SOLIDISK DDFS USER MANUAL

1. INSTALLATION PROCEDURE FOR THE STL DDFS KIT

1) Check the MOS version.
2) Install the ICs.
3) Connect the disk drive.
4) Run the final test.

In comparison with Acorn's DFS and other leading DFS's (Microware, Opus, Watford's, etc.), SOLIDISK's DDFS is the simplest of all to install.
If you already have some experience of the computer hardware it is not necessary to follow too strictly the installation procedure and verification of each step.
The time it takes to install Solidisk DDFS varies from five minutes when fitted by an expert to around half an hour for the absolute beginner.
If your BBC computer is an issue 3 or earlier you will also have to do a standard modification to the computer PCB.

1.1.1 CHECK THE MACHINE OPERATING SYSTEM ROM:

Solidisk DDFS (and any other DFS upgrade) requires a BBC Model B fitted with the machine operating system version 1.20. To find out which MOS (Machine Operating System) your machine has, switch on the computer and enter:

```
*FX 0 <RETURN>
```

You should see:

```
OS 1.20
>
```

Do not attempt to install STL DDFS if your MOS is any other version (e.g. EPROM .10 etc.). Go to your nearest BBC dealer and purchase a new MOS ROM.

1.1.2 CHECK THE CONTENTS:

Check the contents of the disc upgrade kit:
1) The STL DDFS 1.4 ROM with a sticky label.
2) The WD 1770 FDC board measuring 38mm. x 55mm. (IC 78).
3) 2 x 7438s (IC 79 and 80), one is wired.
4) Two metal jumpers (J86 and J87).

1.2 INSTALLATION SEQUENCE

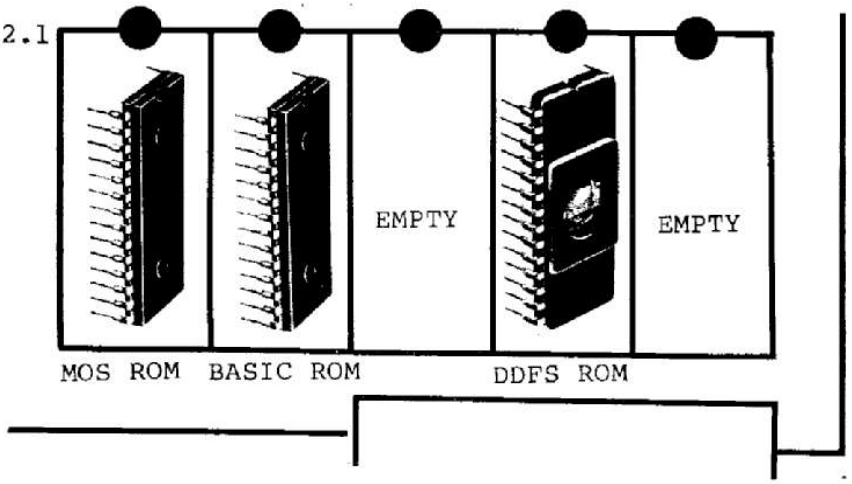
1) Install the DDFS ROM.
2) Install the two jumpers.
3) Install the FDC board
4) Install the two buffers 7438's.

1.2.1 INSTALL THE DDFS ROM:

Switch off the computer.
Undo the four screws labelled 'FIX', two at the rear of the computer and two under the keyboard.
Remove the top of the computer case.
Undo the two bolts found either side of the keyboard and slide the keyboard toward you.
Note the row of 5 x 28 pins ROM sockets at the bottom right corner of the computer PCB. The leftmost of them is the MOS ROM. The

other four sockets are Sideways ROM sockets. If you have not installed any other Sideways ROM(s) since the machine was purchased, the ROM next to the MOS is the BASIC ROM and there will be three empty sockets on the right. Solidisk DDFS ROM can be installed in any of these three sockets, preferably the fourth from the left, IC100.

figure 2.2.1



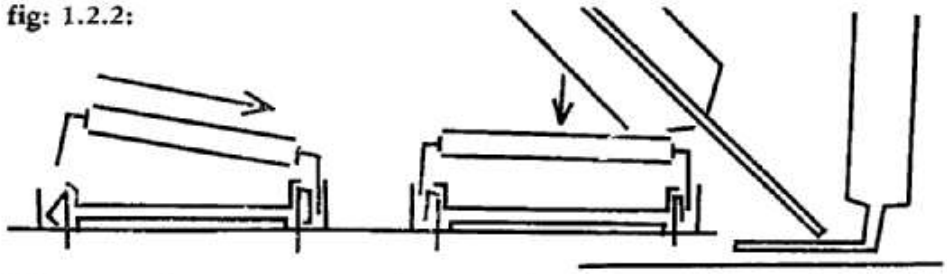
A quick word about Sideways ROMs for those who are not yet familiar with them: Even if you never add Sideways ROMs to your machine it is important to know that the Sideways ROMs (of which STL DDFS is one) are machine code programmed into permanent memory chips called ROM (Read Only Memory, factory programmed and uneraseable), similar in appearance to your MOS or BASIC ROMs (colour black with no sticky label). EPROMs (Eraseable Programmable ROM) are similar in appearance to STL DDFS 1.4 ROM and can be erased, reprogrammed by you with an Eprom Eraser, such as Solidisk's UVIPAC and an Eprom Programmer, such as Solidisk's UVIPROM. ROMs are cheaper and more secure than EPROMs but cannot be corrected if there is any error whereas EPROMs can be erased and reprogrammed. Commercially available Sideways ROMs are plentiful: View and Viewsheet, wordprocessor and electronic spreadsheet from Acornsoft, Scribe from Merlin, Disc Doctor and Graphic Extension from Computer Concepts, Starbase, etc., to cite a few. There are more than 35 compatible Sideways ROMs to choose from, catering for from AI (Artificial Intelligence) to Machine Code Monitor.

Up to four Sideways ROMs can be fitted in your BBC Computer, BASIC and STL DDFS must be installed, leaving only two other choices, possibly a Flash Cable and Solidisk Sideways RAM.

As Sideways ROMs are so important to add value to your computer, a range of add-ons intended to facilitate the use of Sideways ROMs are also available: Sideways ZIF from Watford Electronics, Viglen Cartridge System featuring easy change of the Sideways ROM in use, Sidewise ROM extension board and Solidisk Sideways RAM system to cite a few (Solidisk Sideways RAM system is compatible with Watford's ZIP, Viglen's cartridge but not with Sidewise Extension; check with Sales Office about compatibility). Solidisk has also introduced recently a 'Flash Load' (FL Cable) system using Mini ROM cartridges and retailed at £9.95. You may ring the Sales Office for information.

Refer to fig. 2.2.1 to orientate the DDFS ROM. With the notch facing North (to the rear of your machine), insert the ROM into IC socket 100 (or another sideways ROM socket if you prefer). To avoid damaging the IC legs never use brute force! Hold the IC with your thumb and index finger and if you are right-handed line the right row of IC legs against the right row of socket holes and lower the left row of IC legs. If necessary use a little bit of pressure to bring the left legs in line with the remaining socket holes. When, and only when, you are sure that all IC legs are straight above the socket holes, press down gently but firmly.

fig: 1.2.2:



Right row first. Line up left row. Press down. Correct bent pins

In case of difficulty because the IC pins are bent or splayed out/in too much to be lined up properly in the socket, straighten them in the following way:
Touch your hands to a large metallic object to discharge any static electricity on your body, place the IC legs on a flat surface such as the kitchen table. Using a spatula (such as the one you use for pies) press the IC legs against the flat surface, as shown in the sketch above; bend the IC forward or backward to straighten the IC legs or bring them more vertical to the IC.

Repeat the previous operation.

1.2.2 CHECK THE DDFS ROM:
Ensure once more that the DDFS ROM is facing the same way as the other ROMs.
Switch on the computer.
You should see:

```
BBC Computer 32k
STL DDFS 1.4
BASIC
>
```

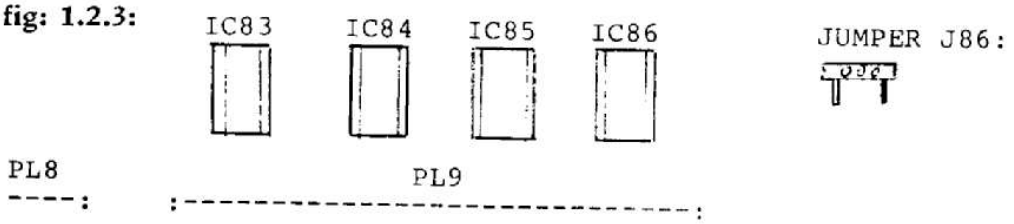
Switch off the computer.

If another language ROM (such as View) is in place in a higher priority socket than the BASIC ROM (i.e. to the right of the BASIC ROM), the installation may still be right although the display differs. Type in *BASIC <RETURN> then press the BREAK key to get the display above.

1.2.3. INSTALL THE JUMPERS:
Two jumpers are provided. They are simple metallic links that you place into the IC sockets to make electrical connection. Please TAKE CARE with this apparently simple operation.
The jumpers are to be placed in IC sockets 86 and 87.

1.2.3.1 LOCATING IC SOCKET 86:
Turn the keyboard over and leave it resting on the computer case and the PSU. Nearest to you are a row of connectors, accessible from beneath. They are numbered PL8 (the disk connector), PL9 (the printer connector), PL1O (the user port), PL11 (the 1 MHz bus) and PL12 (the Tube). Note the row of four empty sockets above PL9. IC 86 is the rightmost of them.

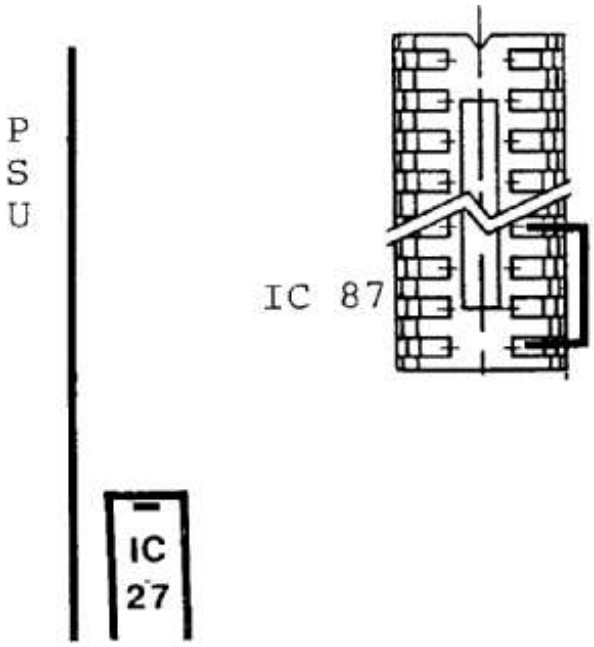
fig: 1.2.3:



Place one of the two jumpers between the top left corner (pin 1) and the 4 pin down on the left-hand side (pin 4).
Do not use brute force but still ensure that the contact is not loose. Avoid any action that could lead to the IC socket holes being enlarged which may result in poor contacts. The jumpers supplied are specially fabricated to comply with IC sockets. If, however, you mislay them, you may replace them with 0.6mm. single non-insulated conductors.

