# F L O P P Y     I N T E R F A C E

# F O R

# Z  X     S P E C T R U M     4 8  K

# O P E R A T I N G     F I R M W A R E

F.I.Z.  Version 3/540.1

************************************************

## U S E R     M A N U A L

(c)    19

83 Macronics Systems Limited

## C O N T E N T S

## FOREWORD - IMPORTANT NOTES

You have made a wise investment in your purchase and it is our wish that you get maximum benefit and long service from your disc drive system.

The following points are worthy of note and regular attention to them will preserve your investment.

<u>Care of the drive unit and interface</u>

1. Use only in a dust-free environment.
2. Move the drive unit as little as possible and when you do, always use the 'dummy' disc card with the drive door closed.
3. Keep the drive unit away from strong magnetic fields.
4. Avoid liquid spillage.
5. Always ensure the drive is switched off before connecting/ disconnecting to/from the mains power supply.
6. Never disconnect the interface from the computer whilst the power is still applied.

<u>Care of diskettes</u>

7. Never boot your disc system (start-up/power-up) with a pre-recorded diskette in the drive unit. Failure to observe this may result in the loss of data on the diskette.
8. Always replace a diskette in it's sleeve when not in use and don't forget to keep at least one back-up copy of all important files. Use the write protect facility as added security - and a copy kept on cassette tape will give you ultimate security and hence complete confidence in your ability to re-create your files should this ever be necessary.

9. Keep recorded diskettes away from strong magnetic fields.
10. Avoid contact of any description with the recording surface of the diskette.
11. Last but by no means least - always use good quality diskettes for maximum reliability - a cheap) diskette could result in permanent loss of priceless data.

## GETTING STARTED WITH YOUR DISC DRIVE SYSTEM

As with most new purchases we expect you will want to get your disc system running as soon as possible, like right now; This is the very reason why 'getting started' comes very early on in this manual. You are well advised to <u>READ</u> the <u>WHOLE</u> of this manual <u>before</u> attempting to use the disc system - it will save you a lot of time in the longer term; you will quickly get to know what can be done and where to find the detail.

It is recognised that there are those who will automatically ignore all advice - therefore, 'getting started' continues from here in Appendix D.

For those who show more patience - continue reading until you get to Appendix D and it will all be simple from there on.

## OVERVIEW

The disc unit used in the system is a Shughart SA200, 5¼", Slimline, Single Sided, Single Density, Soft-sectored drive.

Diskette storage is organised as 39 data tracks, each of 2816 bytes, plus one directory track.

```
     Track 0                      - Directory Track
     Track 1                      - First data track      (2816 bytes)
      ,,                             ,,
      ,,                             ,,
     Track 39                     - Last data track       (2816 bytes)
```

The directory track holds the diskette name and a directory entry for each data track which contains a Track Code to indicate a full, empty or bad track, and a filename plus a file type code.

The maximum number of different programs or data files per diskette is 39 if none were more than 2816 bytes long. There is a storage capacity of more than 100K bytes. Program or data files should be just less than a multiple of 2816 bytes to make optimum use of diskette space.

A number of reserved variables are needed for the system to operate, these are: -

```
            F$    RES   ND    DIR
            BS    BL    CS    CL
            AS    AL    ZAP   BACK
```

these variable names should not appear in a program in any other context.
On 'start-up' (System Boot) the REStart routine automatically loads the operating firmware into the top area of RAM, reserves space for buffers and loads the above Reserved Variables into RAM. 8K at the top of RAM is reserved for the system.
The command routines have mnemonic variable names purely for operator convenience, they are:-

```
     BS          BL     CS    CL
     ND          DIR     AS   AL
     ZAP    BACK
```

## FILE - DATA & PROGRAM

## Systems Variable F$ - 'Filename'

F$ is the variable reserved for identification of the current file (data or program - note that both are treated the same and are referred to in this manual as files). F$ Is a variable length string, initialised to zero length (i.e. NULL).

F$ holds the name of the file to be accessed when a command is executed. The contents of F$ must be enclosed in quotes.
The format is:-

```
                   (LET) F$  = "filename"
             or 10 (LET)) F$  = "filename"
```

where ( ) denotes Sinclair Function Key.

