# SP-DOS

## Spectrum Disk Interface

## User Guide


Watford Electronics

# SP-DOS SPECTRUM

## Disc Operating System

# C O N T E N T S


INTRODUCTION..... a1

INSTALLATION..... b1

OPERATING INSTRUCTIONS..... 1

- CAT (File directory)..... 1
- FILE-TYPES..... 2
- LOADING A PROGRAM..... 4
- FORMATTING..... 5
- SAVING DATA..... 7
- ERASING FILES..... 9
- WILD-CARD ERASE..... 10
- THE 'AUTO' FILENAME..... 11
- THE 'MERGE' COMMAND..... 12
- BASIC OVERLAYS..... 12
- THE 'CLEAR' COMMAND..... 13
- RENAMING DISKETTES AND FILES..... 15
- COPYING DISKETTES AND FILES..... 16
- DISK BACKUP..... 17
- THE MULTI-DRIVE USER..... 18
- HARD COPY OF THE DIRECTORY..... 18
- MULTI-STATEMENT LINES..... 19
- MORE ON 'CAT'..... 19
- RUNNING 'copysys'..... 20
- SEQUENTIAL ACCESS FILES..... 21
- OPEN, CLOSE AND BUFFERS..... 22
- READING AND WRITING..... 24
- INPUT#, INKEY$# and PRINT#..... 24
- 'MAIL/BAS': A SAMPLE PROGRAM..... 28

TECHNICAL SECTION..... 29

I/O PORT MAPPING..... 29

MEMORY REQUIREMENTS..... 29

TECHNICAL SPECIFICATIONS..... 30

DISK CAPACITY..... 31

THE 5.25'' FLOPPY DISK..... 32

RECORDING STANDARDS..... 33

Single Density..... 33

Double Density..... 34

ERROR MESSAGES..... 35

SINCLAIR BASIC ERROR EXTENSIONS..... 35

SP-DOS ERROR MESSAGES..... 37

COMMAND SUMMARY..... 40

SP-DOS USERS REGISTRATION FILE (SURF)..... 44

# SP-DOS SPECTRUM DISK OPERATING SYSTEM



of ABBEYDALE DESIGNERS LTD

## I N T R O D U C T I O N

When Sinclair Research Ltd launched the ZX SPECTRUM computer in the Spring of 1982, very few people could predict its economic, social and educational impact. The various 'computer literacy' programmes had done precious little towards acquainting the very sceptical and reluctant 'lay' humans with a technology that had been in full swing for over ten years. What heavily subsidised official bodies failed to do, materialised in the form of the Mk 14, ZX 80, ZX 81 and ZX SPECTRUM computers: cheap, affordable computing and virtually unlimited expandability and power. Sir Clive Sinclair's aspirations and Da Vincian insight into the technological future have contributed a great deal into forming the shape of computing as we know it.

The ZX SPECTRUM is definitely not a Cray-1 in sheep's clothing. However, although in its unexpanded form it is an excellent and very cost-effective home computer, it can easily grow into a powerful machine of the kind that Apple, Tandy and Commodore used to call a 'Business Computer' and charge thousands of pounds for!

A 'Business' or 'Professional' Computer should be able to be connected to 'Peripherals' such as full-size printers and Floppy Disk Drives. Your SPECTRUM can now be attached to these by means of special boxes of tricks called 'Interfaces'.

An interface is the adaptor which 'matches' the electronic circuitry of your computer to that of the peripheral. A special program must exist within the computer to give the interface the 'intelligence' it is lacking. A printer interface requires a 'printer driver' program while a disk interface needs a 'Disk Operating System'.

A Disk Operating System (DOS) is a master program that controls all the interactions between the computer and its floppy disk drives in an efficient and apparently simple manner. The DOS provides the user with the necessary human interface so that, even when extremely complicated things are being done, the user is simply presented with the desired result rather than a detailed analysis of what exactly took place. The fact that the disk interface is accessed through I/O Ports, the disk is divided into tracks and sectors, the directory has a very powerful and complex structure are matters entirely transparent to the operator who only sees a set of Keyword-Accessible routines performing specific functions like SAVE, LOAD, CAT etc.

Unlike other systems, the Abbeydale Disk Operating System (SPDOS) does not generate a 'DOS command mode' or 'DOS environment' which is independent of BASIC. This was considered to be 'unnatural', cumbersome and requiring unecessary programming skill on the part of the user. SPDOS operates in the background by simply adding extra commands to Sinclair BASIC. You don't have to leave the comforts of BASIC to FORMAT a new disk or rename a file- all the utilities are usable from BASIC.

## INSTALLATION

Your SPDOS Disk System for the ZX SPECTRUM comes to you as a package consisting of:

The SPDOS interface (in a black ABS box)

The System diskette (also containing the disk versions of TASWORD TWO, MASTERFILE and OMNICALC 2)

The manual.

All you need to have your system up and running is your standard SPECTRUM set-up and one to four disk drives. Any drive fitted with the standard Shugart interface (34-way) will do. (The latest 3'' and 3.5'' mini-drives as well as any 5.25'' will do) The non-standard 3'' Hungarian drive (MCD-1) is NOT suitable.

BBC compatible drives, as supplied by WATFORD Electronics and other dealers, will need a 34-way edge connector fitted in place of the IDC connector they come with. WATFORD supply disk drives with the right connector fitted as long as SP-DOS is specified. The drives MUST have their own power supply units.

If you want to use your own drives, a ribbon cable compatible with the TRS-80 Model 1 computer could be used.

First of all ensure that the SPECTRUM power plug is not connected. (You will find that it is rather difficult to plug the interface in with this plug fitted!)

If you are using a compatible printer interface (MOREX etc) you can fit it at the back of the disk interface before proceeding. (For other printer interfaces, check with the manufacturers) Plug the disk interface onto the back of the SPECTRUM making sure it is pushed home.

Connect the drive cable to the interface using the right 34-way ribbon cable and connector. The correct way of doing this is by making sure that the flat cable comes towards you when plugged into the interface.

Switch the disk drive on and insert your System disk in drive 1. (See fig.) The drives are numbered 1 to 4 and the appropriate links inside your drives must be set accordingly. Multi-drive users must make sure that ONLY THE LAST (physically!) drive has its terminating resistor pack connected. By 'last' we mean the drive at the end of the cable and furthest away from the interface. Some terminating resistors are found in the form of a 14-pin DIL (Dual-in-line) package resembling an integrated circuit and as such they can be un-plugged. Some manufacturers (eg. Mitsubishi) use fixed resistor packs and a set of eight links can connect or disconnect them. IF IN DOUBT, ASK!

NOW, CLOSE THE DOOR!

Thread the SPECTRUM's dc power lead through the interface and push it home. (Some users might have to do that BEFORE plugging their interface in)

TURN THE POWER ON...

In most cases, the screen will momentarily clear to black paper and white border, the LED of drive 1 will come ON and whirring noises will follow... Otherwise, just press the RESET button! You are then welcomed by the Copyright message and disk-based computing is available to you.

Before you proceed, you are advised to make a backup copy of your System diskette so that you can store the latter in a safe place. DO NOT remove the little piece of sticky tape from the side of your System diskette as this is the WRITE-PROTECT TAB and you might accidentally erase parts of the disk without it! Backup involves two steps:

1. Copying the system tracks onto a blank diskette.

2. Copying all the copiable files onto the same diskette.

Note that the sub-system disk created by this process will do everything the original System disk could do with the exception of FORMAT.

