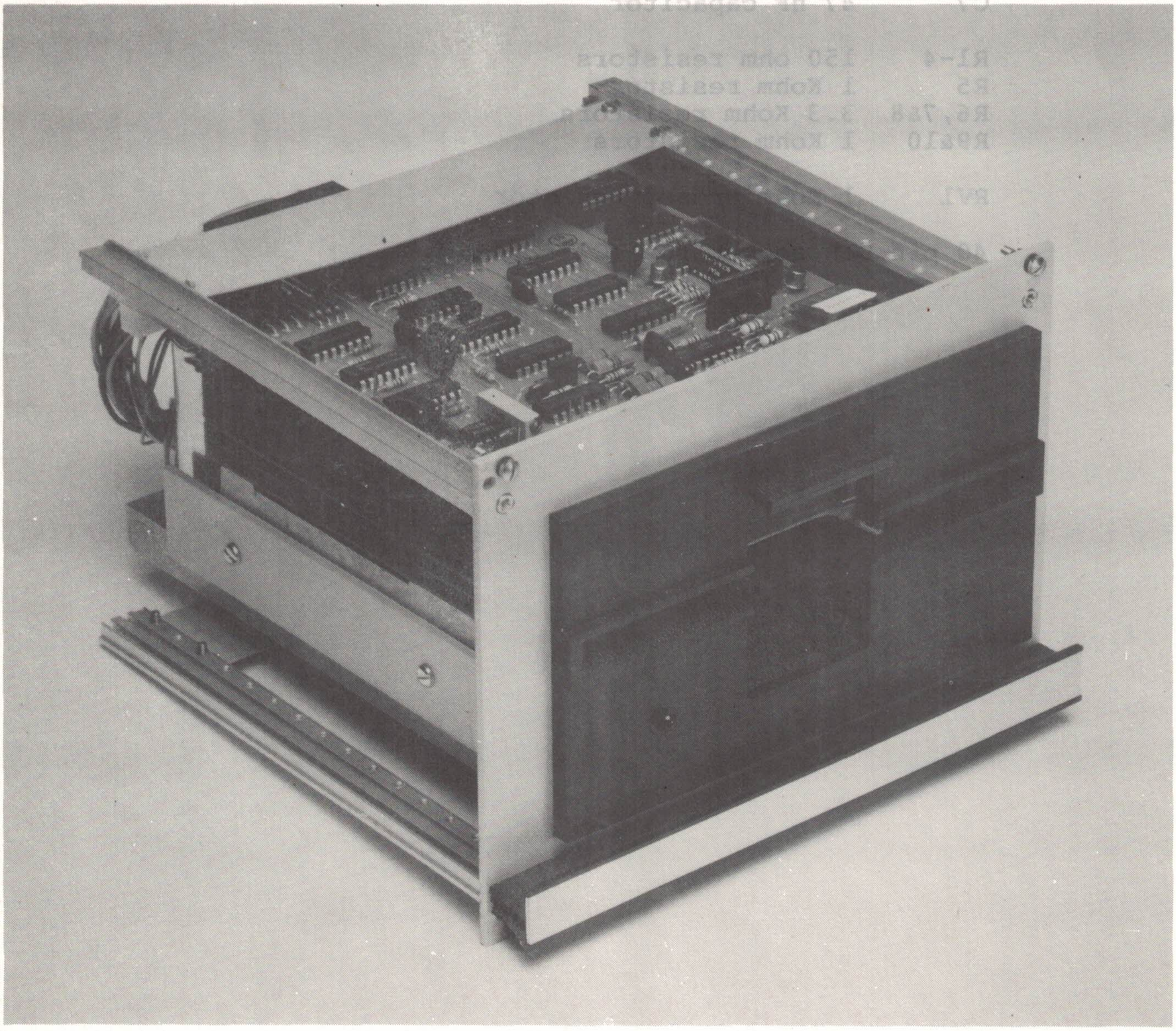
40 pin DIL socket for IC1

The Module

Each disk drive is carried in a seven inch Eurocard module made up of the following parts:-

(1) Front panel
(2) Handle
(3) 4 guide rails
(4) Side plate
(5) Top and bottom straps
(6) Side clamps
(7) Screws, brackets etc.