A filename consists of 1 to 6 None-space characters, any alphabetic or numeric character can be used. DO NOT use special characters since some will be used as logical seperators when handling different types of files, and others will be used for special functions.

### File Type Codes

There are three different types of file:-

| Type of file | mnemonic used (lower case is also acceptable) |
|---|---|
| Array files (string and numeric) | A |
| Basic files (programs) | B |
| Binary Code files | C |

The use of these different types of files and the assigned mnemonic is explained in the following section titled 'Command Routines'.

## COMMAND ROUTINES

All command routines, with the exception of BACK, may either be actioned from the Keyboard or used in program statements, (single or multi-line statements).

The BACK command cannot be used in program statements.

Commands take the form:-

```
      (LET)   E   =  (USR)    command mnemonic  (through the Keyboard)


or 40   (LET)   E   =  (USR)  command mnemonic  (as a program statement)
```

When control returns to BASIC the variable E will contain an Error Code (see Appendix A).
A value of zero indicates that the command was successfully executed, and a program should always check for this condition before proceeding further.
With some commands entered from the Keyboard it is convenient to use the form:

```
     (PRINT)        (USR)    command mnemonic
```

This will print the error code on the screen for immediate inspection following execution of the command but this could cause some confusion if used with DIR since the error code will be printed immediately following any other data.

```
ND  -  New  Diskette  -  Formats  a  diskette  ready  for  use.
(mnemonic)  (definition)               (function)
```

Used in the form:-

```
(LET)  F$  =  "Diskette Name"  (6 characters maximum)
then     (PRINT) (USR)  ND      ( or (LET)  E  =  (USR)  ND )
```

This command is normally used when a New diskette is to be brought into use, although it can be used to erase a previously recorded diskette if it's contents are no longer required.

WARNING: The 'new disk' process DESTROYS all previously recorded informed on the diskette and once executed, data cannot be recovered.

```
DIR        -     Directory     -     displays the contents of the
(mnemonic)     (definition)         directory on the screen.   (Function)
```

Used in the form:

```
     (LET)  E  =  (USR) DIR       (or   (PRINT)  (USR)  DIR)
```

This command will display the diskette name and number of remaining (empty) tracks  (2816 bytes each) on the top line of the screen, followed by a list of the files held on the diskette:-

```
Example:    FIRST1                  27
            (Diskette name)   (no. of empty tracks)

(Type of file)    (Name of file)    (No.  of tracks used by file)

   BASIC          BASPRG                 02
   BASIC          BASIC1                 01
   CODE           CODEIT                 03
   ARRAY          MAIL20                 02
   ARRAY          MAIL22                 04
```

Note    Don't forget that if (PRINT) (USR)  DIR   is used then the error code will always be the last character printed on the screen.

**BS - Basic Save -** Saves a BASIC program on diskette

Used in the form:-
```
      (LET)  F$  =  "filename"
      (PRINT)          (USR)

      BS
```

This command will save a BASIC program, together with it's variables, which is currently in the Spectrum's memory - (having been loaded from tape, microdrive, floppy diskette, or typed in from the Keyboard).

WARNING: Check that F$ holds a unique filename in the proper format before executing the BS command.

If the filename is not unique then an error code will result and the program will not be saved but the program will be retained in memory. User Defined Graphics are automatically saved and loaded with every BASIC program. They do not have to be loaded from a separate code file as would be the case from tape.

LOAD, with automatic RUN is produced as for cassette tape by extending the contents of F$ as follows:-

```
     (LET) F$   =   "filename,  line number"
```

Filename can be up to 6 characters; line number is the program statement line number that it will start execution from as soon as it is loaded. Use the BS command after confirming the contents of F$.

Example:
```
             (LET)    F$   =   "GOBBLE,  40"  (field seperator is mandatory)
             (PRINT)  (USR)  BS
```

Note: An existing file of the same "filename" will not be overwritten but will report an error code 9. The same "filename" can be used for 3 different types of file, i.e. A,B and C. See File Type Codes, page 6.

**BL - BASIC LOAD -** Loads a BASIC program from the diskette into memory.

Used in the form:-
```
           (LET) F$ = "filename"          NB. You must know the filename
                                              - use DIR to find  it.
then       (PRINT)  (USR)   BL            (Note - lower case characters are
                                            perfectly acceptable).
```

This command will load a previously saved program from diskette into memory.