The first step is performed by running "copysys", a special utility provided for this purpose. (See page 20)

Once this is done, you can transfer the programs from your 'Master' System disk onto your sub-system disk according to the instructions found on page 17 . eg

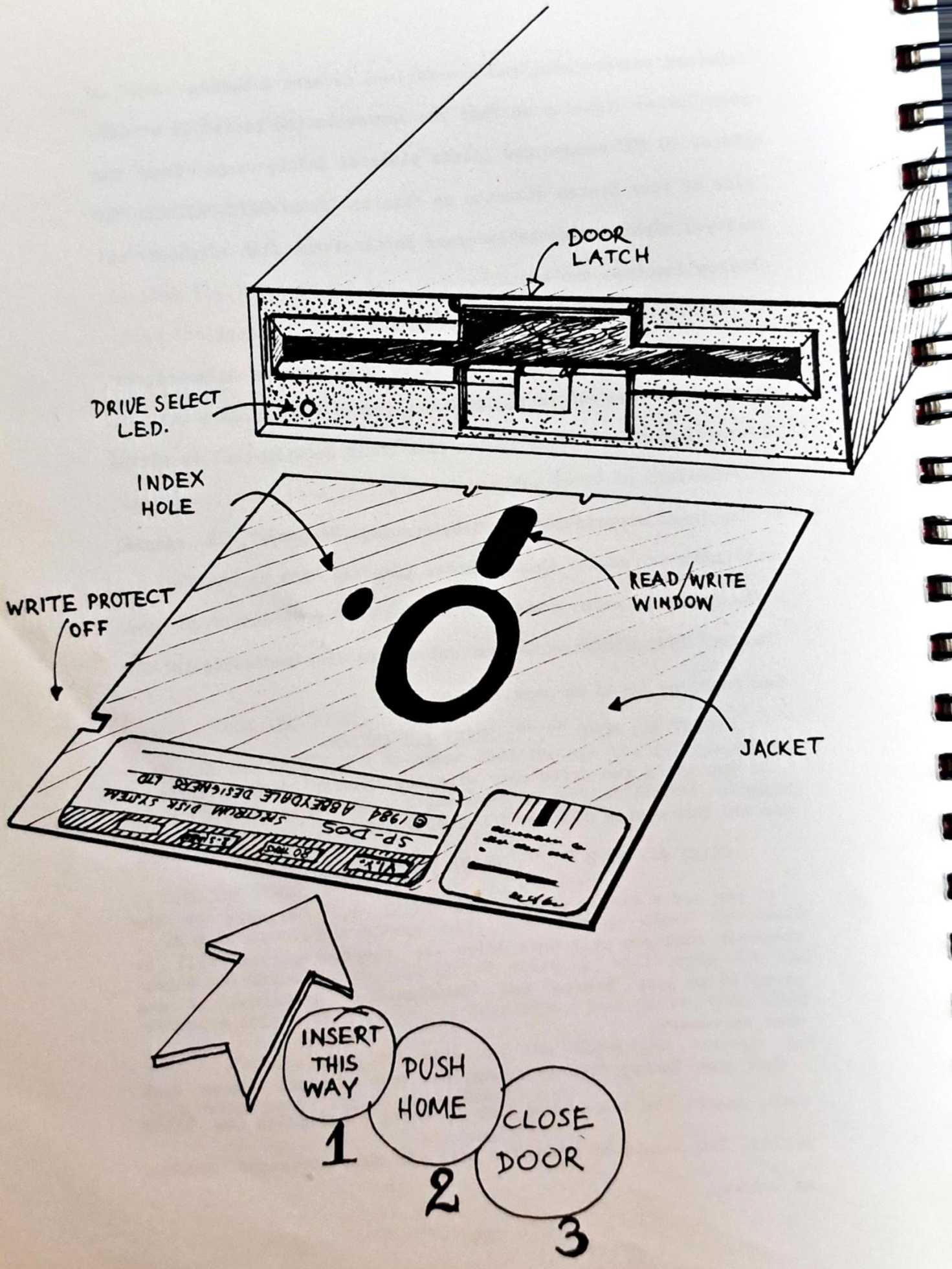
PRINT #4: MOVE "","": PRINT 1,2 <ENTER>

if you are a two drive user with the 'Master' disk on drive 1 and the Sub-system disk on drive 2. Otherwise,

PRINT #4: MOVE "","": PRINT 1,1 <ENTER>

if you are a single drive user. 'PRINT 1,1' signals to the computer that you only have drive 1; therefore, you will be prompted to swap 'Source' and 'Destination' diskettes as and when necessary.

Once your Backup disk is ready, put your original System disk away, insert the sub-system disk in drive 1 and press the RESET button. You should be presented with the same copyright message as before.


DOOR LATCH
DRIVE SELECT L.E.D.
INDEX HOLE
WRITE PROTECT OFF
READ/WRITE WINDOW
JACKET
INSERT THIS WAY
1
PUSH HOME
2
CLOSE DOOR
3

**SP-DOS SPECTRUM DISK OPERATING SYSTEM**



ABBEYDALE DESIGNERS LTD

OPERATING INSTRUCTIONS:

Assuming that the system has been connected up according to the installation instructions, your TV screen will display:

**WATFORD SPDOS V1.1 ,0:0**

**c ABBEYDALE DESIGNERS LTD 1984**

The current version number is 1.1 but membership of the SP-DOS Users Registration File (SURF) will entitle you to updates at minimum cost. The '0:0' printed at the end should remind all SPECTRUM users of the familiar Sinclair BASIC Report format; indeed, SP-DOS uses exactly the same format for its own comprehensive Error Reporting.

Experiment with the RESET button and re-initialise the system a couple of times. This is called 'BOOTING' or 'BOOTSTRAPPING' (!) and it involves loading the Disk Operating System software into memory from the SYSTEM diskette in drive 1. The Disk Operating System or DOS is a very substantial program, written entirely in Machine Code, that takes over from the Sinclair operating system whenever a Disk or Extended BASIC command is detected. An efficiently written DOS can make a tremendous difference when it comes to speed of access or economical use of disk space.

Let's use some of the commands now: As your System disk comes with some excellent software bundled-in, it might be worth having a look at the disk Directory. This is accessed by means of the keyword CAT (CAPS SHIFT+SYMBOL SHIFT+9) which was originally meant to work with the Microdrives.

Normally, when you type CAT, the SPECTRUM comes up with an Error Report indicating that no Microdrive is present. SP-DOS' way of by-passing the Sinclair run-time check is by using a software Switch in front of all disk related commands so that -as soon as this switch is detected- all further processing is done by SP-DOS. This special 'prefix' is the command 'PRINT #4:' which, essentially, assigns stream number 4 to the DISK channel. You need never OPEN this stream for disk access as this is done on initialisation. However, you should never CLOSE #4 because this would disable SP-DOS and you would have to re-boot the system obliterating whatever was in memory at the time.

So type: PRINT #4: CAT <ENTER>

The filetype on the left can be :

'basic' for basic programs.

'bytes' for machine code, SCREEN$, text files etc.

'$data' for string arrays. ("Character array")

'ndata' for numeric arrays. ("Number array")

'sequ' for Sequential Access Files.

The file size is displayed in multiples of 1 kilobyte (1k=1024) as the basic disk space allocation unit (DSAU) is 1 kilobyte or 2 disk sectors. There is no maximum file size; it all depends on the space left on a diskette and the Spectrum memory. Your version of SPDOS supports FULL Sequential Access File handling which virtually eliminates the dependance on memory and makes 800 Kilobyte Data Files a reality!

Now a bit of arithmetic: If you add up all file sizes plus the remaining disk space, you will find that you are a fair number of kilobytes short. Don't worry! This isn't an indication of SP-DOS' numeracy. The truth of the matter is that a SP-DOS directory will only show files that can be called up by name. The disk operating system has been made invisible but it still accounts for the missing space.

Filenames can be up to 10 characters long, they can begin with a letter or anumber and they can contain keywords like SCREEN$ or CODE which, although they count as only one character, can mess up the directory display by creating blank lines between entries.

## LOADING A PROGRAM:

Type: PRINT #4: LOAD "tasword" <ENTER>

Less than 5 seconds later, you are in TASWORD text entry mode and you can start typing your first literary creation... During the 5 seconds it took the system to put powerful Word-Processing at your disposal, SP-DOS consulted the directory 3 times and loaded 3 separate programs amounting to 22 kilobytes altogether. This would have taken over 3 minutes from tape! (assuming that the volume was right and the head was clean!)

