CONTENTS

1 INTRODUCTION

The Floppyone Disc System was designed to fill the gap that has been left by Sinclair in the BULK storage aspect of personal computing. A valiant attempt was made to fill the gap using the microdrives, but unfortunately the speed and reliability of these devices has proved to be a greater disadvantage than the advantages of the low price.

This booklet will explain the operation of a relatively sophisticated, easy to use and above all reliable disc system.

Programming hints will be found at the appropriate sections of the booklet along with useful little programs to help both the experienced and the inexperienced user.

This system will be found to be most reliable when used with a modern disc drive. It is often false economy to purchase an old second hand drive from someone as you will not know the true condition of the drive and also servicing may be a problem if the drive should need alignment or replacement parts. Also as the system is a DOUBLE DENSITY system certain older drives, manufactured when only single density was in use may give unsurmountable reliability problems.

Before going any further one word of warning must be given:

```
**********************************
****** BACKUPS MUST BE MADE ******
**********************************
```

It is indeed false economy to think that backups merely cost unnecessary money and time. A system crash can destroy months of hard work in less than 1/2 a second. Or the cat could knock a cup of coffee over a disc you have been working on with disasterous results.

THERE IS NO SUBSTITUTE FOR A BACKUP

It is not always necessary to make your backups to disc, as you will see under the section on MOVING files

2 GETTING STARTED

2.1 CONNECTING UP THE SYSTEM

a. Make sure that ALL the power is turned off before starting.

b. Plug the DOS board into the back of the Spectrum or the Interface One if you have it.

c. Connect the disc drive cable to the DOS board with the arrow on the connector uppermost.

d. Making sure that there is no disc in the drive, turn on the power to the disc drive.

e. If the red light on the drive comes on continuously, then you probably have the cable to the interface the wrong way round. Repeat from step (a) but with the cable the other way round.

f. Now make sure that the switch to enable the DOS board is down, and if you have an Interface One then make sure the Interface One switch is also down. IF YOU HAVE AN INTERFACE ONE CONNECTED, THEN THIS SWITCH MUST BE DOWN AT ALL TIMES !!!

g. Turn on the Spectrum, the drive(s) you have connected will now start up one at a time and as the computer recognises each drive it will put it up on the screen. If no drives are connected then 'Drives = NIL' will be displayed and instead of the usual Sinclair copyright message you will get -   No Drive, 0:1

h. If the display shows the appropriate number of drives connected then all is well and you will want to continue to FORMAT your fist disc.

2.2 Formatting

Before you can save or load programs you have to format the disc that you wish to use. This is so that the computer will know where to put the information on the disc.

Before you can format the disc you must also know the specifications of your disc drive, for instance is it double sided and does it have 40 or 80 tracks.

Note that with this system it is possible to mix any combination of disc/drive types and that unlike some systems, double sided drives are treated as two separate drives.

The command to format a disc is :

```
   !FORMAT "discname";"password";number of tracks
```

where the discname and password must be strings of between 1 and 10 characters in length, and the number of tracks must be less than or equal to the number of tracks that the drive can handle.

In all of the commands any Sinclair Basic string form could be used where a string is required and similarly any numeric variable could be used when a number is required.

A typical example of this is :

```
  10 LET a$="Test":LET b$="bat":LET a=40
  20!FORMAT a$;b$;a
```

This will format a disc with the name 'Test', the password will be 'bat' and it will be formatted to a size of 40 tracks, which will give a usable capacity of 192 kilobytes.

The capacity of the disc can be calculated from the formula (5*(number of tracks))-8, so a disc formatted to 80 tracks will in fact give 392 kilobytes.

HINT- Many disc drives will allow you to format the disc to 2 or 3 tracks more than the nominal amount, giving 10 to 15 kilobytes more storage. This is because the manufacturers have had to allow for manufacturing tolerances in their parts so that the read/write head can actually move further along the disc than the nominal 40/80 tracks, giving typically 202 kb for 42 tracks or 402 kb for 82 tracks.

2.3 The password

The need to enter the password can be avoided if the password is made equal to, or starts with CHR$ 0, i.e. you can format a disc with the following command so that it is not necessary to enter a password :

```
   !FORMAT "Test";CHR$ 0;40
```

, the disc formatted using this command will not require a password to be input when in use.

Discs can not be formatted if they have already been formatted unless the password is known, as you will be asked for the password of the disc that you wish to format if it has one. A new disc obviously does not have a password, so there is no problem there, and if it has CHR$ 0 as a password you also won't be asked for a password.

For more details on the password see section 3.

2.4 PRODUCING A CATALOGUE OF THE DISC

There are a number of ways of producing a catalogue of the disc.

a. Let's first deal with the standard DOS method:-
!CAT will produce, on the screen, a directory of the disc. On the top line of the directory you will see the name of the disc, followed by the number of the current drive.

Below this will follow the type of file, name of the file and the extent (size) of the file in kilobytes.

