SPECTRUM DISC INTERFACE
KEMPSTON

KEMPSTON DISC OPERATING SYSTEM

(c) By D.J. Farmborough BSc DipEd & D. Koveos BSc
ABBEYDALE DESIGNERS LIMITED

The Kempston disc interface for the ZX Spectrum provides a simple upgrade from cassette or microdrive based systems to floppy discs. The operating system provides a powerful command library, including all the standard file manipulation functions and a number of additional commands to make K-DOS arguably the most powerful disc operating system (DOS) currently available for the Spectrum.

1.0 INTRODUCTION

The Kempston interface can be used with any standard independently powered BBC disc drive. Up to a maximum of four drives can be connected to the system, be they 5.25, 3.5 or 3" drives, provided they are capable of operating in double density. Connection between your disc drive and interface should be made through a 34-way ribbon cable terminating with a 34-way male IDC connector at the interface end and a standard Shugart connector at the drive end. If in doubt your supplier will be able to advise you on a suitable cable.

IMPORTANT: UNDER NO CIRCUMSTANCES SHOULD YOU ATTEMPT TO FIT OR REMOVE THE INTERFACE WHILST THESPECTRUM IS POWERED UP AS SERIOUS DAMAGE MAY OCCUR TO BOTH THE INTERFACE AND YOUR SPECTRUM.

1.1 INSTALLATION

The interface fits into the expansion port at the back of the Spectrum. It will plug in directly, or if you are using other interfaces (ie. Kempston's Centronic E) then via the Kempston interface adaptor. It is advisable to apply power to the Spectrum before the disc drive; damage may occur if this sequence is not adhered to.
On powering up the system the Spectrum will display a copyright message to indicate the presence of the disc interface;

```
KEMPSTON     DOS  V2.0          ;0:0
ABBEYDALE DESIGNERS LTD 1985
```

and the LED will be illuminated. If this message does not appear then check that you have installed the interface correctly. If you can find no fault in installation then consult your supplier for advice.

If the message appears then your disc system is ready for use.

Before a disc can be used it needs to be formatted, that is the process of dividing the disc into tracks and sectors. To format discs you will need to know the characteristics of your drive(s), and supply the DOS with the following information:-

a) the number of tracks (35, 40 or 80)
b) the number of sides (1 or 2)
c) the stepping rate (6, 12, 20, 30 ms)

where the stepping rate is the time taken by the head in moving from one track to the next. When formatting it is necessary to select a stepping rate greater than or equal to the stepping speed of your drive. The FORMAT command is described in full in section 2.8.

The disc system can be reset at any time by the reset switch on the top of the interface.

WARNING: A system reset will destroy any programs currently in the DOS workspace. Care should be taken in using the reset function.

2. DISC INTERFACE COMMANDS

2.0 COMMAND SYNTAX

The Kempston disc system operates as an extension to Sinclair BASIC and enhances the performance of the Spectrum by combining high speed and capacity with a powerful set of commands. The following file types are supported:

BASIC programs saved with or without a line number
CODE for machine code, SCREEN$, text files etc
Character arrays for string data, eg. DATA n$( )
Number arrays for numeric data, eg. DATA n( )
Sequential access files

All K-DOS commands are prefixed by a ‘software switch’ to distinguish disc commands from standard Sinclair syntax. The keyword PRINT following a DOS command passes parameters needed for specific operations.

So the K-DOS command syntax is as follows:

```
PRINT #4 : command : PRINT p1, p2...
```

where p1, p2 etc refers to any parameters needed by the particular command.

The parameters are particularly important when using commands to access multi-drive systems. So for a number of commands p1, p2 etc refer to the particular drive selected. It should be noted that once a drive has been selected, it becomes the DEFAULT drive and does not need specifying again. The PRINT statement can thus be omitted. Any references to files other than on the default will result in a “Record not found” error.

The following sections describe the commands available under K-DOS and it is recommended that these are read carefully before using the system.

2.1 Multi-statement lines

K-DOS commands can be accomodated on the same line as standard BASIC statements.

However, since PRINT is used to pass parameters into the DOS care needs to be taken in the use of the standard PRINT statement following a K-DOS command.
To avoid confusion when using multistatement lines use of a double colon will distinguish innocent PRINT statements from a DOS keyword.
For example:

PRINT #4: command : : PRINT “string”

would execute the K-DOS command, followed by a print “string”.