For a program that did not have the auto-run feature when saved then GO TO.... or RUN will be necessary to get the program working.

NB. The use of RUN or NEW will clear the reserved variables which were previously set up for disc operation and these will need to be rebooted if you want to continue using the disc system. This can be done by using PRINT USR 64000 but remember that F$ will have been initialised to NULL character.
You cannot reboot if the variable ***F$*** exists - use CLEAR first.

User Defined Graphics are automatically saved and loaded with every BASIC program. They don't have to be loaded from a separate code file as would be the case when using cassette tape.

However they do occupy the variable U$ during the save/load process and this is the reason for U$ being reserved.

**CS - Code Save** - Saves binary code on diskette.

Used in the form:-

```
(LET)   F$ = "filename,  start address,  length,  auto-run start address"
                           (decimal)    (decimal)    (decimal)
                          (mandatory)         (optional)
then   (PRINT)  (USR)    CS
```

This command will save a binary file on diskette and allow automatic execution following a load, the auto-run start address for automatic execution of the code following a load is optional.

Field seperator s (Commas) are mandatory

<u>Example</u>:-

```
(LET)   Y$   =   "Auto,  50000, 64, 50010": (PRINT) (USR)  CS
```

Where Auto is the name of the file, 50000 is the start address, 64 is the byte length of the file to be saved, and 50010 the commence execution address.

Note that the field seperators are mandatory and must be present at all times.

**CL - Code load** - Loads a binary code file from diskette into memory.

Used in the form:-

```
(LET)   F$ = "filename,  start address,  length,  auto-run address"
                          (mandatory)            (optional)
(Field seperators are mandatory when an option is present)

then      (PRINT)  (USR)   CL
```

This command will load a previously saved binary file from diskette into memory and allows automatic execution of the code, if the option is specified, e.g.    LET    F$  =  "AUTO,  50000,  64,  50010,".
Note it may be that the code was previously saved with auto-run, therefore, the option need not be used with the CL command, e.g.   LET  F$ = "AUTO"
It may be necessary to stop the automatic execution of code after loading.
This can be done by specifying F$ as follows:-

```
(LET) F$   =   "AUTO,50000,64,0"    where 0 specifies no execution
or
(LET) F$   =   "AUTO,,,0"    Note that seperators are mandatory in this case
```

**AS - Array Save** - Saves String and Numeric array files on diskette.

Used in  the form:-

```
(LET)   F$  =  "filename, nn,  list of numeric and/or string array names"

(nn is the member nc.  of the set ) (array names to be seperated by commas)
(of array lists can can range from) (e.g. A$,B$,n,g,a                     )
(0-99                             )

then    (PRINT)   (USR)   AS
```

Numeric and string arrays share the same common file which is treated as a set of up to 39 members  (a whole diskette).
Each member of the set is a list of arrays occupying one track of the diskette.
Arrays must have been set-up in RAM and contents filled before saving.

Chapter 24 of the Sinclair Manual identifies the way arrays are handled -please read this chapter giving particular attention to the OVERHEAD CARRIED FOR EACH TYPE OF ARRAY.

Should the total length of arrays together with their overhead exceed 2816 bytes then error code 15 will be reported. (see Appendix A)

All the rules in chapter 24 of your manual are applicable so don't assume any difference when handling arrays within your disc system, e.g. be careful when re-dimensioning arrays - if F$ contains the same filename and member no. of a previously saved array, then the command AS will cause the saved array to be overwritten.

**AL - Array Load** - Loads Array file from disk into memory

The form used is exactly the same as the AS command, the "nn" descriptor will identify which member of the set of array lists is to be loaded.

```
i.e.       (LET) F$  =  "ARRAY,  12"

then   (PRINT) (USR)  AL
```

N.B. No array list is specified for AL - the arrays loaded have exactly the same names, types, dimension and contents as the ones previously saved, and your BASIC program should be written to accommodate this.