1.2.3.2. LOCATING IC SOCKET 87:

fig: 2.2.4:

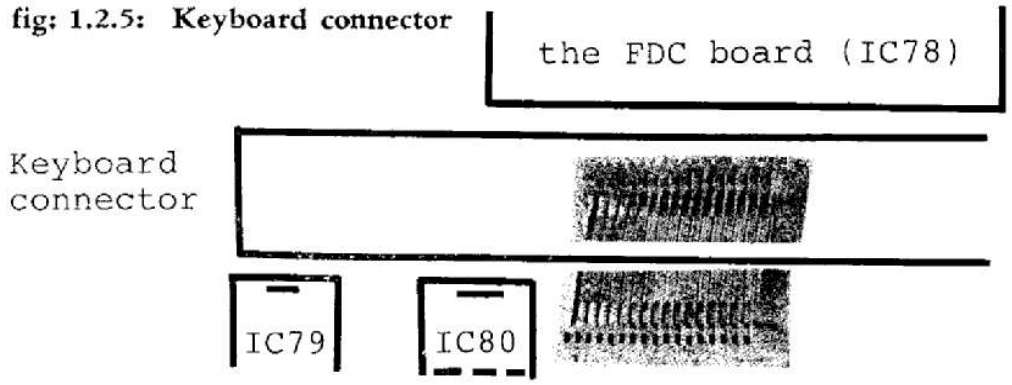


Replace the keyboard. About 3in. from the rear, 2in. from the PSU you should see a 16-pin socket with IC 87 written just above it. Apart from a small 8-pin IC this should be the one nearest to the top left corner of the computer PCB. Place the second jumper between the bottom right corner (pin 9) and the fourth pin up, right-hand side (pin 12). By convention, IC pins are numbered anticlockwise, from 1 at the top left corner down the left-hand side (pin 8 in this case) then bottom right corner (9) then up to the top right-hand side (16). Once more, ensure that the jumper is not loose.
1.2.3.3 Power on, you should see the previous display.

1.2.4 INSTALL THE FDC BOARD:
Remove any polystyrene chip from the 40 pins found under the FDC board. Ensure that all 40 pins are straight. Straighten them if necessary (see 2.2.1 above). Identify the word 'North' on the FDC board; it should be orientated later to the rear of the computer.

1.2.4.1 LOCATING IC 78:
IC78 lies just above the keyboard connector. If you find any difficulty in placing the FDC board into IC socket 78, you may detach the keyboard connector, but, once more, avoid brute force.
Avoid pulling the keyboard connector by the keyboard flat ribbon cable (important); instead use a small screwdriver. From the righthand side, and levering on the computer PCB, prise the translucent plastic plug about 5mm. to the right. You can now hold the plug with both thumb and index finger of both hands and lift it up gently.

fig: 1.2.5: Keyboard connector



1.2.4.2 INSTALL THE FDC BOARD IN IC 78 SOCKET:
Install the FDC board with as much care as with the DDFS ROM except that this time you cannot really see much of the legs of the board, but don't despair! The board is fitted on a specially made Jermyn blue socket to facilitate your job. You can 'feel' the board 'sink' a little when all its legs are lined up. Check by looking under the board from the right-hand side of the computer and from the rear, then press down gently with the index finger in the middle of the FDC board. It will 'snap' into its correct position without any effort. Replace the keyboard connector if you detached it earlier.

1.2.3.4 CHECK THE FDC BOARD AND JUMPER J86:
Power on the computer. You should see the same display as previously.
If the computer locks up, the FDC board or the keyboard connector or both have not been correctly placed. Switch off the computer and check. Now enter:

```
?&FE81=O <RETURN>
PRINT ?&FE81 <RETURN>

You should see:
0
>

If the number is not 0, but, say, 254. try:

?&FE81=15 <RETURN>
PRINT ?&FE81 <RETURN>
```

If the answer is still, say, 254, jumper J86 is loose. But if it is 255, or a bigger number than the previous result, one of the FDC board legs on the left-hand side is bent and out of the socket. Switch off the computer and check. Jumper 86 supplies the 8MHz clock to the FDC board, without it, the FDC cannot write into &FE81 hence change its contents.

1.2.5 INSTALL THE TWO BUFFERS 7438's:
The remaining two IC's in the kit are 7438's. One is wired across from pin 4 to pin 11 which will be IC 79, the other one will be IC 80.

1.2.5.1 LOCATING IC SOCKETS 79 and 80:
They are situated on the left-hand side of the keyboard connector South of the FDC board (see fig: 2.2.5).
Turn the keyboard over and leave it resting on the computer case. IC socket 79 is the leftmost with the number 'IC79' written on its left side. Place the 7438 with the wire in this socket and ensure the same orientation, i.e. notch to the North as with all other IC's of the computer. Hold the chip with the index and thumb (index to top, thumb to bottom for right-handed), line up right row of IC legs first, then bring the left row in line with a little pressure. When the chip is evenly lined up, press with the index finger in the middle of the IC. It should snap in with no effort at all.
Do the same thing with IC 80: place the 7438 (without wire) in the same orientation and insert it into socket 80.

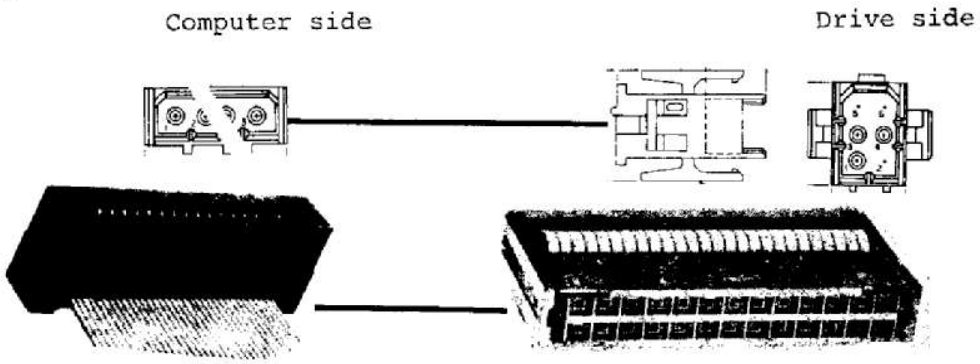
1.2.5.2 CHECK THE 7438's BUFFERS AND JUMPER 87:
The installation is now completed. Proceed to the final hardware check:

1) Connect the disk drive(s).
2) Read a blank disk (to make a known error).
3) Format this blank disc.

Note: While jumper J86 can be proven by writing into the FDC registers, jumper J87 can be proven positively only if the DFS can catalog a formatted disk. In our experience, of the 1150 DDFSs fitted in October 1984, bent pins and loose jumpers were responsible for all defective installations.

2 CONNECT THE DISK DRIVE:
Please refer to appendix A2 if you need more details. Incidentally, you need to connect the data cable to the 'Disk' connector and the power lead to the 'Auxiliary PSU outlet'.
The data cable is a flat grey ribbon cable with a red line on its righthand side; the power lead is round grey three-core wire as shown below:

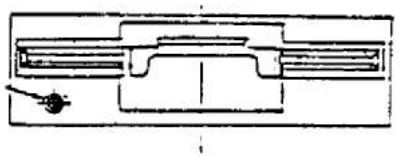
fig: 2.1:

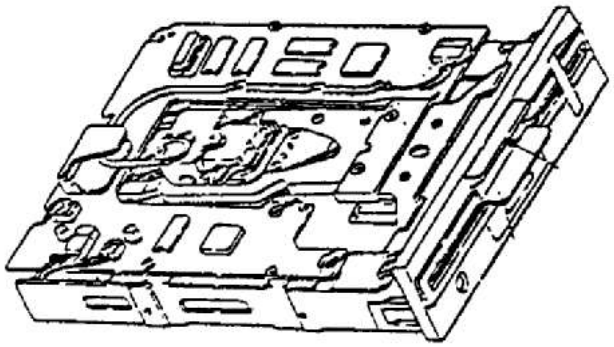


This figure relates only to Solidisk Mitsubishi drives.

Connect the power lead to the Auxiliary PSU socket. The connector has two cut corners, so wrong connection is impossible.
Connect the data lead to the disk header: open the latches to expose the header, with the bump on the data lead facing upwards. Position the data terminator socket against the pins of the header and push firmly. You will see that the two retaining latches will close on the socket with a small 'click'. Do not worry if the latches do not close properly. Simply ensure that the socket is straight and fully engaged.
Connect the cables to the drive side; the power lead has two cut corners so that it cannot be inserted the wrong way. The data lead has no key so ensure that the red marking line on the cable is to the right-hand side when you are facing the disk drive. Wrong connection WILL NOT DAMAGE your drive, so do not worry.
Remove the cardboard locking the drive door.
Close and open the drive door as many times as necessary until you feel that you are sufficiently familiar with the door mechanism. On the Mitsubishi the door is closed by pushing the latch downwards until it catches, opened by lifting the latch to the right. If you are using a disk drive for the first time, do this exercise at least 20 times before proceeding further.

fig: 2.2:





The drive door mechanism.

Place a BLANK diskette in the drive, push it in firmly until it is fully engaged and small 'click' is heard. Close the door.
If you are using a disk for the first time, repeat this exercise several times until you are familiar with the disk insertion and the way to open the drive door so that the drive ejects the diskette easily.