2.2 CATalogue command

CAT will display a catalogue of all files on a specified disc drive.

Syntax: PRINT #4: CAT
PRINT #4: CAT : PRINT d
PRINT #4: CAT : PRINT d , “string”

The catalogue format as specified by CAT is as follows:

file_type:= the type of file, ie. BASIC, bytes, $data, ndata, sequ
file_size:= the size of the file in multiples of 1 kbytes (1024 bytes)
file_name:= a BASIC filename

So for example,

PRINT #4 : CAT : PRINT d

will catalogue all files on the specified drive (1 – 4). The current drive is selected if the keyword PRINT is not included.
It is possible to do a selective catalogue by:

PRINT #4 : CAT : PRINT d , “string”

where d is the drive specification and “string” is an alphanumeric string.
This will select all files with the file_name containing “string”. Specification of the drive is compulsory.
To direct the catalogue through a suitable printer interface then,

OPEN #2 ,”p” : PRINT#4 : CAT : CLOSE #2

will print the catalogue if the interface is initialised correctly.

2.3 CLEAR command

CLEAR will delete a block of BASIC lines starting at a specified line and ending at a specified line

syntax: PRINT#4 : CLEAR : PRINT m1, m2

where 0 < m1, m2 <= 9999. (m1=first line, m2=last line)

CLEAR offers the programmer the ability to use ‘BASIC OVERLAYS’ to reduce the memory usage in large BASIC programs (see section 4)

2.4 CLEAR 0 command

CLEAR 0 will substantially reduce the amount of memory taken by a BASIC program without altering the logic flow of that program.

syntax: PRINT#4 : CLEAR 0

The BASIC size reduction is achieved by replacing all numbers by ‘VAL “nn” ‘(saving 3 bytes for each number), and integers 0, 1 and 3 by NOT PI, SGN PI and INT PI respectively (saves 5 bytes).
When the CLEAR 0 command is used the screen is cleared and the amount of free space before and after the command is shown. Some large BASIC programs (eg. TASWORD II) will liberate MORE memory than is reserved by the DOS.

If a RENUMBER utility is available then re-numbering should be done prior to using this command.

2.5 CLOSE command

CLOSE is a Sequential Access file handling command. It will close stream #n and reclaim the buffer.

syntax: PRINT#4 : CLOSE #n

where n is the stream 0 <= n <= 15 n< > 4.

Using stream n without opening it again (after CLOSE) will cause an ‘Invalid stream’ error report. No drive specification is required by the CLOSE statement.
See the section on Sequential Access files for further details on the use of this command.

2.6 COPY command

COPY is a tape to disc transfer utility which allows transfer of most cassette programs to disc.

syntax: PRINT#4 : COPY : PRINT d

where d is an optional drive specification indicating the destination drive. Default drive used if d is not specified.

Once invoked, the command batch processes your tapes until a BREAK is detected.

IMPORTANT: THE TAPE MUST BE STOPPED ON A COMMAND PROMPT TO ALLOW BLOCK STORAGE TO DISC.

On saving the program the command displays:

file_name	:= disc filename (identical tape names stored with number appended to the tape_name, incremented for subsequent files)
start_address	:= start address of file
file_length	:= length of file in decimal

Headerless blocks are saved using the filename “default” with appended numbers if necessary.

Use of the COPY command retains whatever protection was originally built into the program, so there is no guarantee that programs will run under the DOS. A break into the BASIC loader will allow modification of all occurences of LOAD, SAVE, MERGE to correct K-DOS syntax. Each file call needs to be modified if the filename on disc is different to that on tape.

BASIC programs that use machine code imbedded in REM statements will only run if the code is relocatable and the USR address is expressed relative to BASIC variables (PROG etc).

2.7 ERASE command

ERASE allows the deletion of files from the specified drive’s directory.

syntax: PRINT#4 : ERASE “file_name”
PRINT#4 : ERASE “file_name” : PRINT d

where d is an optional drive specification (1 – 4), can be a constant, numeric variable or expression.

For example,

PRINT#4 : ERASE “file_name” : PRINT 1

would delete a file “file_name” from drive 1.