After all the files (if any!) have been printed to the screen, the storage capacity remaining on the disc will be displayed, followed by the number of directory writes that have occurred since the disc was last formatted. This can in fact give an indication of the condition of the disc, as it gives a record of how many times the disc has been accessed for saving or changing an entry in the directory, which can give an indication of the amount of use the disc has had.

Following this is the number of tracks that the disc was formatted to.

Then there are the statuses of two system flags that you will find quite useful as you progress with the system.

b. Then there are a number of forms that are available to NON microdrive/interface one users. In other words if you have an Interface One connected these commands will NOT work!

CAT will do the same as !CAT

CAT drive number +1 will catalogue the drive number specified minus one AND change to that drive as the current drive. i.e. CAT 1 will catalogue drive 0 and change it to the current drive. The reason for the difference is that Sinclair chose to start his drive numbering with one, whereas the DOS starts with drive 0.

CAT #stream,drive number +1 will send the catalogue of the selected drive to the specified stream. For example

CAT #3,1 will send the directory of drive 0 to the printer.

NOTE that all of the CAT commands that specify a drive number will change the current drive number to that number.

2.5 Changing the current drive

The current drive can be changed by using the DOS command:

!d=drive number The 'd' can be upper or lower case.

2.6 SAVING

Normal saving from BASIC takes place in the same way for the DOS as it does for tape.

So all of the following will work:

```
  SAVE "name"                     to save a program
  SAVE "name" LINE number         to auto run the program
  SAVE "name"CODE start,length    to save bytes
  SAVE "name"SCREEN$              to save a screen
  SAVE "name"DATA a()             to save a numeric array
  SAVE "name"DATA a$()            to save a string array
```

In the last two, a and a$ are merely examples, any other array names could be used.

For NON Interface One users, the microdrive format can be used to save programs/bytes/data using the following format:

```
   SAVE *"m";1;"name"
```

to save a program called name to drive 0. Again note that the drive to which the program is saved, is the drive number specified minus one.

2.7 LOADING

Loading takes place in much the same way that saving takes place as far as the user is concerned, since again the load commands are identical to their tape counterparts.

The following are available:

```
  LOAD "name"                     to load a program
  LOAD "name"CODE                 to load bytes
  LOAD "name"SCREEN$              to load a screen
  LOAD "name"DATA a()             to load a numeric array
  LOAD "name"DATA a$()            to load a string array
```

In the last two, a and a$ are merely examples, any other array names could be used.

For NON Interface One users, the microdrive format can be used to load programs/bytes/data using the following format:

```
   LOAD *"m";1;"name"
```

to load a program called name from drive 0. Again note that the drive from which the program is loaded is the drive number specified minus one.

2.8 ERASING FILES

There are two main forms of this command:

a. Erasing a file directly by using its name.

```
  !ERASE "name"                     to erase a program
  !ERASE "name"CODE                 to erase bytes
  !ERASE "name"DATA                 to erase a numeric array
  !ERASE "name"DATA $               to erase a string array
```

b. Implied erasing.

You will already have seen in the directory listing the following line :- Erase flag = 0

This means that if you have a program on the disc called 'name' and you save another one also called 'name', you will have 2 programs on the disc called 'name'. However, if you had used the command !ERASE 1 , then the erase flag would have been set to 1, and saving the second program as 'name' would have AUTOMATICALLY erased the first one. This can be quite useful. !ERASE 0 will turn off the auto erase flag.

For NON Interface One users, the microdrive format can be used to erase programs/bytes/data using the following format:

```
  ERASE "m";1;"name"
```

To erase a program called name from drive 0. Again note that the drive from which the program is erased is the drive number specified minus one. Also note that in this case, unlike loading and saving, there is no need for the qualifiers CODE/DATA, as the first entry in the directory with the specified name, will be erased regardless of the type.

3 CHANGING NAMES

3.1 Changing names of files

Very often you will want to change the name of the program or code that you have saved to the disc, to make it shorter or more recognisable. Sometimes you might even want to change the name of a 'run' program to prevent the auto-run after a reset or power-up.

The basic form of the command is

```
   !"oldname" TO "newname"  .
```

The 'TO' is the keyword 'TO', symbol shift F.

This form of the command was adopted to minimise the amount of typing in command mode. However, it does cause a problem when trying to use strings to change the name. It is necessary to concatenate the nul string with the string desired. The following example will illustrate the point.

```
  10 INPUT "Type in the OLD name of  the  file  you  want  to
change >";LINE a$;"Now the NEW name >";LINE b$
  20 !""+a$ TO b$
  30 INPUT "Do you want to change any more ?";LINE a$:IF CODE
a$=CODE "y" THEN GO TO 10
  40 STOP
```

NOTE line 20- the empty quotes have the string variable added to them. Also note that no file descriptors are required at all.

