SEDORIC DOS
Version 2.0

A USER'S GUIDE

based on the French SEDORIC V1.0 manual
by Fabrice Broche and Denis Sebbag

by Jon Haworth and Allan Whitaker

Issue 3 – First Revision November 1992

CONTENTS

SEDORIC - A USER'S GUIDE

INTRODUCTION

SEDORIC is a fast, sophisticated Disc Operating System (DOS) for the Oric computer. It is capable of reading or writing 24 kilobytes of code in less than 3 seconds. Despite the sophisticated features of the DOS the amount of memory taken away from the user is minimal. Pages 1 and 2 are not used. On booting the SEDORIC DOS disc the DOS code is loaded into memory addresses #C000 to #FFFF in RAM overlay. The DOS makes use of memory locations in page 0 and page 4 is entirely allocated to SEDORIC. The Oric ROM is switched in and out as needed in order to access the DOS or BASIC.

Booting SEDORIC carries out a NEW command on any program in memory, though a BASIC program can be recovered with the OLD command. Before control is passed to BASIC, SEDORIC executes a series of initialisation instructions particular to the disc. These may be modified by the user, to include the loading and execution of your own program to personalise the boot. Two types of discs can be formatted - a 'Master', containing the full DOS, and a 'Slave' which can only be used when the DOS is already in memory. If the DOS is missing, the error message '**WARNING** DOS is altered!' is displayed, requiring the insertion of a Master disc into the system drive if the DOS in memory has been affected.

You are strongly advised to make a backup copy of the SEDORIC disc for daily use, preserving the original. To do this type BACKUP (or BACKUP A TO B if you have two disc drives of the same type) and press RETURN. Press RETURN again and answer 'Y' to the question 'Format target disc Y/N'. Then follow the displayed instructions to carry out the BACKUP.

To display the contents of your SEDORIC disc, type DIR and press RETURN. On the Oric Atmos you can press FUNCT and] together instead. To load a program displayed in the directory, simply type in its name and press RETURN. To exit and reboot SEDORIC, just press the disc drive RESET button.

To save a BASIC program, type SAVE "NAME" and RETURN. A DIR will prove it has been saved to disc. You will note that the directory of the disc displays the file as NAME.COM. The '.COM' is known as the filename extension and will be covered in more detail in the section on filenames. For now we can ignore it. To recall a program, just type its name (as before), i.e. NAME + RETURN. You can also type LOAD "NAME" which has the same result.

If you wish to remove a file from the disc, type DEL "NAME" and RETURN. To rename a program, use REN. Thus REN "NAME" TO "TEST" will change the name of 'NAME' to 'TEST'.

To initialise a blank disc for use by SEDORIC only requires that you type INIT and RETURN. You will probably come across the message "INSERT MASTER DISC IN DRIVE A". Do so, and press RETURN; the routine you are using will then be loaded into memory. This is necessary because the whole of the DOS is too large for the available memory. Certain commands therefore have their routines stored on disc ready to be loaded when required. Once loaded the command can be used repeatedly until another such command routine has to be loaded (e.g. BACKUP) or the computer is switched off. Put in your blank disc, then answer the displayed questions.

As with the BACKUP of a disc, you can make a copy of your programs by using the COPY command (e.g. COPY "TEST") and again follow the displayed instructions.

After this brief introduction, you are recommended to read the section entitled 'Disc operating commands' starting on page 4 of the manual to discover the full potential of SEDORIC.

Furthermore, SEDORIC provides you with some 60 additional BASIC instructions, and a comprehensive suite of sequential and random access file routines. You will find plenty to keep you occupied while learning to get the best out of SEDORIC!

Note that the SEDORIC disc contains several utilities in addition to the DOS. These are described in detail in the final appendix, but take particular note of the CONVERT utility to transfer Oric DOS programs to SEDORIC format.

SEDORIC DOS makes full use of the FUNCT key on the Oric Atmos. In conjunction with the other keys it can provide many functions which are used simply by pressing FUNCT and another key or keys. You can set up your own function keys with the KEYDEF and KEYUSE commands. Thus, FUNCT and the / key executes the LIST command; FUNCT and RETURN together provides automatic numbering of program lines.