---------------------------------------------------------------------14

**ZAP - Delete File** - deletes file names from directory

Used in the form:

```
(LET) F$  =  "filename . file  type  code"

then       (PRINT) (USR) ZAP             (Note: File type code is mandatory)
                                          (and full stop used as seperator  )
```

```
File type codes:  B = BASIC file  (lower or upper case is acceptable)
                  C = CODE file (binary)
                  A = ARRAY file
```

Example:-

```
(LET) F$ =  "DELLO.  B'

then     (PRINT) (USR)  ZAP
```

This command is the only way of deleting a program or data file,

WARNING: Use with caution - there is no second chance to stop the process. Keep copies of important programs or data on separate diskettes or under different filenames.

MAKE SURE that F$ contains the name and type of the file you wish to kill before using this command.

---------------------------------------------------------------------15

**BACK - Diskette Copy** - Copies the total contents of one diskette onto another.

Used in the form:-

PRINT (USR) BACK

This command enables a complete diskette "Back-up" copy to be made.
The routine ignores F$ and copies the complete contents in 4 stages, each stage is prompted for diskette change-over. The diskette that you copy from (Read) is called the 'Source' and the diskette that you copy onto (Write) is called 'Destin' (short for Destination). Before you start to use the BACK Command make sure you 'write protect' your source diskette. The routine will prompt for 'SOURCE ' then 'DESTIN'; now insert the appropriate diskette (close the door) and type ENTER. Repeat this a further three times and you will find on the last occasion that the routine destroys the contents of RAM and it is necessary to re-Boot the system:-

```
  Type:  (PRINT) (USR) 64000 (ENTER)
```

Note: You can use 'BACK' to copy onto a new diskette without having to format (ND) the diskette first.

## APPENDIX A

ERROR CODES

| E | MEANING |
|---|---|
| 0 | No error |
| 1 | System already initialised or F# already exists |
| 2 | System not initialised |
| 3 | Bad Disc (cannot write Directory) |
| 4 | Write protected (notch covered) |
| 5 | Write error (read after write verify fails repeatedly) |
| 6 | Hard Read Errors (track may be unusable) |
| 7 | Bad file specification (wrong format) |
| 8 | No room on disc (full or not enough capacity available |
| 9 | File already exists on diskette |
| 10 | File not found or Mandatory item missing |
| 11 | No such Member of set of arrays |
| 12 | Parameter missing |
| 13 | Member No. 99 |
| 14 | Can't find Array |
| 15 | Array list too long |

## APPENDIX B

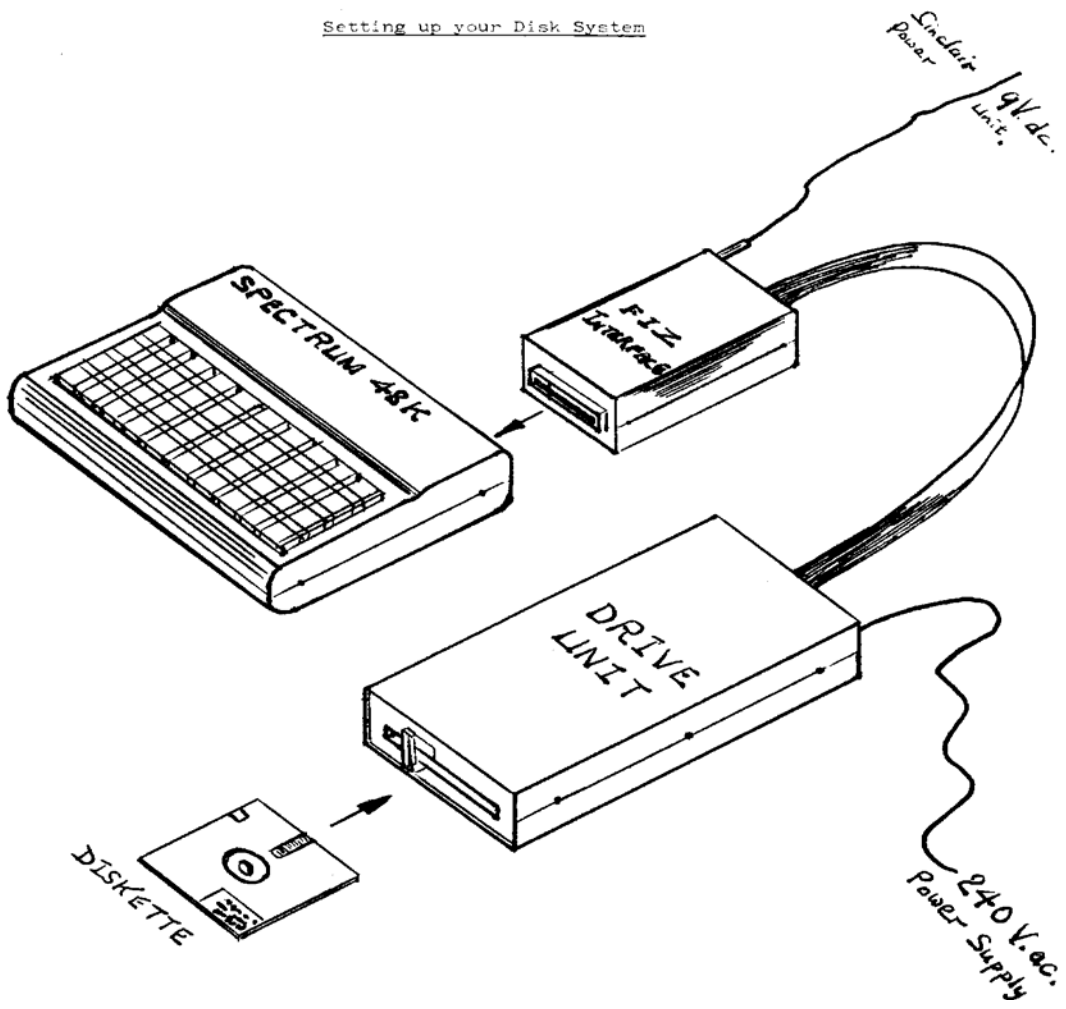
COMMAND ROUTINES
(and Reserved Variables)