3.2 Changing the name of the disc

You will not often have cause to change the name, but if you feel that the name does not adequately describe the content of the disc then you can change it to something more suitable using the command:

```
!n"newname"
```

3.3 Changing the PASSWORD

You may want to remove the password, or if someone finds out what it is you may want to change it to prevent unauthorised use of your programs or data.
The command for changing the password is:

```
!i"newpassw"
```

It is necessary to enter the old password before the command is executed, even if the password has already been entered, to prevent people from changing your password if you leave the computer un-attended for a short while after loading the program.
The password also has to be input when a disc is re-formatted.

`!iCHR$ 0` will effectively remove the password, preventing the system from asking you for a password. The password can easily be re-installed if necessary.

4 NON MASKABLE INTERRUPT

4.1 THE NMI

Operating the NMI button will cause the entire contents of the Spectrum's memory to be dumped to disc, and then allow the program to continue execution from the exact point at which the button was pressed.
If, after doing an NMI, you examine the directory, you will find that there is a BYTES file called 'nmi0' which can be loaded with the command LOAD "nmi0"CODE . If another NMI is made before the name of the file is changed from 'nmi0' then the new one will be called 'nmi1' until a maximum of 'nmi3'. After that the error report Nmi exists XX:Y will be given where XX is the line number and Y is the statement number that the program was busy executing when the NMI button was pressed.

4.2 PIRACY

It would be illegal in most countries to load a piece of software from tape and copy it to disc for your own use UNLESS you already owned the software OR you were checking the software for compatability with the disc system before purchasing the software. You should not have any problem with software houses objecting to you copying software, that you already own, to disc.

Piracy occurs when you borrow a friend's tape and copy it, whether to disc, tape or even microdrive.

```
        **************************************
        * REMEMBER:- PIRACY DEGRADES SOFTWARE *
        **************************************
```

NOTE: There will be certain programs where the NMI may not work correctly. You should always use a blank {freshly formatted} disc when trying to NMI a program for the first time in case the program causes the system to crash, which could possibly wipe out an entire disc. If the NMI is successful then you can do the NMI on the disc that you actually want it on. Backups should however not really be necessary as you should have the original on tape if it is a commercial program.

HINT:- Remember to switch the DOS on after loading the tape, otherwise the NMI will probably have the same effect as resetting the computer.

5 RUN-ON

5.1 Run-on time

If you have been trying out the drive so far you will have noticed that, with the exception of the NMI, the drive carries on running for a short while after the disc access command (loading, saving etc.) has been completed.

The reason for this is to speed up access to the disc by avoiding having to wait for the disc to startup if the disc has just been used, as the disc can take up to a second to start up if it has stopped. It has been found that disc accesses normally occur in bursts, with relatively long pauses between these bursts of activity.

This means that if a disc operation occurs then it is likely that another one will occur soon. Therefore it makes sense to keep the drive running for a short time after it has been accessed. This is called the RUN-ON time.

The run-on time is nominally set to 1,6 seconds. It can be adjusted to have values from one fiftieth of a second to as long as five seconds in steps of a fiftieth of a second.

```
    !t=n where 0 < n < 255 , so
    !t=200 will give a run-on time of 4 seconds
```

NOTE !t=0 or !t=255 will cause the drive to run-on forever.

5.2 RUN-ON on/off

If you load a program with a machine code part which disables interrupts when it is running, (typically a game !) then the run-on timer would never mature and the drive would run-on forever, or until the interrupts are re-enabled.

To see the effect of this, try the following little program.

```
  10 SAVE "runon test"LINE 20: STOP
  20 BEEP 10,0:GO TO 20
```

RUN the above program, NEW or reset the computer and then load the program. The drive will continue running for quite a long time (much longer than the normal run-on time) or until you press BREAK. This is because the BEEP command disables interrupts for its entire duration.

Now try the following:

```
   5 !ERASE "runon test"
  10 !r=1:SAVE "runon test"LINE 20: STOP
  20 BEEP 10,0:GO TO 20
```

RUN the above program, NEW or reset the computer and then load the program. Now the drive will home to track 0 (the parking track) BEFORE the program starts.

!r=0 will enable the run-on timer again.

Normally the run-on should only be disabled when saving the last segment of a multi-part program.

NOTE: Run-on is determined by a flag stored in the directory entry for each file, therefore you have to decide whether you want it on or off when SAVING the program, NOT when loading it. Changing the run-on flag before loading a program will have no effect whatsoever.

6 64 CHARACTER MODE

6.1 There are 4 different possibilities in the 64 character mode:

```
   !6=0 for normal 32 character mode.
   !6=1 for main screen in 64 character mode.
   !6=2 for lower screen in 64 character mode.
   !6=3 for both parts of the screen in 64 character mode.
```

NOTE that the lower screen has got problems in 64 character mode because of the way Sinclair handles the line editor.

6.2 The UDGs will always be printed full size as there is no way the software can decide what shape should replace the 8*8 matrix.