Press 'SYMBOL SHIFT' and 'A'. The TASWORD TWO menu is displayed using the familiar 32 character/line format. Press 'd' for 'Directory' and <ENTER>. The directory listing is 'nearly' the same as the one you got earlier. Can you see any difference? Well, the Copyright message shows us that we are now running under MiniDOS- not SP-DOS. MiniDOS is a stripped down version of the Disk Operating System so that it occupies just over 3 kilobytes of user RAM leaving TASWORD with about 16 kilobytes of available Text space- more than 7 Double-spaced A4 pages.

Press <ENTER> to return to the menu. You probably noticed a file called 'tutor/TAS'. This is a 16 kbyte text file that forms a step-by-step tutorial guide into the intricacies of TASWORD TWO. Use 'J' to load it into the TASWORD text buffer making sure you spell the filename correctly and you specify device number 1 as your source drive. (Device 0 is the cassette tape, so if you are a TASWORD user already, all your precious text files can be saved on disk.) You will notice that the 'tutor/TAS' file (the largest possible text file to be created by TASWORD) takes under 3 seconds to load!

## PREPARING A BLANK DISKETTE:

Before you can start using a diskette, it must be FORMATTED.

'Formatting' is the name given to the process of magnetically dividing the diskette surface into 'TRACKS' and 'SECTORS'. (See the Technical Information Section if you are REALLY keen!)

You need to know a few things about your system before you can format a new disk:

a. The number of tracks (40, 80 etc)

b. The number of sides (1 or 2!)

c. The stepping rate (6, 12, 20, 30 ms)

The stepping rate is the time it takes the drive head to travel from one track to the next. We have assigned numbers from 1 to 4 to the standard rates of 6, 12, 20 and 30 ms respectively.

Let's assume that you are a single drive user with a 40-track single sided drive. (200 kilobytes)

With the System Diskette in your drive (and the drive door closed!) press the RESET button to exit MiniDOS. Then type:

PRINT #4: FORMAT "F.Bloggs 1": PRINT 1,40,1,1

and press <ENTER>.

You will be instructed to put the blank diskette in drive 1 (your only drive) and then press <ENTER> - The formatting will then start. The FORMAT progress will be displayed on the screen by means of a 'VERIFIED track xx" message. When the SP-DOS copyright message is shown again, you will have a DATA diskette ready to go! You can get an idea of its capacity by typing:

PRINT #4: CAT <ENTER>

The total formatted capacity is shown to be 200 k but only 195 k is available to you. Again, this is not an error of calculation but simply the fact that 5 kbytes have been reserved for the Disk Directory.

Now let's look at that weird PRINT 1,40,1,1

The keyword 'PRINT' following a SP-DOS command is SP-DOS' way of accepting parameters ie additional information needed by specific Disk functions. The FORMAT command is of the following general form:

PRINT #4: FORMAT "Diskname": PRINT p1,p2,p3,p4

"Diskname" is any valid name up to ten characters long.

p1: The number of the drive that will accomodate the disk to be formatted. (1 to 4)

p2: The number of tracks of that drive.

p3: The number of sides. (1 or 2)

p4: The stepping rate. (1 to 4)

NOTE: FORMAT is not supported by MiniDOS - so you need to be in the SP-DOS Command Mode before you can format a disk. The way to exit MiniDOS is by pressing RESET.

## SAVING DATA:

Replace the System Diskette in drive 1 with a formatted diskette; the SP-DOS module resident in memory is adequate for most things.

Type-in a short BASIC program or LOAD one of your programs from tape. Don't forget that all the tape commands are still perfectly functional. Then type:

PRINT #4: SAVE "prog" <ENTER>

Your program will be saved on your DATA diskette and it will appear on the directory. To test that it has really been saved on disk, the program must be deleted from memory. This can be done with the familiar 'NEW' command or by simply pressing RESET. However, using 'NEW' will also CLOSE #4 and the system must be initialised again by means of the statement:

RANDOMISE USR 58500 <ENTER>

Pressing RESET with a DATA diskette in drive 1, will print a flashing message on the screen and wait for the user to put the SYSTEM diskette in.

Having cleared the program and having made sure that the data diskette is in drive 1, type:

PRINT #4: LOAD "prog" <ENTER>

and LIST. Chances are, your program will be there.

The SAVE command accepts all the standard Sinclair BASIC data formats and conventions. You can SAVE BASIC programs with line numbers for Auto-running, CODE xx,xx , SCREEN$, DATA b(), DATA b$(), CODE USR "a",21*8 .

VERIFY isn't necessary anymore since SAVEing on Disk is followed by an automatic Verify .(Which, in fact, accounts for the time difference between SAVEing and LOADing.)

You can try LOADing a SCREEN$ from cassette. (The HORIZON tape has some nice pics!) Type:

LOAD "" SCREEN$: PRINT #4: SAVE "picture 1" SCREEN$

and start your tape just before the beginning of a SCREEN$.

About 40 seconds later, you will have a picture on your screen and some disk activity will become apparent. When the O.K. report is displayed, clear the screen (CLS) and type:

PRINT #4: LOAD "picture 1" SCREEN$ <ENTER>

How long did it take? Not very long... It was probably about 20 times faster than the tape!

## ERASING FILES:

Erasing a disk file is a very fast operation as SP-DOS does not actually erase the stored file but it simply erases its Directory Entry and reclaims the space allocated to it. The format is straight-forward:

PRINT #4: ERASE "Filename"

Like most commands, an optional drive specification can be appended to the ERASE command in the form of a PRINT p1 statement, where p1 corresponds to a whole number between 1 and 4. (p1 could be a constant, a numeric variable or an expression). Let's now try it out:

Type:

PRINT #4: LOAD "picture 1" SCREEN$: FOR i=2 TO 6:

PRINT #4: SAVE "picture "+ STR$ i SCREEN$: NEXT i

When the O.K. report appears again, have a look at the directory by typing: PRINT #4: CAT <ENTER>

If everything is alright you should find that there are six 7-kilobyte files stored on disk under the very original names 'picture 1', 'picture 2', 'picture 3' etc. Let's now ERASE one of them:

PRINT #4: ERASE "picture 6" <ENTER>

Catalogue the diskette to check whether the file 'picture 6' has gone. Noticed what happened to the displayed FREE SPACE? Deleting 'picture 6' has freed 7 kilobytes.

Your DATA diskette contains now 35 kbytes of experimental junk. (5 remaining SCREENs). To reclaim the space they take, one would have to ERASE the unwanted files one after the other - sometimes the only way! Luckily, in this case, the Filenames are very similar; in fact, they only differ by one character. Under these circumstances, the SP-DOS WILD-CARD character '^' (Up-Arrow) is the best choice. A Wild-card character is Computer jargon for a symbol that can be assigned to ANY alphanumeric character. For example, the string "^^3" describes ALL character strings whose last character is '3'.

So, "picture ^" corresponds to more than 250 different filenames (like 'picture *', 'picture a', 'picture $' etc) and as such, it can be used in the block erasure of files:

PRINT #4: ERASE "picture ^" <ENTER>

PLEASE NOTE: You don't have to ERASE a file before you can SAVE an update under the same filename. SAVEing automatically OVER-WRITES an existing file (unlike the Sinclair Microdrive) as this was found to be common practice in most good Disk Operating Systems.

Erasing a file from a diskette with the WRITE PROTECT tab on, will produce the very comforting report 'Write protected' and will return control to SP-DOS with the file left INTACT.

## THE 'AUTO' FILENAME:

Remove the WRITE PROTECT tab from your System diskette, put the diskette in drive 1 and type:

```
10 PRINT FLASH 1; AT 10,3;"Hello, SP-DOS welcomes you"
20 PRINT #0;AT 0,0;"Press any key to continue": PAUSE 0
30 PRINT #4: CAT: PRINT #0;AT 0,0;"Press any key": PAUSE 0
40 PRINT #4: LOAD "tasword"
```

Now, SAVE this short program on drive 1 as shown:

```
PRINT #4:SAVE "AUTO" LINE 10 <ENTER>
```

Replace the WRITE PROTECT tab, insert the system diskette in drive 1 and press RESET. Interesting? Well, you can put this facility to very good use if, whenever you boot the system up, you go through the same 'chain' of operations or always load the same program. Any program saved with a LINE number and the filename AUTO will AUTO run when you power the computer up or when you press the RESET button. The program need not even be written in BASIC. A machine code program given the name AUTO will also AUTO run from the FIRST memory location it occupies. Obviously, a machine code program does not expect a LINE number when SAVED!