The drive is placed through the hole in the front panel and two side clamps then hold it in place. In a single drive system the controller card is mounted in the left side of the module and it plugs into the end socket on an Eight Card Backplane. In a dual drive system the two modules are placed in the top half of a double height rack, and the controller card is plugged into the backplane seperately. Four plastic module guides are used to hold a module into the card frame.



Operating System Specification.

This is the definition of the environment for user programs provided by all Acorn operating systems, e.g. Acorn DOS, Acorn COS and ATOM. A set of indirect vectors are stored in page 2, least significant byte lower in memory:-

0200	NMIVEC	nmi routine entry
0202	BRKVEC	brk routine entry
0204	IRQVEC	irq routine entry
0206	COMVEC	operating system command line interpreter
0208	WRCVEC	write character subroutiine
020A	RDCVEC	read character subroutine
020C	LODVEC	load program subroutine
020E	SAVVEC	save program subroutine
0210	RDRVEC	read arguments subroutine
0212	STRVEC	set arguments subroutine
0214	BGTVEC	read byte from random file
0216	BPTVEC	write byte to random file
0218	FNDVEC	find random file
021A	SHTVEC	shut random file

Page FF00 contains the following calls:-

```
FFCB   OSSHUT   JMP (SHTVEC)
FFCE   OSFIND   JMP (FNDVEC)
FFD1   OSBPUT   JMP (BPTVEC)
FFD4   OSBGET   JMP (BGTVEC)
FFD7   OSSTAR   JMP (STRVEC)
FFDA   OSRDAR   JMP (RDRVEC)
FFDD   OSSAVE   JMP (SAVVEC)
FFE0   OSLOAD   JMP (LODVEC)
FFE3   OSRDCH   JMP (RDCVEC)
FFE6   OSECHO   JSR OSRDCH
FFE9   OSASCI   CMP # $0D
FFEB            BNE OSWRCH
FFED   OSCRLF   LDA # $0A
FFEF            JSR OSWRCH
FFF2            LDA # $0D
FFF4   OSWRCH   JMP (WRCVEC)
FFF7   OSCLI    JMP (COMVEC)
```

Interrupts

On NMI any operating system interrupts are serviced, otherwise

```
PHA
JMP (NMIVEC)
```

is executed.
On IRQ/BRK

```
STA $00FF
PLA
PHA
AND # $10   check B flag
BNE BRK
LDA $00FF
PHA
```

```
                JMP (IRQVEC)
        BRK     LDA $00FF
                PLP
                PHP
                JMP (BRKVEC)
```

Reset

On reset the operating system initialise pointers NMIVEC to SHTVEC to point to it's own handling programs.

Zero Page

Acorn Operating systems use memory in Zero Page starting from location $00AC up to location $00FD for their own purposes. Location $00FE contains the code of an ASCII character which is not sent to the printer, this is initialised to $0A a line feed. Location $00FF is used for IRQ/BRK service as above.

Subroutine Actions

OSCLI

This is a subroutine which 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.

OSWRCH

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

OSCRLF

This subroutine generates a line feed and then a carriage return using OSWRCH. A will contain $0D on exit.

OSASCI

This subroutine is as OSWRCH except that a carriage return will be output as line feed carriage return.

OSECHO

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

OSRDCH

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

OSLOAD

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

```
X --->  -----------
        |         |          string of characters
        -----------  --->terminated by $0D
        |         |          which is file name
        -----------
        |         |      address in memory of
        -----------   the first byte of
        |         |   the destination
        -----------
        |         |   bit 7 = 1 : use above address
        -----------           0 : use file's address
```

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

OSSAVE

This subroutine saves all of an area of memory to a specified file. On entry X -ust point to the following data in Zero Page:-

```
X --->  -----------
        |         |      string of characters
        -----------   --->terminated by $0D
        |         |      which is file name
        -----------
        |         |   address in memory where
        -----------   the file is to be
        |         |   reloaded to
        -----------
        |         |   address of machine code
        -----------   to enter when data
        |         |   is RUN
        -----------
        |         |   start address in memory
        -----------   of the data
        |         |
        -----------
        |         |   end address +1
        -----------   of the data
        |         |
        -----------
```

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

OSRDAR

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

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. X and Y are retained.