7 AUTO BOOT

If there is a program call "run" on the disc, it will be loaded when the computer is powered up, after a reset or after NEW.

This feature allows the system to be used by someone who does not even know how to LOAD a program as the entire operation can be menu driven.

The name 'run' must be in lower case.

8 PEEKING AND POKING THE DISC

8.1 Peeking the disc

Individual bytes on the disc can be read by using the following function:

```
LET a=!PEEK(sector,byte)
```

This will give byte from the specified sector on the disc in the variable 'a'. Any other variable could be used instead of 'a', PRINT !PEEK(10,35) will print the value of byte 35 of sector 10.

It is also possible to peek an entire string from the disc by using the function

```
LET a$=!PEEK (sector,byte),length
```

Which will give a string consisting of the bytes peeked from the disc starting at the byte specified by (sector,byte) and ending after the amount of bytes specified by the length have been loaded into the string.

Note that the sectors start from 0 on track 0 through to 5 times the number of tracks -1, i.e. a disc formatted to 40 tracks numbers from 0 to 199. The value 'byte' can vary from 0 to 1023 for the 1024 bytes on the sector. The maximum value of the length will depend on the amount of free memory available.

You will notice that the drive only runs for the first byte of a particular sector as that sector is loaded into the buffer. The drive will only run again when a byte from another sector is requested, so if you are peeking one disc and wish to compare the same sector on a different disc then you will have to peek some other sector to force the system to read the sector off the new disc that you have inserted.

8.2 Poking the disc

With the following command any byte on the disc can be altered at will:

```
!p sector,byte,value
```

For the range of the sector and byte values see 8.1 . The value poked to the disc can be anything from 0 to 255.

The numeric value to be poked can be replaced by a string or string variable so it is possible to poke an entire string to the disc with only one command.

```
!p23,12,"hello"
```

will store the string 'hello' on the disc starting at byte 12 of sector 23.

WARNING - caution must be exercised when using this command as writing will not take place immediately but only when reading or writing to another sector. What actually happens is that the sector that you want to poke (write to) is loaded into buffer ram and the byte that you have specified will be changed BUT the sector will not be written back to the disc yet because it is most probable that you will want to change another byte on the same sector, so writing it back to the disc at this stage would merely be a waste of time.

To force the system to write that sector back to the disc you must PEEK some other sector on the disc. Any other sector will do.

Also DO NOT CHANGE THE DISC WHILE THERE IS AN ACTIVE POKE!! as this can cause the sector from the previous disc to be inadvertantly copied to the new disc with disasterous consequences. (Only if you haven't made regular backups !)

9 MOVING FILES

The DOS allows files to be moved from the current disc to a specified disc or tape for backup purpose.

The command has the following forms:

a. !MOVE "filename" TO drive number

where drive number = 0 to 7 for the discs and 8 for moving files to tape.

e.g. !MOVE "name" TO 3 will move a file called 'name' from the current drive to drive 3

b. !MOVE file number TO drive number

Will move the file specified by its position in the directory to the specified drive.

e.g. !MOVE 3 TO 8 will move the third file in the directory to tape.

c. !MOVE 0 TO drive number

will move all the files on the current drive to the specified drive. This is useful for making backups.

e.g. !MOVE 0 TO 8 will copy all the files on the disc to tape.

NOTE NMI files have specifically been excluded from the move command as the system would be unable to handle the length of NMI files. The system actually checks the length of the file before moving it, not the name, to determine if it is an NMI.

10 ERROR TRAPPING

10.1 On Error Goto

This command allows you to trap errors such as input errors etc. or numeric range errors with the utmost simplicity.

The command has the form:

```
  !TO  line number
```

After this command the first error that occurs will cause the program to GO TO the specified line number.

```
  !TO  0
```

will turn off the on error goto.

10.2 On Error Gosub

```
 !^line number    ( the ^ is symbol shift H )
```

This command is essentially the same as the !TO command except that the error handling routine must end in a RETURN because effectively a GOSUB has been executed.

10.3 WHAT NUMBER ?

To find out what error has occurred one can use !THEN as a function :

```
    PRINT !THEN  will simply print out the error number -1
```

But this is obviously not much use as you could see the error number on the screen anyway. LET a=!THEN is much more useful as it allows the PROGRAM to find out what the error was.

10.4 Example

```
  5 DIM d(10)
 10 !TO 10: INPUT a: IF a<>0 THEN INPUT d(a): GO TO 10
 20 !TO 0
```

The above use of the error trapping function avoids tedious range checking during the input statements. Note that the on error goto is disabled (or changed!) so that other errors don't end up going back to line 10.

11 TRACK TO TRACK STEPPING SPEED

The track to track stepping speed of the DOS is controlled by the command - !s=speed where speed can take on the values 0 to 3 inclusive.