A new control character – CTRL P – has been added to provide an on/off toggle for the cursor flash. Within programs use PRINT CHR$(16).

Facilities are provided to avoid inopportune system 'crashes'. SEDORIC controls any BREAK and displays a 'BREAK ON BYTE #XXXX' message. The ESC key provides a way out of all interactive commands such as DIR, DEL and INIT.

Whereas all Disc Operating Systems previously offered for the Oric computers have required keywords to be prefixed with the exclamation mark (!) SEDORIC DOS allows the entry of keywords without this need. The exception is following the use of the QUIT command which makes the prefixing of DOS keywords or filenames (in direct mode) with the exclamation mark mandatory.

Finally, all DOS keywords can be typed in upper or lower case, or even in a combination of the two. BASIC keywords can safely be used in filenames, providing they do not form the start of a filename being loaded in direct mode, e.g. STOPPER. Filenames can also be typed in lower case letters although if BASIC keywords are embedded they must always be typed in upper case, as in 'STOPper'.

Conventions and notations

The following conventions and notations are used in this manual:

A COMMAND appears in bold capital letters, general syntax in ordinary capitals. Optional parameters are enclosed in brackets. A command is entered by pressing the RETURN key.

Abbreviations:

Drive – the parameter is a drive number – A, B, C or D (as in MS-DOS). The default is the current drive.

FN – File name required, must not contain wildcards

FNA – File name ambiguous – File name required, but can include wild card characters (? and *). It is possible to omit the filename altogether or just to specify the drive name.

NE – Numeric expression

AE – Alphanumeric (string) expression

NV – Numeric variable – a variable name is required

AV – Alphanumeric (string) variable

LN – Logical number – refers to a block number used in the control of files.

FLN – Field name – used in random access file control

The ESC key enables you to exit from all positions where input is required from the user.

Filenames

A filename is generally a string of characters enclosed in double quotes, or a string expression. An exception is adopted for the loading of files in direct mode, when the filename alone suffices.

A filename may consist of up to three parts:

(i) the drive to use (optional)

The drives are designated A to D, and when specified the letter is followed by a hyphen. When not specified the current (default) drive is used. The use of letters to denote each drive allows the drive to be specified in direct mode. The use of numerals would be treated as lines of a BASIC program.

(ii) the file identification name ('the filename')

Of up to 9 characters, this must be specified. Only alphabetic and numeric characters may be used, save for the wildcard characters '?' and '*'.

(iii) the file extension (optional)

When specified, this is separated from the filename by a full stop. It comprises up to three characters, enabling files to be categorised (e.g. .BAS, .BIN etc). When no extension is specified the default is assumed (see the EXT command). On initialisation the default is .COM.

Ambiguous filenames (FNA) are valid for such commands as DIR, DEL and COPY etc. The '?' is substituted for individual letters, the '*' for a group of characters to the end of the name. So DEL "*.BAS" deletes all files with the extension .BAS, whereas DIR "M*" lists all filenames starting with the letter M sharing the default extension. The asterisk does not extend beyond the part of the filename it is placed in. In DOS commands where ambiguous filenames are allowed it is possible to omit the drive, extension or filename characters, or to substitute wildcard characters (e.g. *.*). In the latter example, all files on the default drive are accessed.

DISC OPERATING COMMANDS

(drive)–

On its own, it sets the default (or current) drive. Used with a filename it allows temporary access to the specified drive but the default drive is unchanged. The drive name is optional in the filename. If used with the LOAD command it must be within the quotes for the filename. If not specified, the default drive is accessed.

B–	selects drive B as the default
A–MENU	load MENU(.COM) from drive A, but B would remain the default drive

LOAD "C–HELP.BAS"

EXT AE

Specifies the default file name extension to 3 characters. On initialisation the default name is .COM

EXT "BAS"

EXT ?

Displays the current default extension.

EXT

Resets the default extension to .COM

DIR FNA

Catalogues the disc. The display can be interrupted by pressing a key; press the space bar to restart, ESC to exit. Each filename is shown with the number of sectors it occupies. Protected files are marked with a 'P'. The number of files is limited only by disc capacity.

```
DIR
DIR "*.BAK"
DIR C
```

LDIR FNA

Catalogues the disc sending the output to the printer.