A ‘WILD-CARD’ deletion is available to allow selective block deletion of files on a disc. The wildcard character is ‘^’ (up arrow). The wild-card can be assigned any alphanumeric character, so for example:

PRINT#4 : ERASE “file_name^” : PRINT 1

would delete all files on drive 1 with a ten character filename referenced by the first nine characters “file_name”. (This includes file_name1, file_name2, file_name*, file_name$, file_name? etc)

Erasing a file from a disc with a WRITE PROTECT tab on, will generate a ‘Write protected’ error report, and a return to the DOS

NOTE: A K-DOS SAVE DOES NOT REQUIRE THE PRIOR DELETION OF A SIMILARLY NAMED FILE. SAVE AUTOMATICALLY OVER-WRITES AN ALREADY EXISTING FILE.

2.8 FORMAT command

FORMAT will format a disc in the specified drive.

syntax: PRINT#4 : FORMAT “disc_name”: PRINT p1, p2, p3, p4

where disc_name is any valid name up to 10 characters and,

p1 := the number of the specified drive (1 – 4)
p2 := the number of tracks
p3 := the number of sides (1 or 2)
p4 := the stepping rate (1 to 4

where the stepping rates correspond to 6, 12, 20 and 30ms.

So for example: A Mitsubishi 80-track, 2-sided drive with a stepping rate of 3ms could be formatted on drive 1 as follows,

PRINT#4 : FORMAT “disc_name” : PRINT 1,80,2,1

A CATalogue of which would show a disc capacity of 800 kbytes 795 of which are available to the user. (The other 5 bytes are used by the directory).

2.9 GOTO command

GOTO is equivalent to an “ON ERROR GO TO line number” command.

syntax: PRINT#4: GOTO m1

where m1 is a line in the BASIC program.

All errors, including the error reports ‘OK’, ‘STOP’, etc are trapped and control is passed to a line in the BASIC program. So it is possible to trap an error, and either re-run the program or report your own error. The GOTO command gives complete anti-BREAK protection and the facility to do all necessary housekeeping work (closing OPEN files etc) before reporting an error.

2.10 INKEY$ command

INKEY$ will read the next character of a Sequential file opened on #n.

syntax: INKEY$ #n

where n is the stream.

INKEY$ returns the next character from a sequential file. So from a file ‘file_name’ on drive 1, INKEY$ would return successive characters from the file. If an attempt is made to read a character after the end of a file a ‘End of file’ error report is generated.

2.11 INPUT command

INPUT will read the next record of a sequential file opened on stream n.

syntax: INPUT #n;var 1;var 2;var 3... var n

where var n is an individual data record.

INPUT #n is similar to the keyboard INPUT command, but the keyboard is replaced by an open disc file. Data is input sequentially, the file pointer being incremented by one complete record on each read. CLOSE #n followed by OPEN #n resets the file pointer to the start of the file.

INPUT needs information on the format of the data in the file, ie destination (string or numeric variable), terminator used to data save, to read successfully. Any characters not recognized by the standard keyboard INPUT are also un-acceptable to INPUT #.

So for, example,

INPUT # n ; LINE f$.....

will input a sequential file containing string records.

2.12 LOAD command

LOAD will load a program from the specified drive.

syntax: PRINT#4 : LOAD “file_name”
PRINT#4 : LOAD “file_name” : PRINT d
PRINT#4 : LOAD “file_name” file_type : PRINT d
file_type := CODE, SCREEN$, DATA a( ), DATA a$( ), etc
where d is an optional drive specification.

For example,

PRINT#4 : LOAD “file_name” SCREEN$: PRINT 1

will load a file ‘ file_name ‘ from drive 1, saved in the format of a screen.

Programs will AUTO-BOOT after a reset or power-on if they are SAVEd as file ‘AUTO’ with a line number (see sec 2.18).

2.13 MERGE command

MERGE will load a BASIC file from the specified drive and merge it with the program currently in memory.

syntax: PRINT#4 : MERGE “file_name”
PRINT#4 : MERGE “file_name” : PRINT d

where d is an optional drive specification.

MERGE offers the programmer the ability to use ‘BASIC OVERLAYS’ to reduce the memory usage in large BASIC programs (see sec 4).