Leave the door closed and a blank diskette in the drive.
Now power on the computer.
The red LED (Light Emitting Diode) in front of the drive must be off within three seconds, otherwise check the data lead.
If the light persists or you can still hear the disk spinning, IC 79 has got a bent pin or the data cable on the disk drive is reversed. Switch off and check.
Now enter:

*.<RETURN>

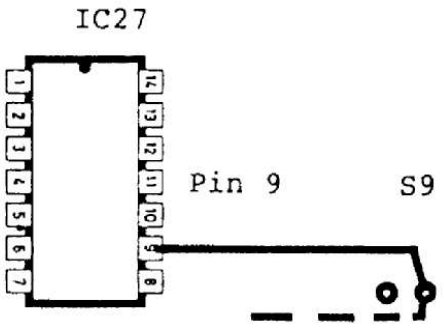
The LED on drive 0 (the one that has been connected) must be on, otherwise check that you have not connected the wrong disk drive (if you have twin drive) or IC 79 or the FDC board has not got a bent pin. The disk should spin for 10 seconds then the following message appears:

Disk fault.
>

If the drive does not stop after 15 seconds, press the BREAK key to stop the drive and check that the diskette has not jammed (the Mitsubishi spins up automatically when a disk is inserted or removed; you can hear it spinning).
Check the issue of your BBC computer. The issue number is written just above the FDC board:
Copyright 1982 203,000 issue ...
If your computer is of issue 1, 2 or 3 (all more than two years old) you will have to do a hardware modification to the computer PCB. Locate S9, which lies above the top left corner of the FDC board. A track on the PCB runs between its East hole and pin 9 of IC27, a 7438 situated on its left, then continues under the IC to the ground. As it is too difficult to correct the error under an IC, this track must be replaced by flying wire.

1) Use a sharp Stanley blade, break the track between two holes of S9.
2) Cut pin 9 of IC 27 as close as possible to the PCB. Bend it out horizontally.
3) Solder a wire from this pin to the East hole of S9.

fig: 2.3



Modification on
issues 1,2,3 only

Give it a second try. If you still cannot get the 'Disk fault' error message, switch off the computer and check FDC, IC79, IC80 and the jumper J87.
If the drive does not spin or the LED is not on, it is probably not J87. If the drive keeps on spinning for ever, it is probably J87.
Consult the Technical Staff between 4.30 p.m. and 5.30 p.m. for help.
Now proceed to the last hardware test.

Enter:
*EN. <RETURN>
*F80 <RETURN> or *F40 <RETURN>
*EN. is the abbreviated form of *ENABLE. *F80 or *F40 are formatting commands. Use *F80 with an 80-track disk drive such as the Mitsubishi, *F40 with a 40-track disk drive such as the TEAC 55A. The computer will display the numbers of the tracks being formatted,

in mode 7 in rows of 10. Four rows will be displayed for *F40, eight rows for *F80.
You can hear a regular clicking noise. Each time a track is formatted the drive head 'steps' to a new track.
When the last row has been displayed a blank line will be printed and the automatic verification will follow.
Again the numbers of tracks being verified are printed in mode 7 in rows of 10. A question mark (?) is printed for each difficult sector.
If you get 40/40 or 80/80 with no query, the installation is completely successful. If you get only a few queries, the installation is still quite successful, but you will have to 'query' your diskette supplier(s) or check for compatibility of your disk drive(s). Tried and tested disk drives include Mitsubishi and TEAC 55 series (A, F). More disc drives are being put through test by the Software Support Service. Diskettes tested include Verbatim MD525 to MD577, Wabbash DD/DS and Dysan.
If you did not hear the stepping noise, IC 80 has got a bent pin. Switch off and check. If you get 'Disk fault' as opposed to the Verify display, the FDC board has got a bent pin. Switch off and check. If you got a lot of '?'s while the disk was being verified, check with Solidisk Sales Office for compatibility.

IMPORTANT:
ON BOAROS ISSUE 4 TO 7 CUT THE LINK 59

3 FORMATTING AND VERIFYING DISKS

What is formatting ?

Formatting can be thought of simply as something that you have to do before you can use a new magnetic disk. However, a little more explanation is probably in order.

One of the advantages of disks compared with tape cassettes is that the filing system keeps track of the files it holds and can access them by name. For this to be possible, the files must be located in precisely defined positions on the magnetic surface and a record of this position must be kept against the filename.

Therefore the disk surface has to be "marked out" - defining the tracks with a pattern of 40 or 80 concentric circular rings, and defining the sectors with 10 or 16 radial lines, in single- and double-density respectively. Furthermore, in the Acorn Disk Standard, the catalogue (for filenames and positions) is located on Track 0, on the first and second sectors, counting from the index hole.

Since formatting consists of "marking out" the sectors and in fact filling them with dummy data, it effectively erases the whole contents of that surface of the disk. You should therefore always check any disk for files that you may want to retain before formatting it.

You can determine the contents of a disk by putting it in Drive 0 and typing *CAT. If it has been formatted you will get at least the catalogue header (see under *CAT) though no title or files may be shown. If it has not been formatted then the computer will eventually report "Disk fault".

Prior to actually formatting a disk, you have to decide between 40 and 80 tracks and between 10 and 16 sectors per track (i.e. single or double-density). The choice between 40 and 80 tracks is dictated in part by your disk drives - if they are 40-track drives there is no option. Similarly the choice between single and double density is dictated in part by your floppy disk controller - if it is (like the Acorn standard 8271 chip) only capable of single density, there is again no option. However, many modern (half-height) drives are capable of 80-track operation and may well have a switch to select 40 or 80 tracks. Also many of the modern floppy disk controllers that are sold for the BBC Micro are capable of double density operation and can be set for single or double density by a command typed in at the keyboard. For the Solidisk DFS, this is *DDFS.

In addition to the capabilities of his/her own disk system, the user must also consider those of any target or recipient system - as not all are capable of reading disks formatted at 80 tracks and even fewer are capable of reading double density and extended catalogs.

Clearly - like choosing double-sided over single-sided disk drives - choosing 80 over 40 tracks and double over single density can greatly increase the storage capacity of a disk. However, these same factors all increase the cost of both disk systems and disks. On balance, though, the user gains substantially from choosing the higher specification.

The format commands *F40 and *F80 require you to specify the drive number – which, on the Acorn/BBC Micro system, can be 0, 1, 2 or 3. According to a rather strange convention, Drive 2 is the second side of Drive 0, Drive 1 is the first side of any second physical drive unit while Drive 3 is the second side of that. It is therefore necessary to specify Drive 2 (or set Drive 2 to be the current Drive) in order to format the disk surface for the second side of a double-sided drive.

Verifying a Disk

The action of formatting usually includes verifying-which tests the magnetic surface for the quality of the recording - and this is true for the *F40 and *F80 commands of this DFS. The test compares the checksums stored with each sector with the contents of that sector.
Wherever the formatting process includes verification there is no need separately to verify a disk that has just been formatted. However, verification may be in order if you are having trouble reading a file on a disk or have other reason to suspect that it may be corrupted.
Unlike formatting there is no need to specify either the number of tracks or the recording density as the Solidisk DFS is "inte1ligent" - i.e. is capable of detecting these parameters and setting itself accordingly. This extends to the number of tracks because it is assumed that 80-track drives will be left switched to 80 tracks and automatic track skipping will be brought in as necessary to deal with 40-track disks.

4 USING DISKS TO LOAD AND SAVE PROGRAMS

The first thing that you will want to do with your new disk drives will be the same as you have been doing with cassettes.

Loading and Saving

The LOAD and SAVE commands (issued from e.g. BASIC) work much as the do with cassettes. However, there are certain differences:

1) The filename must not be longer than seven characters (compared with ten characters with cassette files). In addition to <space>, they must not now include * # ! - all of which have special meanings in disk filenames.
2) No messages show on the screen during loading and saving - as everything happens so rapidly anyhow.
3) It is not useful to type LOAD "" or CHAIN "" because, as files are added or deleted, they are not retained in a fixed sequence on the disk. Instead the separate, recorded catalogue allows them always to be recalled by name.

Further Simple Commands

As with cassettes the names of the files on a disk can be listed by typing *CAT. However, the response with a disk is almost instantaneous. Also, the header of the display shows additional information about the disk, such as a title and disk capacity.
The commands *LOAD, *SAVE and *RUN, usually used for machine code programs, and *SPOOL and *EXEC, used for ASCII text files, all work as they do with cassettes. They are described in detail in Chapter 35 of the BBC Microcomputer User Guide and in Section 9 of this Disk System User Guide.

Especially in the early days with your disk system, you may still wish to access files on cassettes. Although a machine fitted with a disk interface switches on in disk filing system mode, it can be reset using the command *TAPE and set back again by typing *DISK. These commands can even be included in programs. Transferring programs from cassette tape to disc is dealt with in Appendix 2.

5 DRIVES, DIRECTORIES AND FILE SPECIFICATIONS

Drive Numbers

The disk filing system differs from that for cassettes in two respects.
It can control up to four drives (rather than one) - which enables more advanced data handling as well as increased on-line storage capacity. The first drive is numbered 0 and - if double-sided - the reverse would be numbered 2. Similarly any second physical drive is numbered 1 and - if double-sided - the reverse would be numbered 3.
At switch-on (or after pressing < Break>, the current or default drive is 0. It may be changed to 2 by typing *DRIVE 2. All accesses (LOAD and SAVE etc.) will then be on this drive until changed again (or you press <Break> again). The general syntax is therefore *DRIVE n - where n is 0, 1, 2 or 3.
The default drive number may be overridden by including the desired drive number in the filename. Thus LOAD :2.STELLA will result in the file STELLA being loaded on drive 2, regardless of the current drive setting. This is especially valuable within programs, which should behave predictably. The leading colon <:> defines the next character as a drive number while the full stop <.> separates this from the filename.

Directories

A second difference between disk and cassette filing systems is the provision of directories. These are (optional) single character extensions to the filename, which may be used to identify different groups of files – e.g. BASIC programs, machine code programs, graphics screens, text files, etc.
One reason for the existence of directories is that the disk filing system can be expected to accommodate many more files per side of the disk - say two or even four times - up to a maximum of 31 files in the standard catalogue.
[Another reason is that a file can then be stored in two different forms under the same filename but in two different directories.]
At switch-on (or after pressing <Break>, the default or current directory is $. This may be changed by typing *DIR <char> - where <char> can be any character other than :.# or <space> . Thus LOAD F.STELLA will result in the file STELLA being loaded from directory F, regardless of the current directory setting.
Both drive and directory may be included in the filename. Thus LOAD :2.F.STELLA will load the file STELLA from directory F on drive 2.

Both drive and directory may be changed together using the *DIR command.
e.g. *DIR :2.F
while both can be reset to the switch-on values (of 0 and $ respectively) by typing just *DIR.

Libraries

A library can be thought of as a special directory which is usually used for machine code programs, such as utilities (e.g. sort routines, graphics screen, dumps, etc.). It defines a drive and directory which will be searched (as well as the current drive and directory) if a command such as *RUN UTILITY is entered. This may also be entered as *UTILITY, so that the command acts as an extension to the Machine Operating System.
If the current drive and directory is :0.$ and the library is :2.M then *UTILITY will result in both :0.$ and :2.M being searched.