SEARCH FNA

Checks if a file is on disc. If so, the system variable EF=1, if not 0. If EF is to be used in a logical expression then -EF must be used to achieve the correct logic values (-1=TRUE, 0=FALSE). Ambiguous filenames may be used, when EF=1 if at least one match is found.

```
10 SEARCH "TEST"
20 IF EF=1 THEN LOAD "TEST" ELSE PRINT "File not found"
```

LOADING AND SAVING A FILE

LOAD FN (, A, NE) (,V) (, J) (, N)

Loads a file. Options:

- , **A,** NE loads the file at the address specified in NE
- , **J** joins the file to a BASIC program
- , **N** stops autorun after loading
- , **V** shows the file status, but does not load it.

File status is shown in the format *SSSS FFF SS EEEE*
where *SSSS* is start address of file (no. of records for random access file)
FFFF is end address of file (length of random access file)
SS is the file status byte –

80 = BASIC
81 = BASIC auto
40 = Machine code
41 = Machine code auto
(see APPENDIX on file status byte coding)

EEEE is execute address for a machine code program – default value is 0.

Alternatively the file status can be obtained when loading a file by examining the system variables ST (STart address), ED (EnD address), FT (File Type), and EX (EXecution address). Note that these variables will be cleared if a BASIC program is allowed to run on loading. They may also be corrupted if a file is loaded into the area of memory occupied by the variables. To preserve them, use the *,V* option which will only load in the status byte and not the whole file.

```
LOAD "TEST"
LOAD "B-ROUTINES", J, V
LOAD A$ + ".TXT", A#4000
```

(FILENAME) (, A, NE) (, N) (, J) (,V)

It is unnecessary to use *LOAD* unless the filename is expressed as a variable, e.g. *LOAD A$*. It is enough simply to type the filename to load it – provided the filename does not start with a BASIC keyword or number. If it does, specify the drive, e.g. *A-LOADER*. Note that after *QUIT* is used, the ! must be used as a prefix to the filename.

```
TEST
B-DESIGN.SCR, A#1000
ROUTINE.BAK, V
```

SAVE FN (, **A** NE) (, **E** NE) (,**T** NE) (,**AUTO**)
SAVEO " " " " "
SAVEM " " " " "
SAVEU " " " " "

saves the file with the name FN to disc.

The four save commands only have different effects if the filename already exists on disc. The syntax is as for CSAVE, i.e. the A and E extensions are for machine code only. As on cassettes, it is the status of the saved file which determines whether it autoruns or not. The status can easily be altered subsequently. Using , *T* NE enables an execution address to be specified that is not the file start address. This is not operative for a BASIC program.

If the file already exists:

SAVE gives a 'File already exists' message
SAVEO overwrites the existing file unless it is protected
SAVEM the new file saved is appended to the existing file on disc
SAVEU the existing file is preserved as a .BAK file. Any existing .BAK file is lost.

SAVE "TEST", AUTO
SAVEM "DESIGN.SCR", A#9800, E#BFDF

ESAVE FN

Saves the current screen (TEXT or HIRES). The computer must be in the same mode when the file is reloaded.

ESAVE "DESIGN.HGR"

DELETING AND MODIFYING A FILE

DEL FNA

Deletes an unprotected file. Wild cards are permitted, in which case confirmation is requested for each file. Use ESC to exit.

DEL "TEST.COM"
DEL ".BAS"*
DEL
DEL B

DESTROY FNA

DANGEROUS! As for *DEL*, but with no request for confirmation. *DESTROY* alone will delete all files on disc. If used in error, RESET to salvage as many files as possible.

DESTROY ".BAK"*

DELBAK (drive)

Deletes all .BAK files.

DELBAK
DELBAK B

REN FNA (old) **TO** FNA (new)

Renames a file. Wildcards are allowed, provided they are in the correct corresponding positions.