When a disc is formatted it is given the slowest possible stepping speed (!s=0) and you do not have to change it, but if your drive is capable of handling the higher stepping rates it can give a significant improvement in loading times if you increase the stepping speed to the maximum that your drive can handle.

```
!s=0  gives 30 milliseconds/step  This  information    is
!s=1  gives 20 milliseconds/step  stored on the  disc  so
!s=2  gives 12 milliseconds/step  this  command  must  be
!s=3  gives  6 milliseconds/step  used after FORMATTING.
```

The manufacturers data should be consulted to get the correct stepping rate.

12 SCREEN SAVER

If you have had the system running while reading through this booklet you will have noticed that the screen blanks out after 5 minutes of screen and keyboard inactivity.

Pressing any key or printing any character will restore the display.

The blanking can be disabled with the command

```
!y=1
```

and re-enabled with the command

```
!y=0
```

13 POWERUP COLOURS

The colours of the paper, ink and border can be set so that on power-up the Spectrum will power up with any colours that you would like to specify.

!u=paper,ink,border will set the desired colours in the DOS system area of the directory so that when the computer is reset, powered up or NEWed the DOS will look up the colours if there is a disc in the lowest numbered drive and replace the standard paper=7,ink=0,border=7 colours of the Spectrum with the colours that you had specified using the !u=p,i,b command.

Note that your specified colours can only be used if the system initialises with a disc in the drive.

14 INTERRUPT ROUTINES

14.1 The Dos can be made to call a user's or a ROM routine 50 times a second (every time an interrupt occurs) by executing the following command: !w=address

where address is the start address of the machine code routine. The byte before the start address MUST be a machine code RET instruction, 201 in decimal notation for the !w command to operate. This a a safegaurd to prevent accidently jumping into undesired places in memory.

!w=0 will turn off the interrupt calls.

14.2 Restrictions

To execute properly, the routine should complete execution within a 50th of a second, otherwise it will obviously not be called 50 times a second but at some slower rate.

The registers that you do not have to preserve (because the DOS preserves them for you) are HL, DE, BC and AF. You must save all the other registers that you use.

14.3 TYPICAL USES

Typical uses are the running of background routines or setting up keyboard/screen/printer buffers to give real spooling capabilities.

Scrolling windows could be set up and the scrolling would then take place while other BASIC routines were running.

15 MICRODRIVE COMMANDS

15.1 Availability

The microdrive commands for loading, saving and erasing program/files are only available if the interface one is NOT connected and if they are used from BASIC. It is quite improbable that these commands will be usable from machine code, unless the machine code uses the basic interpreter to execute them.

15.2 The commands

See section 2 for loading, saving, erasing and CAT operations.

15.3 The use of the microdrive commands

The reason these commands with their rather long-winded syntax, have been included in the command repetoire is that there are many programs (especially the more serious ones) which are available on tape that have a microdrive option. In many cases the tape option is written in machine code with many protection idiosynchroses which make use of the DOS difficult, but the microdrive operations are relatively straightforward and written in BASIC, allowing the program to run on the DOS with NO modification.

15.4 Interface One variables

Some of the programs designed for the microdrive poke the interface one variables area. If these variables are not present then the BASIC program could be corrupted,
so the command !e is provided to insert the interface one's system variables. Normally one would use it to immediately after power up or initilisation, before loading the main program.

There is no command to remove the interface one system variables, normally the easiest way is to reset the computer.

16 PRINTER INTERFACE

16.1 LLIST

The command LLIST will send a listing to the printer port, with full de-tokenisation taking place. The UDGs and graphic characters will be replaced by '?' whenever they occur.

16.2 MARGIN

If the listing is too wide then set the margin using the following command:

```
  !m=width  where the width can vary from 0 to 255.
```

With a width of 0, there is no right margin and only explicit carriage return/linefeed codes will be sent to the printer. With all other permissable values a CR/lf combination will be sent after the number of characters specified by width have been sent to the printer. The DOS initialises with the margin = 0.

16.3 LPRINT

Using the LPRINT command, characters can be sent to the printer for printing as with any other printer interface. Again graphics codes and UDGs are replaced by question marks.

16.4 Control Characters

To send control characters to the printer there are 2 avenues open to you:

a. Precede every character that has a code less than 32 or greater than 127 by CHR$ 27; . This even applies to CHR$ 27.

b. TRANSPARENT MODE

The interface can be switched to transparent mode by using the command !q=1 . When in transparent mode none of the characters sent to the printer is translated/absorbed in any way. It is a good idea to use transparent mode with TASWORD_2. This mode can be turned off using the command

```
  !q=0 which will restore full de-tokenisation etc.
```

16.5 ABORT

If at any stage you wish to abort printing, but do not want to BREAK into the program, just hold down the SPACE key on the keyboard. This will cause the printer routine to just throw away all the characters that are sent to the printer.

Break can be pressed at any time while printing, it will cause a normal BREAK error report to occur.