| | | |
|---|---|---|
| RES | = | Re-Start (boot-up) |

|  |  |  |
|---|---|---|
| ND | = | New Diskette formatting |
| DIR | = | DIRectory display |
| ZAP | = | File delete facility |
| BS | = | BASIC program SAVE |
| BL | = | BASIC program LOAD |
| CS | = | Code  (Binary)  SAVE |
| CL | = | Code  (Binary)  LOAD |
| AS | = | Array Save |
| AL | = | Array Load |
| BACK | = | Whole Diskette Copy      (Back-up) |

|  |  |  |
|---|---|---|
| F$ | holds | "filename"  (program/data name and descriptors) |
| U$ | holds | User Defined Graphics during Save/Load processes. |

## Setting up your Disk System



## APPENDIX D

SYSTEM OPERATION - GETTING STARTED

1. Check that your system is set-up correctly - See Appendix C. Ensure that there is not a diskette in the drive unit.

2. Apply power to the drive - the switch is at the rear of the drive unit. Now apply power to the Interface. Instead of the Sinclair Research Copyright message a 'start-up' (RESTART) Macronics Systems Ltd., Copyright message will appear and the disc drive unit may click.

3. Select a diskette and with the labelled side uppermost, carefully insert it in the drive unit and close the door. If the diskette is blank (unused) then continue from paragraph 5, otherwise 4.

4. DIR (or dir) directory list

```
Type:   (PRINT) (USR)  DIR (ENTER)   () denotes Sinclair function key.
```

The drive will click and whirr, and in a few seconds the contents of the diskette will be displayed on the screen. If only a two digit number is printed on the screen then an error has been detected.

At this stage the error is probably because the diskette has not been formatted or there is a poor physical connection. Remove the diskette and check the label for signs of previous use. If none, then proceed with paragraph 5. If the diskette has previously been used then check all connections, it may be that the ribbon connector on the drive unit has become loose in transit for example. Repeat the process from the beginning (paragraph 1).

5. ND (or nd) New Diskette Formatting

Warning: The New Diskette process destroys all previously recorded information on the diskette.

Insert a blank (unused or reclaimed) diskette and type:-

```
(LET)  F$  = "DISKl"   (ENTER)  (up to 6 characters for the diskette
                                              name)

then type   (PRINT)   (USR)   ND     (ENTER)
```

The drive will switch on and step from track 0 to track 39 writing formatting information and marking unuseable tracks. If track 0 is unuseable you must discard the diskette - this will be shown by ERROR CODE 3. The whole process of formatting a diskette will take less than 2 minutes; on successful completion a 0 will be printed in the Top Left hand corner of the screen. You can now try the DIR command to print the directory of the diskette.