2.14 MOVE command

MOVE will copy files from a specified drive to another specified drive. It can also rename files on the specified drive.

```
syntax:    PRINT#4 : MOVE “file1”,“file2”
           PRINT#4 : MOVE “file1”,“file2” : PRINT d
           PRINT#4 : MOVE “file1”,“file2” : PRINT d1, d2
           PRINT#4 : MOVE “file1”,“” : PRINT d1, d2
           PRINT#4 : MOVE “”,“” : PRINT d1, d2
           PRINT#4 : MOVE “”,“string” : PRINT d1, d2
```

where d1, d2 are optional drive specifications.

For single drive users d1 = d2 and prompts are displayed automatically.

MOVE can be used to rename discs or files on a specified drive. For example:

```
PRINT#4 : MOVE “file_name1”, “file_name2” : PRINT 1
```

where file_name1 := existing file or disc name
file_name2 := new file or disc name
d := optional drive (1 – 4)

will rename the disc or a file on drive 1 named ‘file_name1’, file_name2. MOVE checks that file_name1 is a disc or file identifier before renaming.
If the file cannot be found on the specified drive then a ‘record not found’ error is generated.

There are two levels of file transfer available; a single file copy and a block copy.

Single file copy can be accomplished as follows:-

```
PRINT#4 : MOVE “file_name1”,“file_name2” : PRINT d1, d2
```

where d1 is the source drive, and d2 is the destination. If file_name2 = “” then a copy with the same name as the source file is created.
The drive numbers, d1 and d2, can be identical to allow file back-up. If d1=d2 then K-DOS prompts the user to swap diskettes as and when necessary.

Diskette copy, back-up, can be achieved by,

```
PRINT#4 : MOVE “”,“” : PRINT d1, d2
```

where d1 is the source drive, d2 the destination drive.

All copiable files are transfered from drive d1 to d2. Single drive prompts are displayed if a single drive is being used.

A variant of the MOVE command is available using a ‘WILD-CARD’ definition of files.

PRINT#4 : MOVE “”,”string” : PRINT d1,d1

where ‘string’ is an alphanumeric string.

For example,

PRINT#4 :MOVE “”,”/BAS” : PRINT d1,d2

will copy all files on drive d1 with the extension /BAS to drive d2.
This command can prove useful if an extension system is applied to the definition of files, ie. /BAK for back-ups, /BAS for BASIC programs, /OBJ for object files etc.

2.15 NEW command

NEW is similar to the BASIC NEW in that it will clear any program and variables in memory. However the K-DOS NEW does not overwrite the DOS variables, and the DOS remains enabled.

syntax: PRINT#4 : NEW

2.16 OPEN command

OPEN is a Sequential file handling command. It opens stream n to sequential file ‘file_name’ on the specified drive, and assigns a buffer to it. If the file already exists, it is opened for reading, otherwise it is opened for writing.

syntax: PRINT#4 : OPEN #n, “file_name” : PRINT d

where n := stream number assigned to buffer 0,= n ,=15, n < > 4.
d := optional drive specification

Whenever a file is used one of more buffers need to be assigned to it. Before a file can be read or written into it has to be OPENed. OPEN creates a buffer, accessible to the user by its attachment to a stream. 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 buffer by opening it to different streams (memory allowing). An attempt to re-open a newly created file will generate a ‘file already open for writing’ error report.

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

Whenever a buffer is created, the BASIC program and its variables are ‘pushed’ up by about 570 bytes.
Consequently machine code programs imbedded in a REM statement will not execute correctly unless they are fully relocateable. Furthermore, having all 15 buffers open accounts for 8 kbytes of buffer space.

NOTE: DO NOT REMOVE A DISC CONTAINING OPEN FILES. CLOSE SAVES THE FILE BUFFER, AND SETS THE END OF FILE MARKER.

It should be noted that it is possible to OPEN streams 0 – 3 for sequential input or output, and when they are closed 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, LLIST as well as PRINT #3 and LIST #3. So, by

OPEN #4 : OPEN #3, “print_file”

it is possible to re-direct all printer output into a disc file which can be read and printed after the main task has been completed.

2.17 PRINT command

PRINT will write individual data records into a buffer for subsequent transfer to a sequential file.

syntax: PRINT #n ;var 1’var 2’ ..... ‘var n

where var n is an individual data record.