Catalogues

The command *CAT displays the catalogue (i.e. a list of the filenames) on the specified disk drive. For example:

```
*CAT 0
  190                  TITLE (96)
  Drive :0             Option 3 (Exec)
  Directory :0.$       Library :0.$
  !BOOT                CHRIS
  STELLA L             TONY
A.DATA               B.BASIC
```

The header shows the disk title and the (decimal) number of write operations [accesses ?] made to the disk. "Drive" shows the current drive number, "Option" shows the current setting for *OPT 4,n, "Directory" shows the current default drive and directory and "Library" shows the library drive and directory.

File Specifications and Wildcards

The filenames used so far have referred to single files and are formally described as "file specifications" - or <fsp> for short. However, some of the DFS commands can be used on many files at once and this is done by using "ambiguous file specifications" (<afsp>). These include one or more wildcard characters which are defined as # for any single character and * for any sequence of characters (including just one). Wildcards may not be used for the drive number.
Hence-using the *INFO command as an example - whereas, when in the default directory, *INFO "FILE" produces information about the file "FILE" in directory $ on drive 0, *INFO #.File produces information on any files called FILE in any directory on drive 0 and *INFO #.F* produces information on any files beginning with F in any directory on drive 0.

6 COPYING, RENAMING AND DELETING FILES

Operating System Commands

In the BBC Microcomputer, the (Machine) Operating System (MOS) commands are quite independent of any current "language" (e.g. BASIC) commands. This is so that virtually all commands that would be required whatever the "language" are stored in only one location inside the machine - the MOS ROM chip. The same applies to the Disk Filing System (DFS) - which may be thought of as an extension to the Machine Operating System. The DFS commands, too, are stored in only one location - the DFS ROM (or EPROM) chip.
"Languages" include both programming languages, such as BASIC and FORTH, and application programs, like the VIEW word processor and the TOOLKIT programmer's utility. These two are stored in separate locations - more ROM or EPROM chips.
There are several differences between "language" and other commands:

1) Commands issued from within a language are acted upon if recognised - like LOAD and SAVE in BASIC. Commands intended for the MOS or DFS (or indeed other "languages") are identified by the leading * symbol and are offered around in a predetermined fashion by the Command Line Interpreter until recognised. (If not, "Bad Command" is reported). For this reason, such * commands must always be last on any line of a BASIC (or other) program.
2) Whereas BASIC recognises the difference between upper and lower case letters, the Operating System does not. [Thus case is not recognised for MOS and DFS commands]. Nor is it recognised for filenames - even though the order produced by *CAT may suggest otherwise. Hence * commands and filenames may be given in either upper or lower case: e.g. *LOAD file and *load FILE are equivalent.
3) While filenames require to be enclosed in quotation marks (inverted commas) when used with BASIC commands, they are not in general needed when used with * commands. The only instance when they are required with * commands is when the filename contains spaces.
4) Since the Machine Operating System always uses hexadecimal notation, the & identifier is not required when giving the addresses for *LOAD and *SAVE commands. Similarly, addresses and file lengths given in reply to *INFO are understood to be in hexadecimal.

Copying and Renaming Files

Five DFS commands - *COPY, *BACKUP, *COMPACT, *F40 and *F80 - make use of the main user RAM as a buffer, overwriting anything that is already present. Therefore you should save any wanted program or data to a separate file before issuing these commands.
Although floppy disks are very convenient carriers for information, they can always be lost and are also open to corruption - especially from magnetic fields. Also it is all too easy to issue a wrong command (perhaps at the end of the day) and so to lose the result of days - or even weeks - of work. Therefore it is vital to back up all important files - particularly in a business situation.
The best business practice is to use three disks - known as Grandfather, Father and Son - in rotation, backing up the Son disk - the day's work disk - on to the Grandfather disk at the end of the day.

Then - if, say, the Son disk is lost and the Grandfather disk corrupted, at least you have the Father disk and have lost only one day's work!
For home computing it is probably enough to keep a single backup, but full use should also be made of write protection tabs and locking files (as described below).
In this connection it is always worthwhile to write protect the "source" or "master" disk before using it for *COPY or *BACKUP. This is because it is quite easy to reverse the drive numbers in these commands (or to put the disks in the wrong drives) and copy the old version over the new version !

To copy a file from one disk to another, put the "source" disk in drive 0 and the destination disk in drive 1 and type *COPY 0 1 <filename>.
To copy groups of files you may define the group by the directory: e.g. *COPY 0 1 B.*
However, this would put the files into the default directory ($) of drive 1 unless it had been set to B with *DIR :1.B.
You may also copy a group of files by using an ambiguous file specification - i.e. a filename containing wildcards (see above).
A special case here is to type *COPY #.* - which will copy all files but will omit the spaces left by deleted files. Also it will not overwrite files on the destination disk and if a file of the same name is present it will stop with the message "File exists".
Another special case occurs when you have only a single disk drive.
While you could LOAD and then SAVE (or *LOAD and then *SAVE) each file, it is actually catered for by issuing *COPY 0 0 <afsp>. You are then prompted to insert the source and destination disks in turn, pressing any key in between. Since many disk changes may be involved, a methodical procedure is essential - preferably backed up by write protecting the source disk.
Any file may be renamed (and/or have its directory changed) by typing:

e.g. *RENAME $.OLDNAME B.NEWNAME

Copying is also possible between disks having different numbers of tracks and recording densities. This is particularly easy with the Solidisk DFS because it is "intelligent" - having both track- and density-sensing. These are established by the original formatting (or re-formatting) of the disk and so - after putting a disk having the desired characteristics into the destination drive - the simple command *COPY m n #.* may be issued.
This will even work for copying with a single drive. The only caution necessary is about disk capacities - since an 80-track double density disc has 3.2 times the capacity of a 40-track single density disk.
Another way of backing up from one disk to another may be effected by:

e.g. *BACKUP 0 1

However, since this causes the destination disk to be wholly overwritten without further warning, the DFS requires that *ENABLE be issued immediately beforehand. Moreover, as the disk contents are copied "blindly" (sector by sector rather than file by file), this copies even deleted files to the destination disk. Nor is it possible to have different track and density

settings on the two disks. It does, however, copy the auto-start option set by *OPT 4,n.

Locking files

The command *ACCESS is used to prevent files being overwritten or deleted by accident. It works by "locking" or "unlocking" the file. When locked, the file can be copied but not written to, or renamed - as these involve partial overwriting.
Locking is proof against SAVE, *SAVE, *RENAME, *DELETE and *WIPE. It is not proof against *BACKUP or re-formatting - neither of which read the contents of the disk before acting.
Locking thus provides only partial security for valuable files. It has to suffice if other files on the disk must be left unlocked (e.g. for writing to). However, for maximum security (e.g. against copying in the wrong direction), a write project tab should be used.

Deleting files

The command *DELETE causes a single specified filename to be removed from the catalogue of a disk. The space that was occupied by the file will then become available to future write operations (e.g. SAVE, *SAVE etc.). Even if it has not been overwritten, a deleted file is almost impossible to recover.

Disk Management

When seeking to SAVE or *SAVE a file, the DFS looks for a space (comprising a whole number of sectors) large enough to accommodate it. There may well be ample space in total - as the result of deletions - but it could typically be scattered over the disk. The command *COMPACT moves all the files on the specified disk drive to the beginning of the disk so as to leave all the spare space in a single block at the end.

7 OTHER USEFUL COMMANDS

Help Commands

*HELP may be used to give the syntax of commands or utilities of any sideways ROMs fitted. These include the Disk Filing System and any "language" (such as BASIC).
*HELP with a keyword can be used to limit the response to particular ROMs or even particular commands. Two keywords are relevant to the Disk Filing System - DFS and UTILS.

Auto-Starting

The command *BUILD is used to create the !BOOT file that can be set up to run automatically on pressing Shift-Break.
As well as auto-starting (e.g. by chaining a program), a !BOOT file can also be used to set the default disk drive and to reset the value

of PAGE (which is not possible within a normal program). Other possible contents of a !BOOT file include setting *TV, *<Language>, function key assignments and the display mode.
After terminating the !BOOT file by pressing <Escape> on a new line, the action on pressing Shift-Break needs to be set with *OPT 4,n. For example, *OPT 4,3 results in the !BOOT file being EXECuted - i.e. it stimulates input directly from the keyboard.

Disk Title and Version Number

The command *TITLE may be used to give a disk a title of up to 12 characters. The header produced by *CAT also shows the (decimal) number of write operations made to the disk. With care this can be used as a version number in disk copying or back-up operations.

Types of File

The type of file that many users know best is a program in BASIC.
Such files are normally stored in a compact "tokenised" form which is used by the commands LOAD, SAVE, etc. Another type is called an ASCII text file (after American Standard Code for Information Interchange). It is widely used for transferring files between programs and between computers and is used by the commands *BUILD, *EXEC, *LIST, *TYPE and *SPOOL. *EXEC and *SPOOL may also be used to convert between BASIC and ASCII files.

Hexadecimal Listing

The command *DUMP produces a listing of the specified file in hexadecimal and ASCII. All the contents of the file are shown in hexadecimal on the left side of the screen while only the printable ASCII characters are shown on the right-hand side. Non-printable ASCII characters - such as control codes (below 32 decimal) and graphics codes (above 127 decimal) show only as full stops.
This command is very useful for examining files of any type - e.g. BASIC programs, machine code programs, text, etc. - as part of trouble shooting both the software and hardware of a computer system.

8 THE DISK FILING SYSTEM COMMANDS

The filing system commands comprise two groups:

1) The Machine Operating System (MOS) Commands
These apply to all BBC Microcomputers and are held in a ROM chip. As well as the standard cassette filing system, many of them apply also to other filing systems - such as the disk filing system.
The MOS filing system commands include:

*CAT	*EXEC	*HELP
*LOAD	*OPT 1	*RUN
*SAVE	*SPOOL	

2) The Disk Filing System (DFS) Commands.
These apply to BBC Microcomputers fitted with a disk interface

system and one or more disk drives. They are held in a ROM (or EPROM) chip.

The standard DFS commands include:

```
*ACCESS    *BACKUP    *BUILD
*COMPACT   *COPY      *DELETE
*DIR       *DRIVE     *DUMP
*ENABLE    *INFO      *LIB
*LIST      *OPT 4     *RENAME
*TITLE     *TYPE      *WIPE
```

In addition, the Solidisk DFS includes the following commands:

```
*DDFS      *F40       *F80       *VERIFY
```

For ease of reference all these commands are now dealt with together in alphabetical order.