OSSTAR
This subroutine sets the value of a file's pointer. On entry X points to zero page locations contaning the value and Y contains the file's handle. X and Y are retained.

OSBGET
This subroutine returns the next byte from a random file. On entry Y contains the file's handle. X and Y are retained and the byte is returned in A. The file's sequential pointer is incremented after the byte is read. Errors are met with a BRK.

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

OSFIND
This subroutine returns, in A, a handle for the file whose name is pointed to by

```
X --->  ----------
        |        |      string of characters
        ----------  --->terminated by $0D
        |        |      which is file name
        ----------
```

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.

OSSHUT
This subroutine closes the random file whose handle is in Y. If a handle of zero is supplied all random files are closed. After a file has been closed, the same handle may eventually be used for a different file.

Random Files.

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 several random files active in this DOS up to five. These active files will each have a one byte internal name which will be referred to as a "file handle". Handles are allocated by OSFIND, cancelled by OSSHUT and passed as arguments to all other routines. Proper file handles are all non zero: use of zero as a handle causes 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, ... The sequential pointer holds the number of the next byte to be read or updated. As OSBGET and OSBPUT access 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 disk is set aside for it. The new file will have an extent of 0 and R will show the size of the disk 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 disk 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 the case that

```
          0 <= P <= E <= R
```

and R is a multiple of 256

OSRDAR and OSSTAR provide a means for interrogating P, E and R, and updating P. The ability to change P gives the user random access and 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 hex FF bytes.

OSBPUT writes bytes to a file. On entry A holds the byte to be written and Y holds the 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.

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

Both OSBPUT and OSBGET behave specially if used with a file handle of zero. OSBPUT writes to the output stream using OSASCI and OSBGET reads from the input stream using OSECHO. Note that it is legal to use calls to both OSBGET and OSBPUT for a single file, but that excessive use of OSSTAR to update P may cause a lot of disk transfers.

OSSHUT closes the file whose handle is in Y. This involves writing out any buffers that contain data that has been changed, and updating the main disk catalogue to show the length of the file. A zero handle in Y will cause all sequential files to be closed.

OSFIND opens files for input or output. To call OSFIND, it is necessary to provide a block of store containing the file name (terminated by hex 0D, carriage return). Two bytes in page zero point to this block, and the X register contains the address of the pointer. If the carry flag is set the file named must already exist and E and R will be set to its actual size. If the file is not present on disk 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 a file with the given name already exists, the old file 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 be the same as the old. 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 disk, 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 modified by the current drive and qualifier.

The region of store from $2200 to $27FF is used for file buffers and control blocks. To reduce chance of disk 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 be safe to shut files first. An unrecommended, but possibly useful, action would be to set byte $00C0 to 0, which would cause files to be forgotten without changing the disk.

Control Codes

The following list gives the minimum set of control codes that perform special functions with an Acorn operating system. They are all available from the keyboard, by typing CTRL with the specified key, or from programs by printing, say PRINT $2:

STX (CTRL-B, 2) Start printer

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 VDU until receipt of an ETX code.

ETX (CTRL-C, 3) End printer

Ends the printer output stream.

ACK (CTRL-F, 6) Start screen

Starts the output stream to the VDU screen.

BS (CTRL-H, 8) Backspace

Moves the cursor back one position.

HT (CTRL-I, 9) Horizontal tab

Moves the cursor forward one position.

LF (CTRL-J, 10) Linefeed

Moves the cursor down one line.

VT (CTRL-K, 11) Vertical tab

Moves the cursor up one line.

FF (CTRL-L, 12) Formfeed

Clears the screen, moves the cursor to the top left-hand corner.

CR (CTRL-M, 13) Return

Moves the cursor to the start of the current line.

NAK (CTRL-U, 21) End screen

Ends the output stream to the VDU; the only code recognised when in this condition is ACK.

RS (CTRL-^, 30) Home cursor

Moves the cursor to the top left-hand corner of the screen.

DEL (DELETE,127) Delete and backspace

Backspaces the cursor one character and erases that character.