REN "PROG.DAT" TO "GAME.COM"
REN ".COM" TO "*.BAS"*

STATUS FN (, **A** NE) (, **T** NE) (, **AUTO**)

Allows you to change the status of a file without having to load and save it, following the syntax for *SAVE*. If no option is specified, autorun is removed.

, *A* NE alters the loading start address; the DOS calculates the new end address itself
, *T* NE forces a machine code file to AUTOrun from the address specified in NE
, *AUTO* forces a file to AUTOrun from the start.

```
STATUS "TEST"
STATUS "MENU", AUTO
STATUS "B-PROGMC", A#4000, AUTO
```

PROT FNA

Protects a file from deletion or overwriting. A letter 'P' is displayed by the filename in the directory.

```
PROT "TEST"
```

UNPROT FNA

Removes protection from the file and the 'P' from the directory.

```
UNPROT "*.COM"
UNPROT B
```

INIT (drive , no. of sectors , no. of tracks , **S** or , **D**)

To initialise a disc, follow the on-screen instructions (use ESC to abort). The number of sectors per track and the number of tracks may be specified along with , *S* or , *D* to indicate a single or double-sided drive. The number of tracks per side must be between 21 and 99 inclusive, and the number of sectors per track between 16 and 19 inclusive. Any value outside these parameters will produce an error message. The maximum number of tracks on a 3½" or 5¼" disc is recommended to be 82. Specifying 16 or 17 sectors per track gives the highest reliability and fastest speed. The default value is 17 sectors, single-sided, but this can be preset to a new value, associated with each disc, using the command *DTRACK*.

You are offered the options to format and name the new disc, to enter an initial command statement to run on booting, and to make it a Master disc (with full DOS) or a Slave disc (saving 90 sectors).

INIT *INIT B,17,42* *INIT C,17,80,D*

Several replies are required from the user:

- Format (Y/N): Answer 'Y' to format or 'N' to go directly to the next stage

- Name: Enter a name for the disc. To include attributes in the disc name use CTRL-Z followed by the appropriate code as used with the ESC key in direct mode or CHR$(27) in program mode.

- Init Statement: Enter the instruction(s) to execute on booting the disc. If none, the disc will go straight to BASIC command level

- Master Disc (Y/N) Enter 'Y' for a Master disc, 'N' for a slave disc. A Master disc has a full copy of the DOS, a slave disc must only be used with the DOS loaded, but provides an extra 90 sectors.

NOTE 1: Formatting a disc will erase any existing files on disc. *INIT* affects memory from #3000 to #B0FF, so save any program in memory before using it.

NOTE 2: If asked to insert a Master disc be sure to replace it with the disc being initialised before answering the *Format (Y/N)* request, otherwise you could reformat your master disc if it is not write-protected.

BACKUP (drive) (TO) (drive)

Copies one disc (the source) to another (the target). Affects memory from #0600 to #B4FF. Use identically formatted discs or a 'DISC I/O' error is generated. On a single drive you must exchange the source and target discs as prompted. It is advisable to write-protect the source disc.

BACKUP
BACKUP TO B
BACKUP C

COPY (COPYM or COPYO) (FNA) (TO FNA) (, C) (, N)

Copy files - *M* and *O* work as for *SAVE*.

COPY checks the file is not already on the target disc. If so, a 'FILE ALREADY EXISTS' message is displayed

COPYO overwrites such an existing file, otherwise works as COPY

COPYM permits a merge copy to one file of several files by the use of wildcards - if the target file already exists, the source files are merged to it. If not, a new file comprising all the source files is created.

COPYM "A-.CDE" TO "B-PROG"*

, **C** asks for Y/N confirmation for each file to be copied
, **N** copies files on the source disc itself, omitting the change disc prompts.

With wildcards, two uses exist:

With *COPY* and *COPYO* the source filename can contain wildcards. If so, there must be a corresponding wildcard in the target filename. The various files so specified are then copied with their correct source filenames. The only exception to this is that you may put a wildcard in the source filename and omit a target filename, or compose the target filename entirely of wildcards (e.g. *.*)

With *COPYM* the source filename can contain wildcards, but never the single target filename.

COPY "B-.COM" TO C-*.CDE"* copies all files with the .COM extension on drive B to drive C, giving them the new extension .CDE.