PRINT #n is similar to the screen PRINT command, however the use of terminators and seperators is important.
PRINT #n accepts all the standard separators; semicolon, coma and apostrophe.
Care must be taken in the use of a semicolon as a separator since it will merge two records. For example;

PRINT #n ; a$; b$

would be saved as string1string2 <CR>, which could not be read back into two variables. This can be achieved by;

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

which would be saved as string1 <CR> string2 <CR>. Similar care must be taken in the use of a comma, since records will be saved with the same format as if they had been printed, separated by a number of nulls.

2.18 SAVE command

SAVE will save a BASIC program to the specified drive.

syntax: PRINT#4 : SAVE “file_name”
PRINT#4 : SAVE “file_name” LINE m : PRINT d
PRINT#4 : SAVE “file_name” file_type: PRINT d

where file_name := a BASIC file identifier
file_type := type of file, ie. CODE, SCREEN$, DATA a( ), DATA a$, etc

For example,

PRINT#4 : SAVE “file_name” SCREEN$: PRINT 1

will save a file ‘ file_name ‘ to drive 1 as a screen.

It is possible to save an auto-booting program under K-DOS. Any program saved wit a line number and with a filename ‘AUTO’, will auto run on a system power-up or reset. So, for example,

PRINT#4 : SAVE “AUTO” LINE 10 : PRINT 1

will save an auto-booting program to drive 1.

Auto-booting programs need not be written in BASIC, and a machine code program will auto run from its first memory location.

The auto-booting feature allows self-running, un-BREAKable programs to be written; since if the first line of an auto-running program disbles the BREAK key detection such a program would be impossible to read.

3.0 SEQUENTIAL ACCESS FILES

A Sequential access file is a file consisting of a number of individual data records. A data record being an alphanumeric string or numeric data whose end is marked by a terminating character (comma, apostrophe, CR code etc). The beginning of a record is usually defined as the character after an end of record character. So a sequential access file can be considered as a collection of n records which can only be read starting with record 1, and incrementing the file pointer by one upon each read operation until the n th record has been accessed. Attempts to read a non-existent (n+1) record will generate an ‘end of file” error report. The size of a sequential access file will depend on the available space on the data disc and so, the demand on memory is very modest.

K-DOS contains a number of commands for sequential access file handling OPEN #n, CLOSE #n, INPUT #, INKEY$, and PRINT #. Reference to each command description will give an indication of the application of each of these commands.

4.0 BASIC OVERLAYS

The K-DOS versions of MERGE and CLEAR offer the BASIC programmer immense programming power by allowing the construction of ‘BASIC OVERLAYS’ to reduce the memory usage in large programs.
If a BASIC program, saved with a line number, is MERGed it will auto run from that line number having first been merged with the main program. A large BASIC program could be written in modular form with the 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 be simply to load the appropriate

module (overlay), run it and then return to the menu. The modules need to be auto-running because in this way they are protected and can run from any line number.
If the MERGed module doesn't use exactly the same line numbers as the one before you would expect errors due to the execution of lines that should not be there. However CLEAR simply acts as a block delete to prevent this.

APPENDIX

1. 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. CLEAR n will cure this by raising RAMTOP. If this error occurs while moving a sequential access file it may be too large to MOVE in this way and should be copied by re-writing to the new file record by record.
6 Number too big	: Used to check parameter values eg. drive number in disc command. Check your command carefully. If error persists it may be due to a DOS error, cleared by re-booting the system. Most likely to occur after 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 file name longer than 10 characters. For all commands except ERASE, a "^" in the filename will produce this error.
H STOP in INPUT	: You have used INPUT with a seq. file which contains non-printable 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 vice-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 #4 you must reboot the system or use RANDOMISE USR 58500.
Q Parameter error	: see error 6

The following messages are additions concerned mainly with errors generated by K-DOS commands:-