There is also a group of filing system commands which are available from BASIC and other languages. Although they include many of the same words as the MOS and DFS commands, they are not prefaced by the * symbol and have significantly different effects. They include the well-known LOAD, SAVE and CHAIN commands.

*ACCESS <afsp> L

This command is used to prevent files being overwritten or deleted by accident. It works by "locking" or "unlocking" the file. When locked, the file can be copied but not written to, or renamed - as these involve partial overwriting.
E.g. *ACCESS STELLA L - locks the file STELLA
*ACCESS STELLA - unlocks the file STELLA

*ACCESS will not work on a write-protected disk and will result in the message "Disk write protected". This is because locking (and unlocking) itself writes to the disk catalogue.

After *CAT or *INFO, a locked file will display L after its name.

Locking is proof against SAVE, *SAVE, *RENAME, *DELETE and *WIPE. Applying any of these commands to the file in question will result in the message "File locked". It is not proof against *BACKUP or re-formatting - neither of which read the contents of the disk before acting.

Locking thus provides only partial security for valuable files. It has to suffice if other files on the disk must be left unlocked (e.g. for writing to). However, for maximum security (e.g. against copying in the wrong direction) a write protect tab should be used.

EXAMPLE:

```
*ACCESS *.*          UNLOCK ALL FILES
*ACCESS *.* L        LOCK ALL FILES BEFORE COPYING
```

*BACKUP <source drive> <destination drive>

This command is used to produce a complete copy of a disk. It operates faster than *COPY #.* since it copies "blindly" sector by

sector without reading or checking the contents of any files. Again unlike *COPY #.*, it also copies the disk title and any auto-start options (for the !BOOT file) set by *OPT 4,n.

Since such "blind" copying will overwrite the contents of the destination disk it is necessary to type *ENABLE immediately before *BACKUP - otherwise the reminder "Enable" is displayed.

```
E.g.       *ENABLE
           *BACKUP 0 1
Backup from drive 0 to drive 1
```

Like copying, backing-up is also possible even with a single disk drive - as *BACKUP 0 0 - and the DFS will prompt the exchanging of source and destination disks. Since this operation employs user RAM as a buffer it is necessary to save any wanted program or data before issuing *BACKUP.

```
EXAMPLE:
           *ENABLE
           *BACKUP 0 0
```

*BUILD <fsp>

This command is used to create text files directly from the keyboard. It is most commonly used to create the !BOOT file that can be set up to run on pressing Shift-Break.

```
E.g.       *BUILD !BOOT
           0001 PAGE=&2400
           0002 *DRIVE 1
           0003 CHAIN "FIRST"
           0004 <Escape>
```

This shows that a !BOOT file can be used to reset the value of PAGE (which is not possible within a normal program) and the default drive as well as chaining a program.

Other possible contents of a !BOOT file include *TV, *<Language>, function key assignments and setting the display mode.
After terminating the !BOOT file by pressing <Escape> on a new line, the action on pressing Shift-Break needs to be set with *OPT 4,n (which see). For example, *OPT 4,3 results in the !BOOT file being *EXECuted – i.e. it stimulates input directly from the keyboard. The !BOOT file must be in (and the computer must be set to) drive 0 and directory $, otherwise the message "File not found" will be displayed.

Although *COPY #.* will copy the !BOOT file it will still need to be enabled by typing on the new disk *OPT 4,n - as appropriate.

*BUILD can also be used to create long text files. These can comprise e.g. the instructions - so reducing the size of a game or application program which is subsequently chained. However, they can be tedious to create as - unlike in a word processor (or even BBC BASIC) - there is no provision for editing the file. If you make a mistake, therefore, you have to type it all in again! Hence long text

files are much better created in a word processor and *SPOOLed in.

*CAT (<drive>)

This command displays the catalogue (i.e. a list of the filenames) on the specified disk drive. If no drive is specified, it is taken to be the current drive.
The filenames are shown sorted in two respects. Firstly they are grouped by directory (which see), with the current directory shown first, followed by any other directories in alphabetical order.
Secondly, within each directory the filenames are themselves sorted into alphabetical order. (Actually they are sorted in ASCII order - with lower case names coming after upper case - but there is no functional difference between files whose names differ only in case).

E.g. *CAT 0

```
            190      TITLE        (96)
            Drive : 0             Option 3 (Exec)
            Directory :0.$        Library :0.$

            !BOOT                 CHRIS
            STELLA L              TONY
          A.DATA                B.BASIC
```

The header shows the disk capacity and title and the (decimal) number of write operations made to the disk. With care this can be used as a version number in disk copying or back-up operations. "Drive" shows the current drive number, "Option" shows the current setting for *OPT 4,n, "Directory" shows the current default drive and directory and "Library" shows the library drive and directory. This last is the drive and directory (in addition to the current default drive and directory) that will be searched when *<filename> is issued.

Under the Acorn and similar DFS's, <drive> can have the value 0, 1, 2 or 3. Under the STLDISC DFS, used with the Solidisk Sideways System, it can additionally have the value 4 - which is used for the RAM disk when mechanical drives 0 to 3 are also present.

With the Solidisk DFS, the header also shows the size of that drive in (hexadecimal) number of sectors - each of 256 bytes. E.g.:
&190 sectors equals 100 K, implying a 40-track single density disk
&280 sectors equals 160 K, implying a 40-track double density disk
&320 sectors equals 200 K, implying an 80-track single density disk
&500 sectors equals 320 K, implying an 80-track double density disk

For example, a double-sided diskette can have &500 sectors on one side and &500 sectors on the other side, giving up to 640k bytes of storage.

*COMPACT <drive>

This command moves all the files on the specified disk drive to the beginning of the disk so as to leave all the spare space in a single block at the end. When seeking to SAVE or *SAVE a file, the Acorn-type DFS looks for a space (comprising a whole number of sectors) large enough to accommodate it. There may well be ample space in total - as the result of deletions - but it could typically be scattered over the disk. (Deletion in fact only means removal from the disk catalogue). The purpose of *COMPACT is therefore to try to avoid the message "Disc full". E.g.:

```
*COMPACT 0
$.DATABAV L FF1900 FF8023 000EC2 043
$.DHEAT1  L FF1900 FF8023 001326 02C
$.DCOLL11 L FF1900 FF8023 0009B2 022
$.MICS      FF1900 FF8023 000A7C 014
$.DLLOGO8 L FF1900 FF8023 000725 00C
$.DCOMBO2 L FF1900 FF8023 000230 009
$.BIGWOR2 L FF1900 FF8023 0001C4 005
$.DISPLAA L FF1900 FF8023 0001B6 002

Disc compacted 0E7 free sectors.
```

The number of sectors free is given in hexadecimal. This may be converted into decimal Kbytes by typing PRINT TMnnn/4.

This command employs user RAM as temporary workspace, so you should save any programs or data before issuing it. A frequent situation is that you need to save a long program that you just typed in and discover that there is not enough room on any of your disk to accommodate your program. DO NOT try to compact any disk as it will inevitably destroy your program. Fortunately, with the Solidisk DFS you have a resident formatter; you may format a blank disk and save your program before compacting any other disk. If you do not have any spare disks at all, you could type *TAPE, followed by PAGE=&19OO and save it on tape.
Tidy your diskettes once a week by compacting them. You may even turn on the printer, type VDU2 and systematically *COMPACT all the ordinary diskettes. This will produce a summary of all your software on diskettes.

*COPY <source drive> <destination drive> <afsp>

This command copies the specified file (or files) from one disk drive to another. E.g.:

```
*COPY 0 1 STELLA
$.STELLA L FFI9OO FF8023 00149C 22A
>
```

Since the source and destination drives are specified as part of the command, no drive should be specified as part of the filename. However, the file directory should be specified as part of the filename or it will be taken to be taken as the current directory.
Groups of files can be copied by the use of "wildcard" characters in the file specification - thus making it "ambiguous". E.g.:

```
*COPY 0 1 PROG*
          would copy all files in the current directory beginning with PROG
*COPY 0 1 S.*
          would copy all files on directory S.
*COPY 0 1 #.* (or *COPY 0 1 *.*)
          would copy all files on any directory.
```

Although *COPY would copy the !BOOT file itself, the copy on the destination disk would still need enabling with *OPT 4,n (which see).
This command loads the file(s) into user RAM and then saves it/them on to the destination disk. You should therefore save any wanted program or data file before issuing it. Like SAVE and *SAVE, *COPY will attempt to use any available spaces (the results of deletion) on the destination disk. If a file of the same name exists in the current directory of the destination drive, the message "File exists" will be given and overwriting will not take place.
*COPY can still be used with a single disk drive - with messages to prompt the changing between the source and destination disks.
*COPY will not work on "open" files. Such files may, however, be closed by pressing <Break> before issuing the *COPY command.

*DELETE <fsp>

This command causes a single specified filename to be removed from the catalogue of a disk. The space that was occupied by the file will then become available to future write operations (e.g. SAVE, *SAVE etc.). Even if it has not been overwritten, a deleted file is almost impossible to recover.

More than one file may be deleted at once, using the *WIPE command (which see).

After several deletions the space freed may be consolidated at the end of the disk with *COMPACT (see later).

Accidental deletion is best prevented with a write protection tab - which applies to the whole disk (both sides). If access is required to some files on a disk, the others may be locked by using the *ACCESS command (which see).

*DIR (:<drive>.) <directory>

This command may be used to set the current directory (and drive).

A directory allows files on one disk drive surface to be distinguished from others, even though they may have the same seven-character filename. Typical uses of directories could be B. for BASIC programs, D. for data files, W. for word processor files etc. Thus, in a given work session, any files saved with only the seven-character filename could be put automatically into the appropriate directory.

Directories can have as identifiers any single printing character, such A to Z and even 0 to 9 (though these latter are easily confused with drive numbers). The characters : . # * are not permitted as directory identifiers as they have other meanings in full filenames.

The contents of all directories are shown in the catalogue display, with those of the current directory at the top.

The default directory is (drive 0) directory $ and this is restored whenever <Break> is pressed.

*DRIVE <drive>

This command sets the current or default drive. This is the drive that is used if none is specified in the filename. However, another drive can be specified for any other command (than *DRIVE) by including the drive number in the filename. Thus:

```
           *DRIVE 2 sets the current drive to 2 and
           *CAT will show the catalogue of drive 2.
```

However,

```
           *CAT 0 will show the catalogue of drive 0 and
           *SAVE :0.FILE will save to drive 0
```

- both without resetting the current drive.

Pressing the <Break> key resets the current drive to 0.

*DUMP <fsp>

This command produces a listing of the specified file in hexadecimal and ASCII. All the contents of the file are shown in hexadecimal on the left side of the screen while only the printable ASCII characters are shown on the right-hand side. Non-printable ASCII characters - such as control codes (below 32 decimal) and graphics codes (above 127 decimal) show only as full stops.