DISC CONFIGURATION

When first receiving your SEDORIC V2.0 Master disc you need to check the configuration of your disc system and correct it if it is set wrongly on the master disc. You can check the configuration by using the *DSYS* command. For instance, if you have two 3" disc drives connected and when you checked, using *DSYS*, your system was set to the following:–

drive A ... 42 tracks 17 sectors, single–sided
drive B ... 82 tracks 17 sectors, double sided

(this happens to be Allan's configuration) then use the *DTRACK* command to set the correct configuration on the master disc (ensure the master disc is not write–protected). This is achieved by entering:–

DTRACK 42, 42 or *DTRACK A, 42, 42*

To configure one double–sided 3½" disc drive it would be:–

DTRACK 80; D

SYS

Displays the DOS configuration in memory (not that on disc).

DSYS (drive)

Displays the DOS configuration saved on the specified disc.

DSYS B

DNAME (drive) AE

Modifies the specified disc name, displaying the old name and then requesting and writing the new one. To add attributes, use CTRL–Z as for the INIT command (see p. 10).

DNAME
DNAME A

INIST (drive) AE

Modifies the initialisation instructions on the specified disc, allowing you to personalise the start–up routines.

INIST A
INIST B,"CLS:PAPER4:INK6:MENU"

DKEY (drive) (, **A**) (, **S**)

Sets the disc start-up keyboard configuration - QWERTY if no option is specified, AZERTY with , *A* , with accents with , *S* .

DKEY B, A
DKEY, S

TRACK (NE_A) (, NE_B) (, NE_C) (, NE_D)

Modifies (only in memory) the disc drive configuration by setting the number of tracks for each drive. A value of "0" disables the drive, which then on access will give a 'DRIVE NOT IN LINE' error. To configure a double-sided drive as single-sided, add "**;S**" to the number of tracks.

TRACK 41,,42; D sets drive A to 41 tracks and drive C to 42 tracks double-sided

DTRACK (drive) (NE_A) (, NE_B) (, NE_C) (, NE_D)

As the above command, but the change is made on disc and not the DOS in RAM. The format is then loaded on each initialisation.

DTRACK A, 42

DNUM (drive) (, NE) (, NE)

Modifies on the disc the default initialisation values for respectively the first line and step values for the renumbering commands NUM and RENUM and for auto line numbering.

DNUM B, 100, 10

SYSTEM (drive)

Specifies the disc drive on which is the DOS containing transient command files. The default drive is A.

SYSTEM D

BASIC PROGRAMMING AIDS

RENUM (NE_1) (, NE_2) (, NE_3) (, NE_4)

Renumbers all or a block of a BASIC program. The parameters should be entered in the following order:
- NE_1 new first line number
- NE_2 line increment required
- NE_3 and NE_4 the first and last line numbers of the program to be renumbered.

If any values are not declared the default values are:

Start line number	=	100
Increment	=	10
First/last line	=	the first/last lines of the existing program.

The values for the origin and step can be modified with the *DNUM* command. All GOTO, GOSUB, ON GOTO, ON GOSUB, THEN, ELSE, RUN, and RESTORE commands are correctly modified.
Numeric expressions (e.g. GOTO 100*A) are NOT modified. Use ON GOTO or ON GOSUB instead.

RENUM	=	*RENUM 100, 10*
RENUM1000	=	*RENUM 1000, 10*
RENUM,,,10000	=	*RENUM 100, 10, 0, 10000*
RENUM1000,,2300	=	*RENUM 1000, 10, 2300*

RENUM100,10,0,10000 is the full command.

Note: This command affects only the actual line number. It does not necessarily reposition the block of lines in memory, e.g. RENUM 100,10,20,30 gives:

10 REM EXAMPLE	10 REM EXAMPLE
20 PRINT "TEST"	100 PRINT "TEST"
30 PRINT "EXAMPLE"	110 PRINT "EXAMPLE"
40 END	40 END

MERGE FN (, L)

Merges the BASIC program in memory and the BASIC program specified. If the resultant program is too large for the available memory an 'OUT OF MEMORY' error will be generated. All variables are preserved, but as usual any functions defined by DEF FN are lost. If the same line number appears in both programs, the one in memory will be used and a 'LINE ALREADY EXISTS' message displayed. The command can be speeded up by omitting the on-screen information if you add , *L*.