S Drive inoperable	:	You have specified a drive number not used by your system, the door is open, the disc is inserted incorrectly or the drive is faulty.
T Disc data lost	:	A DOS error has occured. Try repeating the command after a CAT. If this fails the disc 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 disc. Check the spelling against a CAT. Note that keywords or special characters should be used consistently. If the error persists then suspect the disc.
W Write protected	:	You have attempted to write to a disc that has a write protect tab on it. This will also occur if you attempt to OPEN a file that does not exist on a write-protected disc. Channel must be closed to remove the error condition.
X Disc faulty	:	The system has failed to write or read to the disc after 50 attempts. Retry the command and then suspect the disc.
Y Illegal command	:	You have attempted to use commands not supported by the system and not saved as an executive file.
Z Copyright message		
a Wrong filetype	:	Occurs when attempting to load a file with the wrong specification. 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 disc. There may be fewer than 144 files as extra entries are used for sequential files larger than 48K and for all files if the disc is becoming full. The problem may be solved by doing a complete copy of the disc which will ensure efficient use of the directory entries.
d Disc full	:	All space on the disc has been used. The last save operation will have been terminated and the file should be saved to another disc. 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 of 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.
i Verification failed	:	Try reformatting the disc. If this error is repeated the disc should be replaced as it is probably faulty.

2. K-DOS COMMAND SUMMARY

The following disc operating system commands are supported by K-DOS:

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

CAT	Gives a catalogue of all files of the CURRENT drive.
CAT:PRINT d	Gives a catalogue of all files of drive d (1 – 4).
CAT:PRINT d,"string"	Gives a catalogue of all files on drive d, whose name contains 'string'.
CLEAR:PRINT m1, m2	Deletes a block of BASIC lines starting at m1 and ending at m2. 0<m1, m2,=9999.
COPY:PRINT d	Tape to disc transfer utility. Optional drive specification indicates the destination.
ERASE "filename"	Erases 'filename' from the directory of CURRENT drive. UP-ARROW acts as a WILD-CARD character.
ERASE "filename":PRINT d	Erases 'filename' from directory on drive d.
FORMAT "discname":PRINT i, j, k, l	Formats disc on drive 1 with j tracks, k sides, i step rate.
GOTO m	Forces BASIC program to jump to line m on any error.
INKEY$ #n	Reads the next character of a sequential file opened to stream n. 0 <= n <= 15. n ,< > 4
INPUT #n;var 1;var 2;var 3 ..	Reads the next record of a sequential file opened to stream n.
LOAD "filename"	Loads or loads and auto-runs the BASIC program 'filename' on CURRENT drive.
LOAD "filename":PRINT d	As above but the program is found on drive d.
LOAD "filename":filetype	As above but filetype can be CODE, SCREEN$, DATA a( ), DATA a$( ) etc. PRINT d gives optional drive.

MERGE “filename”	Merges or merges and auto-runs the BASIC program ‘filename’ with the program in memory.
MERGE “filename”:PRINT d	As above but program is found on drive d.
MOVE “file1”,”file2”	Rename file1 to file2 in the CURRENT drive.
MOVE “file1”,”file2”: PRINT d	Renames file1 to file2 in drive d
MOVE “file1”,”file2”:PRINT d1, d2	Copies file1 on drive d1 into drive d2 with the filename file2
MOVE “file1”,””:PRINT d1, d2	Copies file1 on drive d1 into drive d2 with the same filename.
MOVE “”,””:PRINT d1, d2	Transfers all copiable of drive d1 to drive d2.
MOVE “”,”string”:PRINT d1, d2	Transfers all files whose name contains ‘string’ from drive d1 to drive d2.

For single drive users, d1 = d2 and prompts are displayed automatically.

NEW	Like BASIC NEW except that the DOS remains enabled.
OPEN #n, “filename”	Opens stream n to seq. file “filename” of the CURRENT drive and assign a buffer to it. If the file already exists it is opened for READING, else for WRITING.
OPEN #n, “filename”:PRINT d	As above but drive d is used for ‘filename’.
PRINT #n;var 1’var 2’	Records var 1, var 2 etc are written into buffer n for transfer to a sequential file.
SAVE “filename”	Saves the BASIC program ‘filename’ onto the CURRENT drive.
SAVE “filename” LINE m	As above but program is to auto-run from line m.
SAVE “filename”filetype	Filetype can be SCREEN$, CODE, DATA a( ), DATA a$( ) etc. PRINT d can still be used as an optional drive identifier.

NOTES

Singer Way, Kempston, Bedford MK42 7AW.
Tel: (0234) 856633.