16.6 COPY

When the copy command is executed a search will be made on the current disc for a CODE file called 'UTC' and this file will be loaded to the hex address 3880. The first byte of the file should be a 3 to identify it and the machine code proper should start execution from 3881 hex. The length of the block of code should not exceed 0380 hex. If it is necessary to call Sinclair ROM routines, it can be done by doing a RST 0010 followed by the 16 bit address of the routine that you wish to call, low byte first. The ROM will be called with all the registers intact, even the flags register, and when a return is made to your routine, only the registers that have been changed by the ROM routine will have been altered. Thus the 'CALL' to the ROM is totally transparent. For instance if you wanted to print a character using the ROM routine at address 0010 normally done by

```
       LD A,02    ; select main screen
       CALL 1601  ; the chans  subroutine in the ROM
       LD A,CHAR  ; the character that you want to print
       RST 10     ; print the character
```

will have to be replaced by

```
       LD A,02    ; select main screen
       RST 10     ; call ROM routine, chans.
       DEFB  01   ; low address of chans
       DEFB  16   ; high address of chans
       LD A,CHAR  ; the character to be printed
       RST 10     ; calls the ROM routine
       DEFB  10   ; low byte of rom routine
       DEFB  00   ; high byte of rom routine
```

The machine code routine should end in an ordinary RET statement as the stack has been set up to return correctly to continue execution of the BASIC program.

A routine to handle certain types of printer can be supplied.

16.7 TASWORD TWO

The interface control codes for Tasword 2 are
0, 0, 0, 2548.

```
Carriage return = 13
Linefeed        = 10
Margin          = to taste....
```

Transparent mode should be selected by editing line 20 to include the command !q=1 .

When all of these changes have been made your customised version of TASWORD can be saved onto disc.

16.8 USING +80 SOFTWARE AND TASPRINT

The software that is shown here can be used for many of the other programs that require specialised printer drivers.

a. INITIALISE THE INTERFACE FOR TRANSPARENT MODE

```
     CD 80 02      CALL 0280    ;page the DOS in
     3E 10         LD A,10      ;set bit 4 of A
     32 E4 3B      LD (3BE4),A  ;set bit 4 of PRINTFLAGS
                                ;for transparent mode
     C3 16 00      JP 0016      ;RET via DOS page out address
```

This routine is 11 bytes long.

b. CHECK FOR PRINTER BUSY

```
   CD 80 02      CALL 0280h     ;page DOS in
   3A 04 20      LD A,(2004h)   ;get BUSY line
   2F            CPL            ;invert line
   CB 6F         BIT 5,A        ;Z = not busy, NZ = busy
   C3 16 00      JP 0016h       ;RET via DOS exit
```

This routine is 12 bytes long

c. OUTPUT A CHARACTER

```
   FD 46 01      LD B,(IY +01)  ;preserve FLAGS in BC
   C5            PUSH BC        ;save BC
   FD CB 01 CE   SET 1,(IY +01) ;select PRINTER
   CD F4 09      CALL 09F4      ;call PRINT routine
   C1            POP BC         ;Get BC back
   FD 70 01      LD (IY +01),B  ;restore FLAGS
   C9            RET            ;return to calling routine
```

This routine is 16 bytes long

The above code should be poked into the or installed into the program in the way the supplier recommends. Normally one would select the interface type 'OTHER' when asked by the customisation program

17. ACCESSING THE SYSTEM FROM MACHINE CODE

17.1 The DOS can be paged in from machine code by a call to 0280h and paged out by calling 0016h.

17.2 The System variables are listed below. They can be accessed from machine code by FIRST paging in the DOS and then reading the appropriate address. The DOS should always be paged out before RETurning to BASIC.

NAME	L	ADDR	DESCRIPTION
TEMP_1	2	3000	Tempory register storage while calling SINCLAIR ROM routines.
TEMP_2	2	3002	
TIMER	1	3004	The RUN-ON timer, is 00 when disc is off and FF when disc is busy being 'HOMED' to track 0.
PORT	1	3005	Copy of the OUTPUT port.
CER	1	3006	Counts the number of retries made during LOADing.
VF	1	3007	The 'VERIFY' flag, bit 0 set to indicate that a block is being verified and equals FF if there was a verify error.
SECNO	1	3008	Sector number

ERRL	2	3009	Contains the line to GOTO if an error occurs or if bit 7 of 300A is set then an GOSUB takes place instead of a GOTO. After 'on error' action has occurred 3009 contains the error number and 300A contains FFh.
PFL1	1	300B	Bit 0 = 0 -- Main screen = 32 col Bit 0 = 1 -- Main screen = 64 col Bit 1 = 0 -- Lower screen = 32 col Bit 1 = 1 -- Lower screen = 64 col Bit 5 = set if AT detected as print item Bit 6 = set to indicate that 2 parameters follow. Bit 7 = set to indicate that 1 parameter is still expected.
PFL2	1	300C	AA or 55 depending on which half-cell the character is to be printed.
PFL3	1	300D	As above but for lower screen
DS	2	300E	Address to which a sector is loaded
BT	2	3010	Number of bytes loaded out of a sector
CR	2	3012	The bit set = the current drive number e.g. bit 2 set means that drive 2 is the current drive.