6. <u>BS (or bs) BASIC SAVE</u>

The following is a simple example of saving a BASIC program on diskette:-

```
Type:    10    (FOR)   n = 1  (TO)  20               (ENTER)
         20    (PRINT) "My first BASIC program"      (ENTER)
         30    (NEXT)   n                            (ENTER)
```

```
Now type     (LET)  F$   =   "Progl", 10     (ENTER)
```
This gives the title to the program as a 'filename'

```
type:     (PRINT)  (USR)   bs    (ENTER)
```

The program will be written to the diskette and if successful a zero (0) will be displayed at the Top Left Hand Corner of the Screen (TLHC) (If a zero is not displayed, then an error code 0 will be; refer to Appendix A to ascertain the type of error).

```
Now type:     (LET)  e  =  (USR)  DIR    (ENTER)
```

The screen will display the diskette directory and identify the presence of your program on it:-

```
DISK1       38
BASIC      Progl      01
```

7. <u>BL (or bl) BASIC LOAD</u>

Let's load the program into memory from diskette and see what happens;

```
Type:     (LET)     F$      =      "Progl"     (ENTER)
                (PRINT)  (USR)   bs                 (ENTER)
```

The screen will display 'My first BASIC program' twenty times. On loading the program an automatic RUN has been initiated.
This results from the program being saved using the 'line number' option - see paragraph 6 where F$ is defined. If this option had not been selected then the instructions you have just typed would have loaded the program but 'RUN' or 'GOTO' would then have to be typed to execute it.

Note: that if RUN (or NEW) is used at any time then the disk system reserved variables will be lost.

8. The principles of using the disk system have now been covered - other commands operate in the same way and you are advised to experiment with them before undertaking any serious work. An example follows, which

serves to demonstrate the way commands cs and CL can be used to re-locate code in addition to just storage on diskette.

<u>CS    (or cs)    code SAVE</u>

```
10  (FOR)  n  =  50000  (TO)  50020     (ENTER)
20  (POKE)   n,99  :   (NEXT)  n        (ENTER)
(GOTO)  10     (ENTER)

(LET) F$ = "CODE, 50000, 20"            (ENTER)
(PRINT)   (USR)    CS                   (ENTER)
```

<u>CL    (or cl)    CODE LOAD</u>

```
(LET)  F$  =  "CODE,40000, 15"          (ENTER)
(PRINT)   (USR)  cl                     (ENTER)

10  (FOR)  n  =  39998 (TO) 40018      (ENTER)
20  (PRINT)  n, (PEEK) n : (NEXT) n    (ENTER)
(GOTO)  10    (ENTER)
```

9. If you have any difficulty whatsoever you are advised to check the detail in the body of the manual.

# Fiz-diskinterface

DATA
Eprom
ADRESS A∅-A11
OE
20
D∅-D3
INDEX 8
TRACK ∅ 26
WRITE PROT 28
READ DATA 30
MOTOR ON 16
DRIVE 1 10
DRIVE 2 12
SIDE SEL 32
STEP 20
DIR 18
WR.GATE 24
WRITE DATA 22
DATA
CE
4.000 kc/s
MEMREQ
A15
A14
A13
ROMCS
RESET
WR
RD
A5
A∅
A1
A2
A3
A4
A7
IOREQ
A6
DATA D∅-D7
ser
mde
clk
pr
clr
data
Q
OUT cont.
DATA D∅-D7
en
ld
OD OC OB OA

|  |  |  |
|---|---|---|
| 1 = EPROM | 8 = 7495 | 15 = 7474 |
| 2 = 7805 | 9 = 7474 | 16 = LS30 |
| 3 = 7430 | 10 = 7403 | 17 = 7400 |
| 4 = 7400 | 11 = LS365 | 18 = LS04 |
| 5 = 7403 | 12 = LS373 | 19 = LS161 |
| 6 = LS373 | 13 = 7496(5?) | 20 = LS04 |
| 7 = LS373 | 14 = 7400 | 21 = LS02 |
|  |  | 22 = LS27 |
|  |  | 23 = LS161 |
|  |  | 24 = 7410 |
|  |  | 25 = 7402 |

Drawing:

Roelof Koning 1991