This facility should be even more useful to people determined to keep their software safe from prying eyes; software companies can take advantage of the AUTO feature and produce self-running, un-BREAKable and virtually uncopiable dedicated programs. For better safety, the first line of the AUTO program should disable the BREAK key detection.

And if you are worried about somebody MERGEing your 'protected' BASIC program, the section on the MERGE command should put your mind at rest!

## THE 'MERGE' COMMAND:

The SP-DOS versions of 'MERGE' and 'CLEAR' offer the BASIC language programmer immense programming power by allowing what is known as 'BASIC OVERLAYS' to be constructed and run.

But let's start from the beginning. As all SPECTRUM 'hackers' know, Sinclair's tape 'MERGE' allows the user to disable the Auto-run feature of BASIC programs saved with a line number. (At least, programs not equipped with some sort of Anti-MERGE mechanism!) For some reason, software companies didn't appreciate this subtlety...

Then the Sinclair Microdrive came along with a solution to the problem: Attempting to MERGE an AUTO-running program simply generated a 'MERGE error'. The Abbeydale Designers SP-DOS has gone a step further. Try to MERGE a program saved with the LINE option and you will find that it Auto-runs! So, what? the sceptics amongst you would ask; MERGE has been converted into LOAD - hardly clever! Well, not quite. The program IS first MERGEd with whatever BASIC program is in memory at the time and THEN a GO TO the saved LINE number is executed. You could even Auto-RUN from line numbers which are not in the MERGED module. So, if there is nothing in memory, MERGE does behave like LOAD but that's where the similarity ends!

A large BASIC program could be written in modular form with the individual modules made to occupy the SAME block of Line numbers and SAVEd as separate Auto-running programs. A Master menu would form the main body of the program and the user responses to that menu would simply load the appropriate module (An 'Overlay'), run it and then return to the menu. The modules need to be Auto-running because this way they are protected and can run from ANY line number.

That sounds terrific but what if the MERGED module doesn't use exactly the same line numbers as the one before? Surely, this would lead to errors due to the execution of lines which,simply, shouldn't be there! Well, trust SP-DOS to have a solution to this problem... Your Disk Operating System has also doctored the familiar CLEAR command! If CLEAR is preceeded by the ubiquitous PRINT #4 statement it simply acts as a BLOCK DELETE command! The format is as follows:

```
PRINT #4: CLEAR : PRINT 11,12
```

where 11 is the first line of the block to be deleted and 12 is the last. Of course, you can use the CLEAR command in direct mode to delete BASIC lines whether you intend to use overlays or not.

Let's now design a simple demonstration program using overlays:

```
Lines 1 to 100: Main program
Lines 1000 to 1030: Overlay area.
```

Type in the following:

```
1000 CLS : PRINT AT 10,10;"MODULE 1"
1010 BEEP .5,20
1020 PRINT AT 12,5;"IT DOES NOTHING!"
```

Now, `PRINT #4:SAVE "module1" LINE 1000 <ENTER>`

Delete the lines by typing:

```
PRINT #4: CLEAR: PRINT 1000,1020 <ENTER>
```

Next:

```
1007 PRINT #4: CAT
1010 PRINT "Press any key":PAUSE 0
1030 CLS: PRINT AT 10,10;"MODULE 2"
```

Also, PRINT #4: SAVE "module2" LINE 1007 <ENTER>

Delete the lines as before.

Next: 1000 GO TO 1030

1011 CLS: PRINT AT 10,10;"MODULE 3"

1021 FOR i=1 TO 10: BEEP .05,15: BEEP .05,20: NEXT i

1027 GOTO 1000

1030 PRINT AT 12,12;"*** END ***"

PRINT #4:SAVE "module3" LINE 1011 <ENTER>

Delete these lines as well. And now the main body:

```
10 CLS:PRINT INVERSE 1; AT 0,10;"---- DEMO ----"
20 PRINT AT 8,10;"1.... Module 1"
30 PRINT AT 10,10;"2.... Module 2"
40 PRINT AT 12,10;"3.... Module 3"
50 PRINT #0;AT 0,0;"Choose option 1,2 or 3";
60 IF INKEY$="" THEN GO TO 60
70 LET i$=INKEY$: LET i=CODE i$-48
80 IF i<1 OR i>3 THEN GO TO 60
90 PRINT #4: MERGE "module"+STR$ i
100 STOP
2000 PRINT #0;AT 0,0;"Press any key to return to the
     MENU";: PAUSE 0
2010 PRINT #4: CLEAR :PRINT 1000,1030: GO TO 10
```

Now save the main program:

PRINT #4: SAVE "mainprog" LINE 10 <ENTER>

Type 'RUN' and see what happens!

## RENAMING DISKETTES AND FILES

The name assigned to a diskette during FORMAT can easily be changed by using the MOVE command. Similarly, a file name can be changed as well. The SP-DOS version of MOVE is extremely versatile as it doubles up as a RENAME and file COPY utility. Let's deal with renaming first:

The required syntax for RENAME is:

PRINT #4: MOVE "Filename1","Filename2": PRINT d;

where: Filename1 is the existing Diskette or File name

Filename2 is the new name we want to assign to the diskette or file.

d is the drive number specification.

; is an optional delimiter which forces disk swapping prompts to be displayed.

eg PRINT #4: MOVE "mainprog","MAINPROG":PRINT 1

will first check to see whether the disk in drive 1 is called 'mainprog' and if this is not the case it will search the directory of drive 1 for a file named 'mainprog'. If this file is found, it will be RENAMED 'MAINPROG'. If it is not found a 'Record not found' report will be given.

If 'mainprog' happened to be the diskette name, SP-DOS would look no further; it would simply rename the diskette 'MAINPROG'.

Make sure that the WRITE PROTECT tab is removed before attempting to RENAME as the MOVE command needs to update the directory by writing into it.

## COPYING DISKETTES AND FILES:

Not all disk files can be copied neither is a FULL System diskette back-up possible. However, your own data diskettes can be copied completely.

There are two levels of file transfer:

a. A single file copy.

b. A diskette copy.

The syntax for a single file copy is:

PRINT #4: MOVE "Filename1","Filename2": PRINT d1,d2;

In this mode, Filename2 is a file in drive d2 and it could be a NULL string and Filename1 is a file in drive d1. d1 is the SOURCE drive number and d2 is the DESTINATION. If Filename2="" then a copy with the same name is created.

The drive numbers can be the same (1 to 4) as this allows the single drive user to back his files up. When SP-DOS senses that d1=d2, it prompts the user to swap diskettes as and when necessary. The optional semicolon (;) also forces a prompt even if the drives specified are different.

eg. PRINT #4: MOVE "MAINPROG","MAINPROG": PRINT 2,3

"MAINPROG" from drive 2 will first be found and loaded, and then the file will be copied across to drive 3 using the same filename.

eg. PRINT #4: MOVE "MAINPROG","NEWNAME": PRINT 1,1

If you are thinking that this is equivalent to RENAME, you are wrong! The file called "MAINPROG" will be found and loaded from drive 1 and then SP-DOS will ask the user to put the DESTINATION diskette in drive 1. A copy will then be made on the destination diskette and it will be assigned the name "NEWNAME".

See the difference? Specifying TWO drive numbers, even if they are the same, instructs SP-DOS to perform a file copy- not just fiddle around with the directory.

The syntax for a diskette copy (BACKUP) is:

```
PRINT #4: MOVE "","": PRINT d1,d2;
```

and all the copiable files of drive d1 are copied across onto a formatted diskette in drive d2. Again, if d1=d2 (Single drive user) the user is prompted to perform the appropriate swaps. If the optional semicolon is used, the user is also prompted to place the right diskette in the drives.

Attempting to use the diskette copy utility on your System diskette will not give you another System diskette! The special utility "copysys" must be used and even so you can only produce a 'SUB-SYSTEM' disk.