18 DISC INTERFACE CONNECTIONS

18.1 All odd-numbered connections (underside of the connector on the drive are connected to ground {0v})

18.2 The locating slot on the drive's connector is between connecotr blades 2 and 4

18.3 Layout of connector:

gnd	1	+	2	--
gnd	3	+	4	--
gnd	5	+	6	sel 3
gnd	7	+	8	index
gnd	9	+	10	sel 0
gnd	11	+	12	sel 1
gnd	13	+	14	sel 2
gnd	15	+	16	motor on
gnd	17	+	18	direction

gnd	19	+	20	step
gnd	21	+	22	write data
gnd	23	+	24	write gate
gnd	25	+	26	track 0
gnd	27	+	28	write protect
gnd	29	+	30	read data
gnd	31	+	32	side select
gnd	33	+	34	---------------

19. LOADING FROM MACHINE CODE

```
---------------------------------
         ORG A000             ;set the origin to any desired
                              ;point in memory
START    LD IX,HEADER_2       ;point to where header must go
         LD DE,0011           ;length of all headers
         LD A,00              ;specify 'header'
         SCF                  ;
         CALL 0556            ;the specified header (if !f=1)  or
                              ;the next header (if !f=0) will now
                              ;be loaded. If !f=0 is in use (this
                              ;is the default) then the user must
                              ;provide his  own  compare  routine
                              ;and loop back to 'START'  until  a
                              ;match is found.
                              ;
BODY     LD IX,DESTINATION    ;can be specified or  derived  from
                              ;loaded header
         LD DE,LENGTH         ;can be specified or  derived  from
                              ;loaded header
         LD A,FF              ;load a block of code
         SCF                  ;
         CALL 0556            ;call load subroutine
                              ;
         RET                  ;return to calling program
                              ;
                              ;
                              ;
HEADER_1 00                   ;these are two  consecutive  header
         "NAMEOFPROG"         ;areas, the first area contains the
         00 00                ;name of the file  wanted  and  the
         00 00                ;second will contain  the  name  of
         00 00                ;the file marked for loading by the
                              ;subroutine  'BODY'.  If  the   DOS
HEADER_2 00                   ;search flag has been  set  by  the
         "nameofprog"         ;DOS command !f=1 (it only needs to
         00 00                ;be done once after a disc has been
         00 00                ;formatted) then the file name  and
         00 00                ;type will be found by the  DOS  or
                              ;if it  is  not  in  the  specified
                              ;directory the a 'file  not  found'
                              ;error will be given.
                              ;
                              ;Note that the header is made up in
                              ;the same way that tape headers are
                              ;and  the  bytes  have   the   same
                              ;meanings as for tape.
                              ;e.g. byte 0 in the header  defines
                              ;the type of file, 0=program
                              ;                  1=data
                              ;                  2=data string
                              ;                  3=bytes
```

         SAVING FROM MACHINE CODE

```
SAVE     LD IX,HEADER_1       ;The name  of  the  program  to  be
                              ;saved should be  provided  in  the
                              ;header area. The header should  be
                              ;set up exactly as for tape use.
                              ;
         LD DE,11             ;The length of the header
         LD A,0               ;Specify that  a  header  is  being
                              ;saved
         PUSH IX              ;save the pointer
         CALL 04C2            ;call the save subroutine
                              ;
         POP IX               ;restore pointer to header
         LD E,(IX 0B)         ;get length from header
         LD D,(IX 0C)         ;
         LD C,(IX 0D)         ;get start address of block
         LD B,(IX 0E)         ;
         PUSH BC              ;copy start of block to IX
         POP IX               ;
                              ;
         CALL 04C2            ;call save subroutine
                              ;
         RET                  ;return to calling program
                              ;
                              ;
```

Typically any program which uses the STANDARD Sinclair header system and calls the standard Sinclair entry points (04C2h for save and 0556h for load) will work with the DOS.