This expanded presentation gives very long listings which are best viewed in page mode, set by Ctrl-N, stepped through with <shift> and reset by Ctrl.-O, or by controlling scrolling by pressing Shift-Ctrl.

This command is very useful for examining files of any type - e.g. BASIC programs, machine code programs, text, etc. - as part of trouble-shooting both the software and hardware of a computer system.

Alternative commands - which will only work correctly on straight printable ASCII files - are *LIST and *TYPE (which see).

*ENABLE

Some of the filing system commands - such as *BACKUP and the disk formatting commands - can lead to accidental deletion of valuable files. *ENABLE is therefore provided as a sort of safety catch - to cause the user to pause and think about the consequences of the intended action (the next command). Part of the reason is that DFS's make limited provision for warning that files already exist on the disk.

*ENABLE has to be typed just before such potentially destructive commands - otherwise the reminder "Enable" will be displayed.

*EXEC <fsp>

This command causes a file to be executed – i.e. to be read in by the computer as though it were being entered at the keyboard. Such a file is useful whenever a given sequence of keyboard entries - whether commands or data - is required repeatedly.

One file that is usually EXECed is the auto-start file !BOOT, which is often created (using *BUILD - which see) to contain initial set-up commands such as *TV, *DRIVE, PAGE=nnnn, CHAIN <filename> etc. To auto-start the !BOOT file (by pressing Shift-Break), it must be enabled with *OPT 4,3 (which see).

A frequent situation that *EXEC (and *SPOOL) is used: when extracting and merging BASIC programs. For example, you may need to copy lines 1000 to 2000 or PROG-ONE to be lines 500 to 700 in PROG-TWO.
You will have to SPOOL lines 1000,2000 of PROG-ONE, then EXEC, RENUMBER, SPOOL it again and finally EXEC it to PROG-TWO:

```
LOAD "PROG-ONE" (prepare for extract)
*SPOOL EXTRACT (open a file on disk for output)
LIST 1000,2000 (get the BBC to type out the wanted portion)
*SPOOL (tell the BBC to finish, so close the spooled file)
NEW
*EXEC EXTRACT (to enable renumbering)
RENUMBER 500,1 (get the line numbers in range for the receiving program)
*SPOOL EXTRACT (reopen the output file)
LIST (get the BBC to type out again, with line numbers in range)
*SPOOL (close the output file)

LOAD "PROG-TWO"
*EXEC EXTRACT
SAVE "PROG-TWO"
*DELETE EXTRACT
```

You can see by this example how tedious the work would be without disk drives. Another use of *EXEC is to enter a text file -such as a BASIC or machine code program which has been created in a word processor. The latter usually provides full screen editing and allows search-and-replace, merge files and block moves which aid the programmer.

*F40 <drive>, *F80 <drive>

One or other of these commands is used to format your disks. The pattern that is laid down on the magnetic surface of the disk guides all subsequent reading and writing operations. [?] As this completely erases any files that may already be on the disk, the DFS requires that *ENABLE be issued immediately beforehand.

The number 40 or 80 refers to the number of concentric circular tracks per surface to be laid down on the disk - and is usually dictated by your disk drive(s). With STL DFS, it is possible to specify *F40 even with fixed 80-track drives, and "track skipping" will take place automatically.

Another choice that is offered by STL DFS is that between single and double density recording. These terms refer (somewhat imprecisely) to the number of sectors per track - either 10 or 16 in this case resulting from different recording techniques.

The Solidisk DFS is "intelligent" in that it has both "track sensing" and "density sensing" - so that a disk formatted at 40-track or 80-track, single density or double density will be read and written to according to those standards.

You can format both 40 and 80 tracks on an 80-track disk drive (which is capable of 80 steps, from the outer ring to the inner ring of the diskette), but only 40 tracks on a 40-track drive.

In addition to the capabilities of his/her own disk system, the user must also consider those of any target or recipient system - as not all are capable of reading disks formatted at 80 tracks and even fewer are capable of reading double density.

Clearly - like choosing double-sided over single-sided disk drives - choosing 80 over 40 tracks and double over single density can greatly increase the storage capacity of a disk. However, these same factors all increase the cost of both disk systems and disks. On balance, though, the user gains substantially from choosing the higher specification.

The format commands *F40 and *F80 require you to specify the drive number - which, on the Acorn/BBC Micro system, can be 0, 1, 2 or 3. According to a rather strange convention, Drive 2 is the second side of Drive 0, Drive 1 is the first side of any second physical drive unit while Drive 3 is the second side of that. It is therefore necessary to specify Drive 2 (or set Drive 2 to be the current drive) in order to format the disk surface for the second side of a double-sided drive.

Some examples:

1) With TEAC FD55A, single sided, 40-track drive, using the normal 1.4 STL DFS: place a blank diskette (such as Verbatim DATALIFE MD525) in drive 0, then enter:

```
*ENABLE <RETURN>
*F40 0 <RETURN>
```

2) With TEAC FD55F, double sided, 80-track drive, using the 1.8 STL DFS: place a blank diskette (such as Verbatim DATALIFE MD557) in drive 0, then enter:

```
           *ENABLE <RETURN>
           *F80 0 <RETURN>
Side 0 is now formatted.
           *ENABLE <RETURN>
           *F80 2 <RETURN>
Side 2 is also formatted.
```

A good habit is to format all blank diskettes, both sides whenever possible, as soon as you can. STL DFS will automatically verify them. Faulty diskettes cannot be formatted; they should be returned to the seller immediately for replacement.

*HELP (<keyword>)

This is a Machine Operating System command – i.e. it is available on all BBC Microcomputers. It is used to give the syntax of commands or utilities of any sideways ROMs fitted (although some ROMs do not do so). This includes the Disk Filing System and any "language" (such as BASIC).

*HELP with a keyword can be used to limit the response to particular ROMs or even particular commands. Two keywords are relevant to the Disk Filing System - DFS and UTILS. E.g.:

```
           *HELP DFS
DDFS 1.40
           *HELP DFS
           ACCESS <afsp> (L)
           BACKUP <source drive> <destination drive>
           COMPACT (<drive>)
           COPY <source drive> <destination drive> <afsp>
           DELETE <fsp>
           DIR <directory>
           DRIVE <drive>
           ENABLE
           F40 <drive>
           F80 <drive>
           INFO <afsp>
           LIB <directory>
           RENAME <old fsp> <new fsp>
           TITLE <disk title>
           VERIFY (<drive>)
           WIPE <afsp>

           *HELP UTILS
           DDFS 1.00
           BUILD <fsp>
           DUMP <fsp>
           LIST <fsp>
           TYPE <fsp>
```

The responses given are detailed in this Section.

*EXEC, *LOAD, *RUN, *SAVE and *SPOOL are Machine Operating System (as opposed to Disk Filing System) commands - and

so are omitted from the responses to *HELP. They are, however, also detailed in this section.

*INFO <afsp>

This command displays information about the files on a disk surface - much more than is given by *CAT. The files are listed in the actual order in which they are stored in the disk catalogue - rather than being sorted into alphabetical order.

D.FILE L SSSSSS EEEEEE LLLLLL PPP

where D is the directory (of that disk drive/surface) in which the file is stored, FILE is the filename, L is present if the file is locked, SSSSSS is the load (or start) address in the computer for this file (which can largely identify the type), EEEEEE is the execution address used in response to a *RUN command (which see), LLLLLL is the length of this file in bytes and PPP is the start sector on the disk for this file.
SSSSSS, EEEEEE and LLLLLL are all in hexadecimal.

The wildcard characters # and * may be used to obtain information about a group of files:

```
E.g.       *INFO *.*
           $.ICOLL11 L 003000 003000 005000 0AB
           $.COMB02  L 003000 003000 005000 05B
           $.RANGE1  L FF1900 FF8023 000838 052
           $.DATABAV L FF1900 FF8023 000EC2 043
           $.DHEAT1  L FF1900 FF8023 001326 02C
           $.DCOLL11 L FF1900 FF8023 0009B2 022
           $.MICS      FF1900 FF8023 000A7C 014
           $.DLLOGO8 L FF1900 FF8023 000725 00C
           $.DCOMBO2 L FF1900 FF8023 000230 009
           $.BIGWOR2 L FF1900 FF8023 0001C4 005
           $.DISPLAY L FF1900 FF8023 0001B6 002
```

NOTES:
Acorn DFS limits the disk and file size to 1/4 Megabyte (256k), STL DDFS extends effectively the disk size to 1 Megabyte while keeping the file size still at 1/4 MB. It is, however, planned that new revision will offer file and disk size up to 16 MB together with a new OSWORD call 7C to allow databases accessing ever faster and bigger disk drives. The proposed format for OSWORD 7C is as follows

```
BYTE POSITION IN FCB:
 0   1     2   3 4  5  6   7     8
DRV:L-ADD:H-ADD :  :  : :COMD::H-STARTsect:
 9            A
L-STARTsect:LENGTH
```

The noticeable difference between this proposed OSWORD 7C and the standard OSWORD 7F is the use of LOGICAL STARTING SECTOR in place of the track number. Files up to 16MB long are allowed and without dependance on any DFS or FDC chip.

*LlB (:<drive>.) <directory>

This command sets the library to a specified (drive and) directory.
This defines a second disk area which will be searched (in addition to the current drive and directory) in response to the command *RUN NAME (or *NAME for short).
E.g after typing:

```
*LIB :2.A
```

typing:

```
*RUN <filename> (or *<filename>)
```

will behave as if you had typed:

```
*RUN :2.A.<filename>
```

*LIB enables machine code programs to be stored on disk and treated as extensions to the Machine Operating System commands (additional to those stored in ROM's). Typical examples of such programs could include utilities like sort routines and graphics screen dumps.

*LIST <fsp>

This command causes a text file to be displayed on the screen, with line numbers. The text comes from the file but the line numbers are supplied by the DFS program. Files other than straight, printable ASCII character text files will display nonsense or even confuse the computer - requiring <Break> to be pressed.

Large files may require listing in page mode - set by Ctrl-N, stepped through with <shift> and reset by Ctrl-O. Alternatively, scrolling can be controlled by pressing Shift-Ctrl.

*LIST may be used on files produced by *BUILD, *SCROLL (which see) and most word processors.

The command *TYPE will list text files without line numbers while *DUMP will list any file without the risk of confusing the computer.

*LOAD <afsp> (<address>)

This command loads a file into the computer memory starting either at the load address saved with the file or at a load (or start) address (in hexadecimal) specified with the command.