If you want to transfer all the copiable files of the system disk across to another drive, say drive 2, if you place the destination disk in drive 2 and then type:

```
PRINT #4: MOVE "","": PRINT 1,2
```

copying will take place straight-away without any prompts. (This is why the semicolon has been omitted)

## THE MULTI-DRIVE USER

If you are fortunate enough to have more than 1 drive, you should make sure that drive 1 contains your WRITE PROTECTED System diskette while all other drives are loaded with data diskettes. In this case, drive number specifications become very important particularly if you want to access a drive other than the one you accessed last. As we mentioned earlier, the drive specification is a parameter (usually optional) needed by commands such as SAVE, LOAD, ERASE, CAT etc. Remember how parameters are passed on to SP-DOS? A 'PRINT' statement following the disk command. eg:

PRINT #4: LOAD "FRED": PRINT 2

The number 2 refers to the second drive. Once a drive has been selected for an operation, it becomes the DEFAULT drive and it doesn't have to be specified again as long as you are working with it; so the PRINT statement can be omitted. As the default drive is the one whose directory is consulted, any reference to files stored on different drives will produce a "Record not found" report unless the drive number is also specified.

PLEASE NOTE: The drive number parameter is required by the FORMAT and MOVE commands as well as the selective CAT.

## HARD COPY OF THE DIRECTORY:

If you want the CATalogue of files to be directed through your printer interface to a suitable printer, first ensure that the interface driver software is loaded. Then:

OPEN #2,"p": PRINT #4: CAT : CLOSE #2 <ENTER>

## MULTI-STATEMENT LINES:

The SP-DOS commands are perfectly happy to share a program line with other BASIC statements. The only point one should be cautious about, concerns the ordinary PRINT statement. As PRINT is used to pass parameters onto SP-DOS, the operating system looks for it after a disk command that didn't end with Carriage Return (ENTER). If a PRINT IS found immediately after, it is taken to relate to a drive specification. To avoid this from happening to your own, innocent PRINT statements, use a double colon after the disk command:

```
PRINT #4: LOAD "prog1":: PRINT "hello there!"
```

Of course, if you DO want to select a drive, use the standard:

```
PRINT #4: LOAD "prog1": PRINT 2: PRINT "hello there!"
```

## MORE ON CAT:

The main SP-DOS CATalogue command can only operate with a system diskette on drive 1. (Unlike MiniDOS CAT which executes from a RAM resident module)

The full syntax for SP-DOS CAT is:

```
PRINT #4: CAT : PRINT d,string ;
```

where d is an optional drive specification, 'string' is an alphanumeric string for selective display and ' ; ' prompts the user to put the system diskette in drive 1.

The provision for a selective directory was rendered necessary by the fact that up to 144 (!) files can be stored on a single diskette. Displaying their filenames -even 22 at a time- takes a fair number of screenfuls!

Here is an example:

PRINT #4: CAT : PRINT 2,"/TXT"

Only the filenames CONTAINING the string "/TXT" will be displayed. If the string select facility is used, specifying the drive by means of 'PRINT d' becomes compulsory.

As mentioned earlier, MiniDOS supports its own, System disk independant, CAT function. The syntax is simply:

PRINT #4: CAT :PRINT d

where d is an optional drive specification. The selective Directory facility is not available .

## RUNNING COPYSYS

'copysys' is a special Utility which allows the user to back up the system diskette. Have a blank diskette handy and make sure you know the characteristics of your drives (Tracks, sides, stepping rate). Then type: PRINT #4: LOAD "copysys": PRINT 1

(The original system disk must be in drive 1)

A MENU will be displayed and you can answer the various questions by pressing SPACE and ENTER. (Avoid BREAKing...)

When you are satisfied with the set-up press SPACE in response to 'Any changes' and then ENTER. The standard disk FORMAT utility will then be activated in order to prepare the disk. After formatting, the System tracks are written onto the disk and the Spectrum returns to the "Sinclair Research" message. Pressing RESET will take you back to the SP-DOS operating environment. Bear in mind that:

ONLY YOUR ORIGINAL MASTER DISK

CAN FORMAT DISKS OR RUN "copysys".

## SEQUENTIAL ACCESS FILES:

So far your data files have been 'bytes' (eg TASWORD TWO), string arrays (eg MASTERFILE) or numeric arrays. Sinclair BASIC is probably one of the few languages to allow such versatility from a tape-based system. However, business programs tend to be rather memory consuming and so the amount of RAM (Random Access Memory) left for data variables is limited.

The Abbeydale SP-DOS BASIC overlaying feature allows massive BASIC programs to be written using very little memory but the problem of large Data files still remains unsolved.

A "SEQUENTIAL ACCESS" file is a file consisting of individual data "RECORDS". A data Record is an alphanumeric string or numeric data whose end is marked by a "terminating" character. (A comma, an apostrophe or more commonly, the Carriage Return code) The beginning of a data Record is usually defined as the character after the terminator of the previous Record.

So, you can consider a Sequential Access file as a collection of n Records numbered from 1 to n, which can only be read starting with Record 1 and proceeding to 2, 3 etc until the n-th Record has been accessed. An attempt to read the non-existent (n+1) Record will produce an "End of File" error report. How big the number n is does not depend on the amount of available memory but on the available space on the data diskette and so, the demand on memory is very modest.

The SP-DOS commands necessary for Sequential Access file handling are:

OPEN #, CLOSE #, INPUT #, INKEY # and PRINT # .

## OPEN, CLOSE AND BUFFERS:

Whenever a file is used one or more "BUFFERS" have to be assigned to it. A buffer is a 512 byte chunk of memory which can be thought of as a 'waiting area' that data must pass through on the way to and from the disk file. This assignement is done by means of the OPEN # statement and cancelled with a CLOSE # statement. So, a buffer is similar to an observation window, probably covering a few records at a time, which moves through the file from beginning to end. Only one window-full of information need be in memory at any one time.

Before a file can read or written into it has to be OPENed. Opening a file is the process of creating a buffer for it, making this buffer accessible to the user by attaching it to a stream and determining whether the operation is Reading or Writing. If the file already exists, it is automatically opened for reading otherwise it is opened for writing. An existing file can be opened for reading into more than one buffers by opening it to different streams (memory allowing!). However, the report "File already open for writing" is printed whenever an attempt is made to re-open a newly created (and hence 'Write') file.

The syntax for OPEN is:

```
PRINT #4: OPEN #n, "Filename": PRINT d
```

where n is the stream number assigned to the buffer and it must be in the range 0-15 but NOT 4. (Stream 4 is permanently assigned to the DISK channel by SP-DOS)

" : PRINT d " is again an optional drive specification. (d must be between 1 and 4). If not used, the current drive is searched for "Filename".

NOTE: A stream MUST be CLOSED before it can be re-opened.

PRINT #4 : CLOSE #n frees stream n and reclaims the buffer assigned to it. Using stream n without opening it again will cause a "Invalid stream" error report to be printed. No drive specification is required by the CLOSE statement.

CAUTION: Whenever a buffer is created, the BASIC program and its variables are "pushed" up by about 570 bytes. As a result of this, machine code programs living in a REM statement will not execute properly unless they are fully re-locatable.

Furthermore, having all 15 possible buffers open accounts for about 8 kilobytes of buffer space, so watch your program length.

DO NOT remove a diskette which contains an open file without closing the file first! Closing a file opened for writing saves its buffer onto disk and sets the End-of-file marker.

## READING AND WRITING:

Reading from a sequential access file is handled by two statements:

INPUT # and INKEY #

INPUT # is very similar to the familiar keyboard INPUT: It reads a whole record up to its terminator but the keyboard is replaced by an open disk file. The data is input sequentially. That is, when the file is first opened, a pointer is set to point to the beginning of the first record. Each time a record is read, the pointer advances to the beginning of the next record and so on. To go back and read from the beginning of the file, you must close the file and open it again.

The syntax for INPUT # is:

INPUT #n;variable1;variable2;variable3;....;variable n

To INPUT # sequential data successfully from a disk file, you will have to know the exact format of that data. ie whether a certain record must be read into a string or a numeric variable and what kind of terminator was used to save it. If the record contains characters which the normal keyboard INPUT does not accept, INPUT # will not accept either. Try using:

INPUT #n; LINE f$ etc. (This will only work for string records.)