21. HEADER DECODER

Also to examine the directory entries virtually any 'header decoder' can be used. An example of one is given here.

```
  10 CLEAR 32511
  20 FOR a=32512 TO 32521: READ b: POKE a,b: NEXT a
  30 DATA 175,55,221,33,16+17,127,205,86,5,201
  40 LET b=32528+17: DEF FN a(x)=PEEK (b+x)+256*PEEK (b+x+1)
  45 POKE b-16,255
  50 RANDOMIZE USR 32512
  60 LET c=PEEK b
  70 IF c>3 THEN GO TO 50
  80 PRINT "Filename: ";
  90 FOR a=b+1 TO b+10: PRINT CHR$ PEEK a;: NEXT a
 100 PRINT : PRINT TAB 4;"Type: ";
 110 GO SUB 1000+100*c
 120 PRINT : PRINT
 125 POKE b,255
 126 PAUSE 0
 130 GO TO 50
1000 PRINT "Program"
1010 PRINT "Total length: ";FN a(11);" bytes"
1020 PRINT "Program length: ";FN a(15);" bytes"
1030 IF FN a(13)>9999 THEN PRINT "Load only": RETURN
1040 PRINT "Runs from line  ";FN a(13)
1050 RETURN
1100 PRINT "number array"
1110 LET a$="": GO TO 1220
1200 PRINT "character array"
1210 LET a$="$"
1220 PRINT "Array length: ";FN a(11);" bytes"
1230 LET d=PEEK (b+14)
1240 PRINT "Original array name: ";CHR$ (64+32*(d/32-INT
(d/32)));a$
1250 RETURN
1300 IF FN a(11)=6912 AND FN a(13)=16384 THEN PRINT "screen
image": RETURN
1310 PRINT "bytes"
1320 PRINT "Start address: ";FN a(13)
1330 PRINT "Length: ";FN a(11);" bytes"
1340 RETURN
```

21. ALTERNATE DIRECTORIES

If directory space becomes a limitation, more than one directory can be created.

This can only normally be done on a disc that has just been formatted with nothing SAVEd on it.

The procedure is as follows:

1) Reset computer
2) Format disc
3) Save block of code with length = 1024 * number of extra directories needed

e.g. SAVE "dirfile"CODE 40000,4096
will give enough space for 4 extra directories.

4) !p5,5,<directory desired>:IF !PEEK(6,0) THEN

```
e.g. !p5,5,2: IF !PEEK(6,0) THEN
```

will give the third directory

Note: The current directory selection will be stored on disc so that to access the base directory you will have to give the command:

```
      !p5,5,0:IF !PEEK(6,0) THEN
```

to get back to the base directory.

22. CONNECTOR LAYOUT

```
  ............................................................
  .                                                          .
  .                                         O                .
  .                                       NMI BUTTON         .
26. :: P                                              D      .
  . :: R                                              I ::   . 1
  . :: I                                              S ::   .
  . :: N                                              C ::   .
  . :: T                                              D ::   .
  . :: E                                                ::   .
  . :: R                                              D ::   .
1 . ::                                                R ::   .
  ......                                              I ::   .
       .                                              V ::   .
       .   EXPANSION CONNECTOR                        E ::   .
       .   ==============================  ====       S ::   .34
       .......................................................
```

This is a rough illustration of the connections to the DOS as seen from the back. Note that it is not to scale.
On most of the drives and printers the marking stripe on the ribbon cable will go to the pin 1 side of the connector

23. GREY-SCALE SCREEN DUMP

Just as the normal screen dump routine is contained in a file called UTC on the disc, a grey-scale dump routine is presented as a UTG file. This routine can only be effectively used with printers that use 8 needle printing.
To allow maximum flexibility, the printer control bytes can be edited using the following instructions. An example will be given for the MANNESMAN TALLY MT140 as an illustration.

To customise UTG:

```
1 CLEAR 32767
2 LOAD "UTG"CODE 32768
```

To set up the size of the linefeed so that the graphics do not overlap or leave big gaps between lines, the linfeed should be adjusted to 8/72 of an inch (normal needle spacing is 1/72 of an inch). If thin white lines appear in the screen copy then change this to 7/72 of an inch.

	MT140
3 POKE 32771,number of bytes to initialise printer to graphics line feed size. The maximum number that can be sent is 5 and the minimum is 0. The printer manual should be consulted about this.	POKE 32771,0
4 POKE 32772 to 32776 with the bytes to set the printer to graphics line feed size	NONE USED FOR THE MT140
5 POKE 32777,1 for carriage return only 2 for carriage return + linefeed	POKE 32777,2
6 POKE 32780,number of bytes to send 704 bytes to printer max = 8	POKE 32780,5
7 POKE 32781 to 32788 with the bytes that will put the printer into grapics mode for 704 bytes (double density)	POKE 32781,27 POKE 32782,76 POKE 32783,192 POKE 32784,2 POKE 32785,0
8 POKE 32789,number of bytes to POKE 32789,0 restore the printer to normal the MT140 will printing max = 5 automatically go back to the 9 POKE 32790 to 32794 with the normal line bytes to restore printer to normal spacing mode	
10 POKE 32795,31 if bit 0 = bottom needle on printhead 23 if bit 0 = top needle on printhead	POKE 32795,31
11 If you would like to have this program as your normal copy program all you have to do is POKE 32768,3	

12 SAVE "UTG"CODE 32768,256 (if 23768 was poked With '3' then SAVE "UTC"CODE 32768,256

13 To make a greyscale screen copy, type COPY g<ENTER> or if saved as UTC then just type COPY<enter>