The command is provided by the Machine Operating System (as opposed to the Disk Filing System) and is intended for machine code or data) (e.g. graphics screen) files. As such it is distinct from the LOAD command provided in BASIC and other languages [such as the VIEW word processor]. With *LOAD, the quotation marks (inverted commas) embracing the filename are optional. If the drive and directory are not specified, the current default values will be taken.

*RUN may be used to *LOAD and run machine code files - much as CHAIN is used to LOAD and RUN BASIC programs.

EXAMPLE:
To load a screen picture in mode 2 from disk:

```
MODE 2 <RETURN>
*LOAD PICTURE <RETURN>
```

To load a sideways RAM program such as PRINTER across the Econet:

```
*LOAD PRINTER 8000 <RETURN>
```

To modify the RELOAD or EXEC address of an existing program such as STLOEOO:

```
*INFO STLOEOO <RETURN> (get the file info)
          STLOEOO L 008000 008000 002000 015
*LOAD STLOEOO 2000 <RETURN> (find a place for it)
*SAVE STLOEOO 2000 +2000 D9CD 8000 <RETURN>
```

The last command will give a new EXEC address to the STLOEOO program. In this example it is &D9CD or reset entry point for MOS 1.2. The overall effect is that you can simply type *STLOEOO to change the present filing system for the new one.

*OPT 1,n

This command is used to enable and disable the display of full file information (as *INFO which see) whenever the file is accessed for reading or writing. E.g:

```
*OPT 1,1 enables the display of full file information
*OPT 1,0 disables the display of full file information.
```

A space maybe used in place of the comma between the figures.

Such information can be helpful during program development but may upset screen displays at the same time. The default setting - at switch-on or after <Break> is full file information disabled.

*OPT 4,n

This command sets the option for the auto-start (!BOOT) file (see *BUILD). This file must be on drive 0, directory $ and auto-start is initiated by pressing Shift-Break. If !BOOT is not present on drive 0, directory $. the message "File not found" is displayed.

*OPT 4,n is used to select the action applied to the !BOOT file thus:

```
*OPT 4,0 does not try to load any file.
*OPT 4,1 will *LOAD the !BOOT file.
*OPT 4,2 will *RUN the !BOOT file.
*OPT 4.3 will *EXEC the !BOOT file.
```

A space may be used in place of the comma between the figures.

The !BOOT file usually contains the line:

```
CHAIN "file"
```

- where "file" is a BASIC program, such as a menu.

The option is usually set with *OPT 4,3 for the !BOOT file - including the line CHAIN "file" - to be EXECuted, as though it was entered directly from the keyboard.

The current option is shown at the top of the catalogue display.

*RENAME <old afsp> <new afsp>

This command changes the name of a file (or files) and can also be used to move it/them to another directory.
E.g.

```
*RENAME :1.A.DFS B.DFS1.4
```

A drive number may be included in the first filename but not in the second.

If a file of the new name is already present on the drive and directory, the message "File exists" will be displayed and the file will not be renamed.

The disk must not be write-protected nor the file locked or appropriate messages will be displayed (i.e. 'Read only' or 'Locked').

NOTES:
Directories and libraries are simple ways of differentiating or regrouping files ON THE SAME DISK. You cannot by way of renaming a file transfer a file in directory A, drive 1, to directory B, drive 0. There are situations where you may want to copy a new version of a program such as (STL) DFS from disk A to disk B which also has an old version of (STL) DFS. One way of doing it is by renaming and copying:

EXAMPLE: Copying DFS (version 1.8) from drive 0 to 1 which already has another DFS (version 1.4):

```
*RENAME :0.DFS DFS1.8 <RETURN>
*COPY 0 1 DFS1.8 <RETURN>
```

*RUN <afsp> (parameters to utility)

This command is used to load and run machine code programs.
This means that it *LOADS the file and then jumps to the execution (or start) address saved with the file - as displayed by *INFO (which see).

The file should be either in the currently selected drive and directory (or the appropriate drive and directory should be included in the filename) or in the library (see *LIB). A permitted shorthand form is *<filename> in place of *RUN <filename>. Together these enable machine code programs, such as utilities, to be treated as extensions to the list of Machine Operating System commands.

*SAVE <filename> <start address> <finish address> (<execute address> <reload address>)

This command is quite distinct from SAVE - which is available in BASIC. *SAVE is used to take a copy of any part of the computer memory - including machine code or data (such as a graphics screen) and save it to the current filing system. The part saved is defined by the start address and finish address (or the start address and the length). E.g.:

```
*SAVE "FILE" SSSS FFFF or
*SAVE "FILE" SSSS + LLLL
```

where SSSS is the start address of the part of memory to be saved, FFFF is the finish address and +LLLL is the file length and the + sign distinguishes it from a finish address.

Optionally the execution address – EEEE - and the reload address – RRRR - may also be given. If not, they are both assumed to be the same as the start address.

The above addresses are in hexadecimal and only four digits need be given for the standard, single processor computer system. The same addresses are displayed by *INFO (which see) to six digits - to cater for any second processor which may be connected.

If the disk has a write-protect tab fitted the message "Disk write protected" will be displayed.

If the disk already has the maximum number of files (31 for the standard Acorn catalogue), then *SAVE will result in the message "Catalogue full" being displayed.

If the second filename already exists and is locked, the message "File locked" will be displayed. If it is not locked then the old file will be deleted and an attempt made to *SAVE the new file. If the new file is longer than the old or no space long enough is found, the message "Disk full" will be displayed.

EXAMPLE:
To save a picture in mode 2:
*SAVE PICTURE 3000 +5000 <RETURN>

3000: starting address in hex, begin of the screen, +5000: size of the file. The first number corresponds with the beginning of the screen (location 3000 in hex is 12288 in decimal), 3000 + 5000 is the first byte above the screen (8000 in hex is 32768 in decimal).

The following command is equivalent.
*SAVE PICTURE 3000 8000 <RETURN>

NOTES:
SAVE (and similarly LOAD) are dependent on the language and normally 'translated' appropriately into a MOS *SAVE or *LOAD command. The user is not requested to supply extra infromation such as SSSS or LLLL or EEEE as this information is intimately 'known' to the language.

For example, BASIC would translate a simple command such as.

SAVE "FRED" <RETURN>
into a:
*SAVE FRED FF1900 FF1ACD FF8023

The first number (FF1900) is the PAGE value, the second number is the TOP value. Other languages such as wordprocessors (e.g. View), spreadsheets (e.g. viewsheet) or databases all have similar commands.

Alternatively, LOAD will be interpreted as well by the language.

LOAD "FRED" <RETURN>
is equivalent to:
*LOAD FRED 1900

where 1900 is the PAGE value.

*SPOOL <fsp>

This command opens up a file to receive ASCII text. This may come from another file already in memory which may be LISTed to the screen. Everything appearing on the screen is included in the ASCII file until it is closed by typing *SPOOL alone.
This is particularly useful for obtaining a text file of a BASIC program. E.g.:

```
LOAD "PROG"
*SPOOL PROGTXT
LIST
*SPOOL
```

Whereas BASIC programs are normally stored in a more compact "tokenised" form, ASCII text versions can be:

1) loaded in to a word processor, e.g. for (full screen) editing, search-and-replace, block moves and merging with other such program files;

2) sent to another computer, either nearby - via a serial cable - or remote - via a modem and the telephone network.

3) sent to a printer. This form allows a wordprocessor to produce delayed printing time.

Some DFS commands, such as *CAT and *INFO disable spooling - to avoid clashes in memory usage. Hence - although unlikely - they should not be included in the program file.

Spooled files maybe viewed using the *LIST and *TYPE commands (which see). This does not re-enter them into the computer memory, which would be done with the *EXEC command.

NOTES:
Although up to five files can be opened at the same time for outputting, only one is used for spooling. There is no difference between the files as far as the DFS is concerned; the difference comes from the way the commands are presented to the DFS. *SPOOL echoes the character going to the screen (which eventually can be

'sunk' or not printed, or OSCHW or &FFEE) to the spooled file.

*TITLE <fsp>

This command changes the title of a disk to a name of up to 12 characters. E.g.:

```
*TITLE "NEW DISK"
```

Quotation marks (inverted commas) are required only if the title includes spaces.

*TYPE <fsp>
This command displays the contents of a text file on the screen without line numbers. Long files may be displayed in page mode, set by Ctrl-N, stepped by <Shift> and reset by Ctrl-O or controlled by Shift-Ctrl.

*LIST affects a similar result, but with line number.

*DUMP displays the contents of any file on the screen in hexadecimal-with ASCII equivalents where these are printable.

*VERIFY (<drive>)

This command causes the whole disk in the current (or specified) drive to be scanned. The test compares the checksums stored with each sector with the contents of that sector.

Verification is often done as a measure of disc quality immediately after formatting but is usually included in the formatting process - as with *F40 and *F80 in this DFS.

Used separately, *VERIFY can detect any subsequent corruption of a disk - e.g. caused by a magnetic field.

*WIPE <afsp>

This command specifies a group of files for deletion from a disk - but asks for confirmation one by one. E.g.:

```
*WIPE *.*
A.FILE (Y/N)-
```

Every file in every directory will be offered in turn for deletion.
The user should press Y for deletion (or any other key for retention).

If you press 'Y' or 'y':

```
A.FILE (Y/N)-Y
A.PROG-ONE (Y /N)-
```

If you press the <space bar>,

```
A.FILE (Y/N)-Y
A.PROG-ONE (Y/N)-N
B.PROG-TWO (Y/N)-
```

Files which are locked (using *ACCESS - which see) are not offerred for deletion.

NOTES:

Many other DFS's have the commands *DESTROY which is equivalent to:
*WIPE *.*

SHORT FORMS:

All DFS commands and utils can have short forms:
*. = *CAT
*LO.= *LOAD
*SA.= *SAVE etc.
Short form command may lead to unexpected results as the commands are offered to all sideways ROM/RAMs, from the highest ROM (ROM 15) to the lowest ROM (ROM 0). DFS ROM is usually one of the lower ROMs; it may not have a chance to offer its services if another ROM has the same short form command.
Fortunately enough, duplication of DFS short form commands are generally avoided by all Sideways ROM manufacturers.

APPENDIX

A1.1 CHOICE OF DISKS:

Use only soft-sectored 5.25" diskettes.
The MITSUBISHI requires 96 TPI (tracks per inch) double sided, diskettes for reliable operation but lower grades can also be used. Most manufacturers better their specifications, so a guaranteed 48 TPI may meet 96 TPI requirements in practical use. The user should decide personally which grade of diskettes he/she wants to buy.
In other words, Datalife MD 525 series (48 TPI, 40 tracks, single sided) selling for £16 per box of 10 can be used in place of MD557 series (96 TPI, 80 tracks, double sided) costing £27 per box of 10.
As a general rule, if the diskette does not format (and verify) perfectly the first time or the second time, do not persist; it will not be reliable and should be returned to the suppliers under normal guarantee.