A slower -but always successful- method of reading ANY character from a disk file involves the use of INKEY$ # which returns the next character in the file irrespective of whether INPUT accepts it or not:

```
10 PRINT #4 : OPEN #5,"textfile" : PRINT 1
20 FOR i=1 TO 1000
30 PRINT INKEY$#5;
40 NEXT i
50 PRINT #4: CLOSE #5
```

Assuming that a file called "textfile" already exists and contains at least 1000 characters, this little program will read it and print out its contents. If there are less than 1000 characters in the file, an "End of File" report is printed.

PLEASE NOTE: You can OPEN streams 0-3 for sequential Input or output and when you CLOSE them, the Sinclair Operating System re-assigns them to the Default Channels ie Screen, Keyboard and Printer. Stream 3, usually attached to the printer, will accept LPRINT and LLIST as well as PRINT #3 and LIST #3. Therefore, if you OPEN #4: OPEN #3,"printfile" all the output meant for the printer will be stored in a disk file which can then be read and printed (normally after the main task has finished).

DO NOT USE OPEN or CLOSE without the prefix PRINT #4; the streams will be freed but the buffers will not be reclaimed and the file will not be CLOSED properly.

Writing into a sequential access file is done with the statement:

PRINT #n; variable1 ' variable2 ' variable3 etc etc.

As PRINT #n behaves just like screen PRINT, the use of terminators and separators is very important. PRINT #n accepts the three standard separators:

semicolon (;)

comma (,)

apostrophe (')

Special care must be taken when using a semicolon as a separator because it will merge two records just like it does on the screen; as a result, the two records have to be read-in as one! eg Let's assume that a$="Joe" and b$="Bloggs"

Then: PRINT #5;a$;b$

would be saved as:

JoeBloggs <CR>

which could not be read back into two variables. However:

PRINT #5;a$;CHR$13;b$

would be saved as:

Joe <CR> Bloggs <CR>

which is OK.

The use of a comma (,) can also produce peculiar effects. If two variables are separated by a comma, then:

```
PRINT #5;a$,b$
```

is saved as:

```
Joe             Bloggs <CR>
```

When INPUT #n is used to read file records, the arrival of each character causes the characteristic keyboard 'click' normally associated with INPUT from the keyboard. It is good practice to alter the value of the system variable PIP if the highest possible transfer rate is required as the lower the frequency of the 'click', the longer it takes to be generated in software.

Try the following:

```
50 GOSUB input
 .
 .
1000 REM subroutine 'input'
1010 LET a=PEEK 23609 :REM old value of PIP
1020 POKE 23609,1     :REM new value of PIP
1030 INPUT #n;n$;a$;n
1040 POKE 23609,a     :REM restore old value
1050 RETURN
```

A sample program called "MAIL/BAS" is provided on the system diskette and you are advised to list it and study it in detail. It is a rather primitive Mailing list program which makes use of a large number of SP-DOS facilities. A file consisting of names, addresses, phone numbers etc can be created, examined, printed, edited etc. The file can be as large as your available disk space although its sequential nature makes access rather slow: to reach the n-th record, all the previous records must be read. To run this program, simply type:

PRINT #4: LOAD "MAIL/BAS" <ENTER>

In order to look at the program, you need to BREAK at an appropriate moment ie when ALL files are closed. The main MENU is the best choice but if you have to BREAK at some other point or if an error report has been generated, you must type:

GOSUB close <ENTER>

and in some cases:

PRINT #4: CLOSE #6 <ENTER>

A sample file, MAIL/DAT is also provided for you to experiment with.

## TECHNICAL SECTION

### I/O PORT MAPPING

The Watford/Abbeydale Designers Floppy Disk Interface for the Sinclair ZX SPECTRUM computer uses a number of I/O Ports for its communication with the computer. Single address line decoding, similar to the method used in the Spectrum, has been utilised throughout for circuit simplicity. The following address lines should not be used by external peripherals:

A1, A3, A4, A5.

As a result, Sinclair's Interface 1 is NOT compatible with this Disk System. A2, A6 and A7 can still be used by printer interfaces etc. The Abbeydale Centronics/RS232 interface marketed by MOREX PERIPHERALS LTD, WATFORD ELECTRONICS and other dealers, makes use of Address lines A2 and A7 and is 100% hardware and software compatible with SP-DOS.

### MEMORY REQUIREMENTS

On initialisation, RAMTOP is set to 57855 to allow for the RAM resident module of SP-DOS, the sector buffer, the system variables, the overlay areas, a printer driver area (64290 to 65367) and the User Defined Graphics. This division of the memory map applies to the main SP-DOS only. MiniDOS occupies just over 3 kilobytes plus its system variables and sector buffer and lowers RAMTOP to 48999. Memory from 54784 upwards is free and can be used for user programs and drivers.

## TECHNICAL SPECIFICATIONS:

The WATFORD/ABBEYDALE DESIGNERS Disk Interface can operate with any Shugart compatible 3'', 3 1/2'' and 5 1/4'' floppy disk drive that is capable of Double Density operation. The following drives have been tested and performed perfectly:

Mitsubishi 5 1/4 '' 80 track/Double-sided.

TEC 5 1/4 '' 80 track/Single-sided.

Pertec 5 1/4 '' 35 track/Double-sided.

Micropolis 5 1/4 '' 35 track/Single-sided.

Hitachi 3'' 40 track/Single-sided.

Hitachi 3'' 40 track/Double-sided.

TEAC, CANON, CHINON, SONY, SIEMENS, EPSON, TANDON, CONTROL DATA, SHUGART, QUME and others should also work with no problems provided they can handle the Double-Density recording method.

Data is written onto and read from the disk at a rate of 250 Kbits/second or 1 byte in every 32 microseconds. As one disk revolution, ie one track, takes 0.2 seconds, the theoretical (unformatted) capacity of a track is 50000 bits or 6250 bytes.

An 80 track, Double sided drive will therefore have a capacity of 2*80*6250= 1,000,000 bytes unformatted. Quoting this capacity is a good advertising gimmick but it is far from the "true" capacity of the drive, ie the storage available to the user.

Formatting a diskette usually sacrifices 34 % of this capacity (Double density) or even 59 % (Single density)! SP-DOS has been highly optimised in order to offer probably the highest possible efficiency out of a disk system with a loss of only 18 % !!

Each SP-DOS track is organised as ten 512-byte sectors. So each track can hold 5120 bytes and the following capacities can be achieved:

| TRACKS | SIDES | UNFORM. | FORMATTED |
|---|---|---|---|
| 35 | 1 | 218750 | 179200 |
| 35 | 2 | 437500 | 358400 |
| 40 | 1 | 250000 | 204800 |
| 40 | 2 | 500000 | 409600 |
| 80 | 1 | 500000 | 409600 |
| 80 | 2 | 1000000 | 819200 |

As SP-DOS can handle up to 4 Double-sided drives, the maximum overall storage capacity is 4*819200 or 3,276,800 bytes.

## THE 5.25'' FLOPPY DISK:

This industry standard magnetic storage medium is basically a circular piece of flexible plastic (5.25'' diameter) which has been coated with a magnetic material similar to the one used for magnetic tapes. The flexible disk is then placed in a protective jacket which allows the disk to rotate freely.

Before a floppy disk can be used, it has to be formatted ie magnetically divided into a number of concentric circles known as TRACKS. Originally, 35 tracks per surface was the norm. 40 tracks followed soon after, while developments in the field of magnetic recording made the 80 track/surface a reality. The track density was thus increased from 48 tpi (tracks per inch) to 96 tpi. This latter density, used with double density recording, has become known as QUAD DENSITY.

Each track is logically divided into SECTORS and the sector boundaries can be identified by means of HARD or SOFT SECTORING. Hard sectoring -virtually abandoned to-day- marks the beginning of each sector with a physical hole. The beginning of the 1st sector and end of the last, is identified by an extra hole known as the INDEX. The disadvantage of this method is that each track has a FIXED number of sectors and hence a fixed capacity.

Soft sectoring (used by SP-DOS) uses an INDEX hole to identify the boundary between the last and first sectors. There are no other holes, so the division into sectors is done in software. A track can then be divided into any number of sectors (within limits!), usually 5,10,16,26.

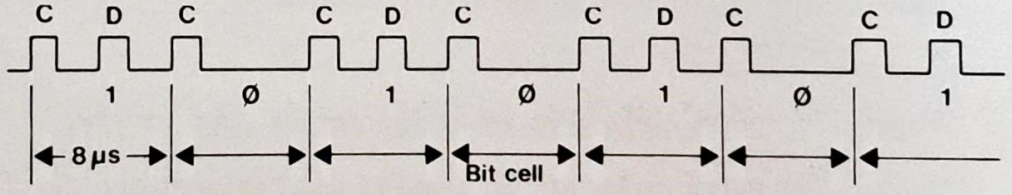
Formatting a soft-sectored diskette identifies and numbers each sector of each track. Information such as the track and sector numbers, the side number, the number of bytes per sector, error detection checksums etc is recorded on each sector so that positioning of the Read/Write head on the right track and sector can be done reliably by simply reading this information off the disk. GAPS of FILLER bytes are recorded between sectors and around the INDEX hole to eliminate the effect of WRITE-SPLICING.

**RECORDING STANDARDS**

a. SINGLE DENSITY or IBM 3740 standard.

This is also known as F.M. (Frequency Modulation).

Data is recorded SERIALLY on each track. Each bit defines an 8 microsecond BIT CELL at the beginning of which, a 200 nanosecond CLOCK bit is recorded. If the data bit to be stored is a logic 1, a flux reversal is inserted in the middle of a bit cell. For a logic 0, no flux reversal is recorded:




As the length of a bit cell is fixed to 8 microseconds, the 'capacity' of a track can be calculated:

No of bits/track=(Time for 1 revolution)/(Time for 1 bit)=

= 0.2 sec/8 microsec= 25,000 bits=

= 3125 bytes/track.

This is obviously the UNFORMATTED capacity.

The data rate can also be found:

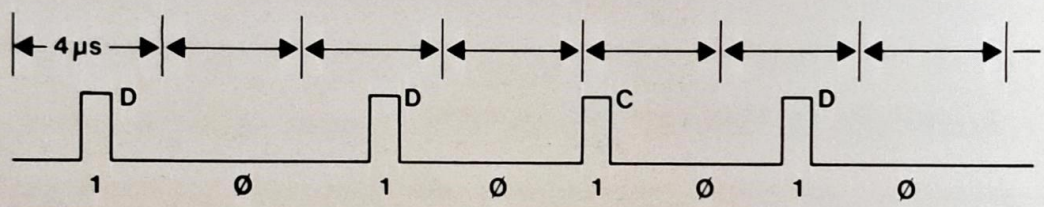
Data rate= 25,000 bits/ 0.2 sec = 125,000 bits/sec.

Single-Density is very easy to implement and works reliably even with low-quality drives. Disk input/output routines are easy to write as data bytes have to be read or written at the rate of one in every 64 microseconds.

b. DOUBLE DENSITY or IBM SYSTEM 34 standard.

This is also known as M.F.M. (Modified F.M.).

Data is also broken into bit-cells, with the data bit written in the centre of the bit-cell if it is a logic 1. Clocks are only written if BOTH surrounding data bits are logic 0. The bit cell is now 4 microseconds long and hence, twice as much data can be stored without increasing the frequency rate:




RULE: 1. WRITE DATA BITS AT CENTRE OF BIT CELL if DATA=1.

2. WRITE CLOCK BIT AT LEADING EDGE OF BIT CELL if:

(a) No Data bit has been written last.

(b) No Data bit will be written next.

Track capacity= (0.2 sec)/(4 microsec)= 50,000 bits= 6250 bytes.

Effective rate= (50,000 bits)/(0.2 sec)= 250,000 bits/sec.

Byte servicing time is now only 32 microseconds which can only be achieved by using a fast microprocessor and highly optimised code.

## ERROR MESSAGES

The following description of error messages includes some already used by the SPECTRUM. The explanation below outlines any differences from the normal meaning.

4 Out of memory :May occur on a LOAD, MOVE or MERGE command and normally means RAMTOP is too low for the file to fit in existing memory. This problem can be cured using a CLEAR n statement to raise RAMTOP. If this error occurs while MOVING a sequential file it may be too large to MOVE in this way and should be copied by re-writng to the new file record by record.

6 Number too big :Used to check the values of parameters eg. the drive number in a disk command. Check your command carefully. If the fault persists it may be due to a DOS error which should be cleared by re-booting the system. This is most likely to occur following a "Drive inoperable error".

8 End of file :An attempt has been made to read beyond the end of a sequential access file.

B Integer out of range: See error 6.

C Nonsense in BASIC: See error 6.

F Invalid file name:This may occur if a SAVE command uses a filename longer than 10 characters. For all commands apart from ERASE, a "^" in the filename will also produce this error.

H STOP in INPUT :You have used INPUT with a sequential file which contains non-print characters. Use INKEY$ for this type of file.

J Invalid I/O device:This will occur if you attempt to write to a file open for reading and vica-versa. If the error occurs when you read AND write to a particular channel it means an error occured when it was OPENed and it should be CLOSEd before proceeding.

O Invalid stream :The stream has not been OPENed. If this occurs for channel 4 you must re-boot the system or use RANDOMISE USR 58500.

Q Parameter error :See error 6.

## SPDOS errors

The following messages have been added to the system:

S Drive inoperable: You have specified a drive number not used by your system, the door is open, the disk is inserted incorrectly or the drive is faulty. Restore normal operation by switching back to the drive you were using before (CAT etc)

T Disk data lost : A DOS error has occured. Try repeating the command after a CAT. If this fails the disk may be faulty.

U CRC error : See error T.

V Record not found: This normally means the filename used in your command does not correspond with any on the disk. Check the spelling against a CAT. Note that keywords or special characters must be used consistently. If this is not the cause of the problem suspect the disk.

W Write protected : You have attempted to write to a disk with a tab over the write-protect notch. This will also occur if you attempt to OPEN a file that does not exist on a write-protected disk. The channel must then be CLOSEd to remove the error condition.

X Disk Faulty :The system has failed to read or write to the disk after 50 attempts. Retry the command and then suspect the disk.

Y Illegal command :If this occurs for a FORMAT command you have not used your master system disk while executing this command. For all other cases you have attempted to use a command not supported by the system and not saved as an executive file. Note mdos does not support all DOS commands.

Z Copyright message

a Wrong filetype :Occurs when attempting to load a file with the wrong specification. E.G. LOAD "filename" can not be used to load a CODE file without the keyword CODE. This error will also occur if you attempt to OPEN a file not produced as a sequential file.

b Undefined error :You have used an undefined command or routine in the DOS.

c Directory full :144 entries have been used in the directory for this disk. There may be fewer than 144 files as extra entries are used for sequential files larger than 48k and for all files if the disk is becoming full. The problem may be solved by doing a complete copy of the disk which will ensure efficient use of the directory entries.

d Disk full :All space on the disk has been used. The last save operation will have been terminated and the file should be saved on another disk. If this occurs with a sequential write file a CLOSE will not work in the normal way and some data may be lost.

e Filename in use :For the MOVE command this warns that a file by this name exists and should be ERASEd before the command is retried.

f Protected file :An attempt has been made to MOVE a protected file.

g Stream already open: The stream is already OPEN for a DOS file and should be CLOSEd before an attempt is made to OPEN it again.

h File open for writing: A file by the same name is already OPENed for writing to the same drive. You may only have a file OPENed to several channels if it already exists and is therefore OPENed for reading.