A1.2 DATA RECORDING DENSITY:

Acorn format is single density. Solidisk DDFS and many other DFS's offer both Single Density, or Acorn compatible format, and Double Density.
You can use diskettes formatted (and recorded) in single density on any BBC computer but not double density diskettes which do not conform to the standard set by a specific DDFS.
Solidisk DDFS will not read OPUS double density diskettes and vice versa.
Most DDFS manufacturers include mass copy commands to enable quick copying of a double density diskette to a single density diskette. Solidisk DDFS does it automatically using, for example, *COPY 0 1 *.*. The first advantage of double density over single density is money saving with bigger storage capacity; the second advantage is speed: data transfer rate between the drive electronics and the computer is 250 kbits per second against 125kbits per second in single density. Although capable of using double density diskettes, many databases (such as STARBASE) and wordprocessors (such as SCRIBE) access the diskettes directly (using OSWORD &7F) and cannot benefit from the extra storage. Solidisk Datafile, VIEW, Beebug Masterfile and Solidisk wordprocessor can.

A1.3 DRIVE NUMBERS AND NUMBER OF USED SIDES

The MITSUBISHI is a double sided disc drive. It has two read/write heads which face each other and hold the media in sandwich when they are loaded. The drive selection is initially fixed by the DS strap (North East corner, next to the data port) in either position 0 or 1. The remaining straps (MM, IU and HS) are preset to the BBC computer standard - *DRIVE 2 selects head 2 on mechanical drive 0, *DRIVE 3 selects head two on the mechanical drive 1. Therefore

*COPY 0 2 *.* will duplicate the right reading side (or top side) of the diskette to the reverse side (or bottom side) of the same diskette. This command will be very often used to make a security backup.

A1.4 TRACK NUMBERS

The MITSUBISHI has a 40/80 track switch found at the rear.
Solidisk DDFS detects automatically a 40-track formatted disk being used with MITSUBISHI and automatically skips alternate tracks.
You should always leave the 40/80 switch in 88-track position.
Certain programs will control the disk drive directly as means of selfprotection (for example BETABASE) and reset the computer (switch it off and on again) before running them.
To control automatic switching from inside your programs, ?&10DE=&FF; to turn it off, ?&10DE=0.

The MITSUBISHI should never be used with diskettes certified for only 35 tracks as data reliability of five tracks from 35 to 39 is not guaranteed when a 35-track disk is used.
Almost all commercially available disks are guaranteed for 40 tracks or 80 tracks.

A2.1 DISK HANDLING

Disk is a precision recording media. Be sure to observe the following precautions:

1) Do not tear, fold or distort the jacket or disk.

2) Do not install a damaged disk in the FDD; a damaged disk not only disturbs the normal read or write operation but also may damage the FDD.

3) Do not touch the open areas of the jacket (magnetic coating area of the disk). Fingerprints left on the disk will cause errors.

4) Return the disk to its envelope for the protection of the window area whenever it is removed from the FDD. The disk should not be left out of the protective envelope (on a desk, for instance) even for a short while.

5) For long-term storage keep the disk in a protective container with the envelope in an upright position.

For short-term storage a few disks may be piled horizontally without a container. Do not lean the disks and do not place any heavy objects, such as books, on the disk as this will cause distortion.

6) Keep and use the disk away from dust. Also do not install a dusty disk into the FDD. Such dust, if carried to the magnetic head, may cause data errors and may shorten the life of the disk and the FDD.

7) Do not clip the jacket. The clipped portion will be distorted.

8) Do not write on the index label of the jacket with a hard-tipped object, such as a lead pencil or a ball-point pen, as these may damage the disk surface. Use a soft-tipped item, such as a felttipped pen.
Generally it is not desirable to write on a label which is already affixed to the disk. Write the label before you attach it to the disk.

9) Do not rub out the information on the label with an eraser as rubbish from the eraser may get into the jacket.

1O) An index label should be applied to the label area shown in fig: 202. Do not apply more than two labels to the same area.

11) Keep the disks away from magnetic fields, such as magnets, transformers, motors, etc. These may degrade the recorded data on the disk. The ambient stray magnetic field should not exceed 50 Oersted.

12) Do not smear the disk with any solvent, such as thinner, freon, or alcohol. It damages the magnetic coating of the disk.

13) Do not expose the disk to sunlight, microwaves or infra-red ray.
Keep the disk away from heaters or stoves. Also do not put the disk on electric apparatus such as a television set.

A2.2 WRITE PROTECT

A write enable notch is located on the right-hand side of the disk. When a disk with an open notch is installed in the FDD, the FDD enables record current to flow into the magnetic head in response to a write command.
To protect the recorded data from accidental erasure due to operational errors cover the notch with a "write protect" tab. Writing or erasing by an erroneous write command is inhibited as the notch cannot be detected.
Be careful not to apply excessive pressure and not to distort the jacket whenever you attach a "write protect" tab to the notch. The tab should not be attached beyond the side line of the jacket. Also the notch should be completely covered by the tab.

A2·3 INSTALLATION AND EJECTION OF DISC

1) INSTALLATION OF DISK

a) Depress the door latch.

b) Take out disk from protective envelope.

c) Hold the label side of the disk and insert it into the Disk Drive with the head window facing the inner side and with the write enable notch located at the LED indicator side.
If the Disk Drive is installed horiontally. the label side goes up.
If the disk drive is installed vertically. with the front lever up. the label side goes to the left.

d) Keeping the disk straight insert fully, and with care, into the Disk Drive.
When it us fully inserted a click sound can be heard. Be sure to insert it fully.

e) Removing your hand from the disk close the front by lowering the door until a click is heard.

2) EJECTION OF THE DISC

a) Depress the door latch until the door swings open; the diskette will be automatically ejected.

b) Put the disk back into its protective envelope.

3) SIGNAL INTERFACE CONNECTIONS:

Signals	Direction	Signal Pins	Return Pins
RESERVED	INPUT	2	1
IN USE	INPUT	4	3
DRIVE SELECT 3	INPUT	6	5
INDEX/SECTOR	OUTPUT	8	7
DRIVE SELECT 0	INPUT	10	9
DRIVE SELECT 1	INPUT	12	11
DRIVE SELECT 2	INPUT	14	13
MOTOR ON	INPUT	16	15
DIRECTION SELECT	INPUT	18	17
STEP	INPUT	20	19
WRITE DATA	INPUT	22	21
WRITE GATE	INPUT	24	23
TRACK 00	OUTPUT	26	25
WRITE PROTECT	OUTPUT	28	27
READ DATA	OUTPUT	30	29
SIDE ONE SELECT	INPUT	32	31
READY	OUTPUT	34	33

POWER INTERFACE CONNECTIONS

VOLTAGE	Terminal Nos.
DC+ 12V.	1
OV	2
OV	3
DC+ 5v.	4

4) INSTALLATION AND REMOVAL OF SHORT BARS

FIRST TURN OFF THE POWER

a) Installation
Install a short bar securely to short the two pins (left and right in parallel). There is no restriction for the approach of the short bar insertion. However, for the easy visual checking of the on-state it is recommended that the metal bar is inserted at the upper side of the post pins.

b) Removal
To protect from unintentional displacement due to vibration, etc., the short bars are very securely inserted. If it is not possible to remove them by hand, pull up slowly with round-nosed pliers. Be careful not to damage surrounding parts.

c) DSO-DS3 Straps
These straps designate the drive number (address) of the FDD in the multiple control by daisy chain connection.

A3. DISK ERROR CODES:

'Enable': *ENABLE was not issued and still required. Fix: enter *ENABLE <RETURN> and repeat the last command.

'Catalog full': the catalog can only hold 31 entries. Fix: use another disk.

'Can't extend': the most inconvenient aspect of the DFS; there is not enough room on the disk to allow a file being updated to expand. Fix: enter CLOSE //0 from Basic, try to copy the file to another disk. This situation can be avoided by organising your disk: save all EXEC files (such as !BOOT), all MENU programs first, all machine code programs next, all BASIC programs and then read-only files and writable files last. Avoid having more than one writable file per disk. *COMPACT the disk before creating the new file, this will allow the new file to expand to the end of the disk.

'Too many': STLDDFS 1.4 can only cope with five opened files at a time. Fix: enter CLOSE //0 from BASIC.

'Read only': the write protect tab is put on the currently used disk.

'Open': the file is already opened. Fix: enter CLOSE //0 before repeating the last command. This situation often arises when debugging your programs as the program drops out without closing opened files and you are attempting to *DUMP or *LIST or *TYPE one of them.

'Locked': the file is protected against overwriting. Fix: enter *ACCESS <afsp> <RETURN>

'Exists': happens when *RENAME an old program using a name already in use. Fix: use a different name or a different directory letter.

'Disk full': the disk is full, the last SAVE will not be completed. Fix: use another disk before *COMPACTing the full one.

'Disk Fault': this will happen sooner or later. Don't panic because the disk has 'crashed'. Fix: check that the 40/80 track switch is not in the wrong position, reset the computer and start again. Check that the software is not 'crafty' and uses some unconventional way of accessing the disk. Check that the disk drive is okay with another disk. Check all other possibilities. The last and annoying cause is that the media is faulty. Fix: use Disk Doctor or similar disc sector editor (Solidisk will offer one) to recover all good sectors, *RESTORE them on to another one. You can avoid this happening by taking great care of your diskettes.

'D.C.': disk changed,.Fix: restart the program all over again.

'Bad Opt': the *OPT has been out of range. The DFS accepts *OPT4, 0 or 1 or 2 or 3.

'Bad name' too long a filename or a filename with non-authorised characters has been used. Fix: change the filename.

'Bad drive': a drive number bigger than three will be rejected. Fix: check the drive number.

'Bad dir': when you request a file, for example R.TEXT, and there is no directory 'R'.

'Not found': the filename is not found. Fix: check the directory, library and disk catalog.

'Syntax': the command is not correctly spelt. Fix: enter *HELP DFS to get the correct syntax.

'Channel': the number given as channel number has not yet been attributed. This happens, for example, when you try to INPUT from an unopened file.

'EOF': the file pointer comes to the end of the file. This happens, for example, when you try to INPUT to the file more than it can hold.

'Bad command': the '*' command issued is not recognised by any ROM or does not correspond to any filename on the currently selected disk.

Solidisk Technology Limited