## COMMAND SUMMARY

ALL commands with the exception of INPUT#, PRINT# and INKEY# are preceded by: PRINT #4 :

(The letter 'M' in brackets indicates that the command is MiniDOS compatible.)

| | | |
|---|---|---|
| CAT | (M) | Gives a catalogue of all files of the CURRENT drive. |
| CAT :PRINT d | (M) | Gives a catalogue of all files of drive d (1-4) |
| CAT :PRINT d, "string" | | Gives a catalogue of all files of drive d, whose name CONTAINS "string". |
| CAT :PRINT d, "string"; | | Same as before, except that the user is prompted to put a SYSTEM diskette in drive 1. "string" could be NULL. |
| CLEAR :PRINT m1,m2 | | Deletes a block of BASIC lines starting at m1 and ending at m2. 0<m1,m2<=9999 |
| CLOSE #n | | Closes stream n and reclaims the buffer. 0<=n<=15. n<>4. |

| | | |
|---|---|---|
| ERASE "filename" | (M) | Erases "filename" from the directory of the CURRENT drive. UP-ARROW acts as a WILD-CARD character. |
| ERASE "filename": PRINT d | (M) | Erases "filename" from the directory of drive d. |
| FORMAT "diskname": PRINT i,j,k,l | | Formats disk on drive i with j tracks, k sides and l step rate. |
| INKEY$ #n | | Reads the next character of a sequential file opened to stream n. |
| INPUT #n;var1;var2;var3 etc | | Reads the next record of a sequential file opened to stream n. |
| LOAD "filename" | (M) | Loads or loads and Auto-runs the BASIC program "filename" from the CURRENT drive. |
| LOAD "filename": PRINT d | (M) | As above but the program is found in drive d. |
| LOAD "filename" filetype | (M) | As above but filetype can be CODE, SCREEN$, DATA a(), DATA a$() etc. PRINT d can still specify the drive. |

| | | |
|---|---|---|
| MERGE "filename" | (M) | Merges or merges and Auto-runs the BASIC program "filename" with the program in memory. |
| MERGE "filename": PRINT d | (M) | As above but program is found on drive d. |
| MOVE "file1","file2" | | Renames "file1" to "file2" in the CURRENT drive. |
| MOVE "file1","file2": PRINT d | | Renames "file1" to "file2" in drive d. |
| MOVE "file1","file2": PRINT d; | | As above but disk swapping prompts are displayed. |
| MOVE "file1","file2": PRINT d1,d2 | | Copies "file1" of drive d1 into drive d2 using the name "file2". |
| MOVE "file1","": PRINT d1,d2 | | Copies "file1" of drive d1 into drive d2 using the same filename. |
| MOVE "","":PRINT d1,d2 | | Transfers ALL copiable files of drive d1 to drive d2. |

For single drive users, d1=d2 and prompts are displayed automatically. Otherwise, prompts can be forced by using a ';' after the drive specification.

| | | |
|---|---|---|
| OPEN #n,"filename" | | Opens stream n to sequential file "filename" of the CURRENT drive, and assigns a buffer to it. If the file already exists, it is opened for READING, otherwise it is opened for WRITING. |
| OPEN #n,"filename": PRINT d | | As above but drive d is used for "filename". |
| PRINT #n;var1'var2'var3 etc | | Records var1,var2 etc are written into buffer n for subsequent transfer to a sequential file. |
| SAVE "filename" | (M) | Saves the BASIC program "filename" onto the CURRENT drive. |
| SAVE "filename" LINE m | (M) | As above but program is to Auto-run from line m. |
| SAVE "filename" filetype | (M) | Filetype can be CODE, SCREEN$, DATA a(), DATA a$() etc.<br><br>PRINT d can still be used to specify a drive other than the current one. |

SPDOS V1.2 ADDENDUM 26/07/85



## COMMAND SUMMARY

ALL additional commands are preceded by :PRINT £4:

None of the additional commands are used by minidos.

| | |
|---|---|
| CLEAR 0 | This command is used to reduce the memory used by a BASIC program by using the most compact form to represent numbers. |
| COPY :PRINT d | Used to transfer programs from tape to disk. It will transfer blocks with or without headers up to about 33 k bytes. The conversion to use the disc can then be carried out by the user. |
| GOTO n | If an error occurs the program will resume at line n. The error type can be established from the usual system variable ERR NR at location 23610. |
| NEW | This removes a BASIC program and its variables but leaves SPDOS intact. |
| MOVE "","string" : PRINT d1,d2 | This form of the MOVE command will move all files containing the string "string" from drive d1 to drive d2 providing a wildcard MOVE. |

In addition to the above commands a number of alterations have been made:

The AUTO loading bug has been removed. Two other potential bugs have also been removed although the chances of them occurring were remote.

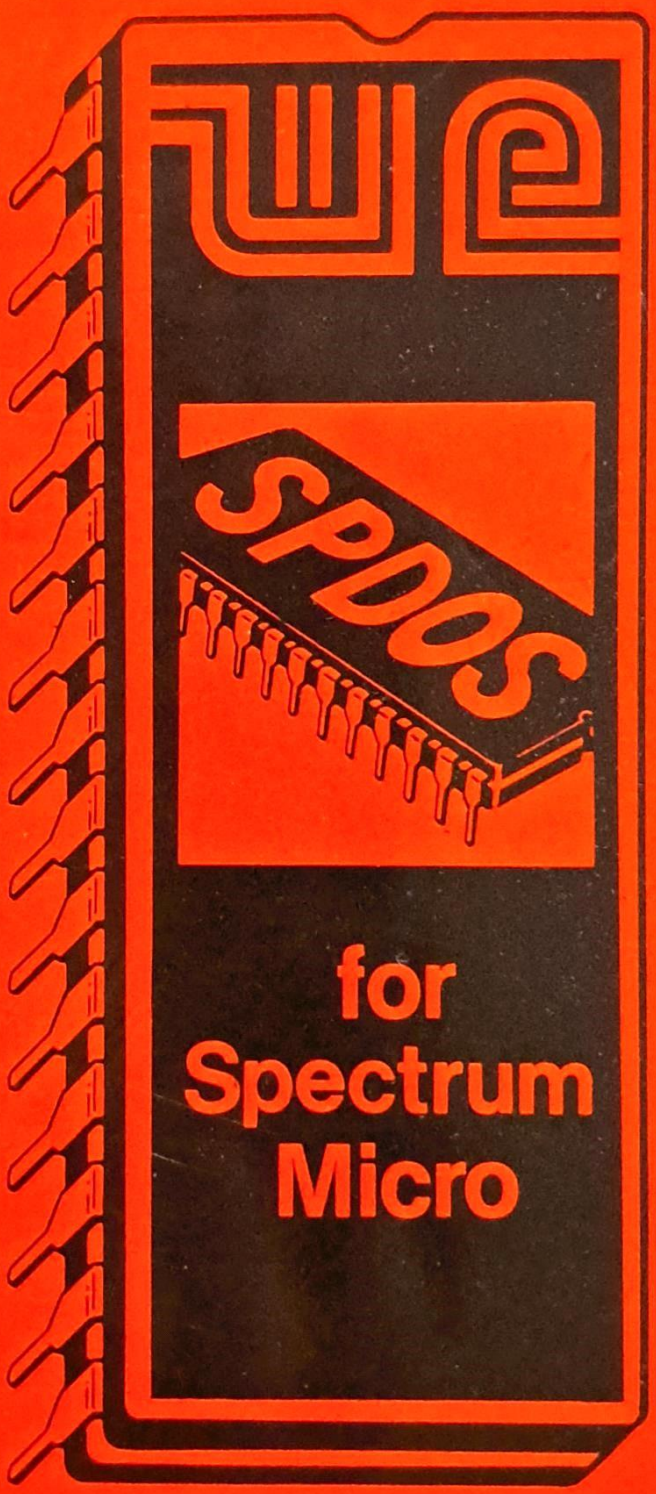
FORMAT has been improved and is now about twice as fast. This shculd cure any problems people were having with copysys.

The memory useage of SPDOS remains as before but it now uses 11 k-bytes of disc space. "minidos" now ends at 53077 decimal and has also been ammended to remove the bugs mentioned above.

There is an additional error message: i Verification failed. This is used by the new version of FORMAT.

N.B. The COPY command has been included to aid the transfer of programs with little or no protection it should not be used to infringe copyright and you must not expect too much of this facility. The full tape to disc program will deal with larger programs but both versions require the user to do some

The format af all commands will ensure that all programs written only in BASIC will work without alteration on both the Watford and the Kempston versions of the DOS.


SPDOS
for
Spectrum
Micro