If using MERGE within a program, ensure no line will be inserted before the line containing the MERGE command, otherwise a 'SYNTAX ERROR' is generated.

MERGE "PROG"
MERGE "TEST", L

DELETE (NE_1) - (NE_2)

Deletes a block of lines from a program without losing variable values. The syntax is as for LIST, but the "-" is mandatory. Without it DELETE is equivalent to a NEW!
If using DELETE within a program, ensure the command is not placed in or after the block to delete. User functions defined by DEF FN located in the deleted block will be lost.

DELETE 100-100 deletes line 100
DELETE -1000 deletes all lines up to line 1000
DELETE 234-987 deletes lines 234 to 987

SEEK (AE) (, S) (, M)

SEEK AE lists all program lines in which the string AE exists. The listing can be interrupted with SPACE or CTRL C as usual. The string must not include the ASCII nul code (0), otherwise an 'INVALID STRING ERROR' is generated.

To search for BASIC keywords AE must contain the token value rather than the text equivalent (e.g. #80 for END). The string may include wildcards; the wildcard character is '£' because of the frequent use of '?' in BASIC.

SEEK CHR$(#80)

The maximum string length is 78 characters.

Option , *S* omits the screen listing, giving simply a total number of occurrences. Option , *S, M* omits any screen information. In each case the system variable SK contains the number of occurrences.

SEEK alone (used after a *SEEK* AE) displays each line occurrence one at a time for each entry of the command. The prescribed string remains in memory until another *SEEK* AE or a RESET.

CHANGE AE_1 TO AE_2

Replaces all occurrences of chain AE_1 by the chain AE_2 in a BASIC program. As for SEEK, the strings must not contain the ASCII Nul code (0). The maximum length of the command is 78 characters. The "£" is the wildcard character.
Note that use of the wildcard in the target string will be treated literally. If the string exchange would result in a line being too long then the error message 'LINE; XXXX ?STRING TOO LONG ERROR' is displayed.

CHANGE CHR$(#BA) TO CHR$(#8F) changes all PRINT commands to LPRINT.
CHANGE "ORIC" TO "ORIC ATMOS"

NUM (NE_1) (, NE_2)

Sets the parameters for automatic line numbering. NE_1 is the start line, and NE_2 the increment. Each press of FUNCT + RETURN ends the current line and numbers the next.

The default values are start line 100, increment 10. They can be modified by *DNUM*.

NUM 1000 numbers in tens from line 1000
NUM, 5 numbers by fives from line 100
NUM 3300, 20 numbers by twenties from line 3300

NUM END

If this command is entered, the DOS automatically finds the number of the last line of a BASIC program already in memory, and starts automatic line numbering at the line following.

KEY SET or KEY OFF

Inhibits (OFF) or authorises (SET) the keyboard. Switching the keyboard off speeds up a program by 20%.

Note: Beware of using KEYOFF in direct mode. Do not use a WAIT command after KEYOFF, since WAIT uses the system clock which is no longer updated after KEYOFF. Use a FOR...NEXT loop instead.

KEYIF NE GOTO or KEYIF NE THEN

Seeks a keypress set by the argument NE. Works even if the keyboard is inhibited or several keys are pressed together. The syntax is as for an IF THEN ELSE.

NE is a special code, listed in the APPENDIX at page 46.

100 KEYIF #84 THEN SHOOT
...
200 KEYIF AB THEN 100 ELSE PING

QWERTY

Sets the keyboard to normal (usually after an *AZERTY*) and executes an *ACCENT OFF*. Inoperable in Hires mode.

AZERTY

Sets to French keyboard –

Q	becomes	A
W		Z
A		Q
Z		W
M		;
;		M

N.B. Does not work on the Oric–1.
Also effects an *ACCENT SET*.
KEY$ is not affected – it reacts to the QWERTY keyboard even after an *AZERTY* command.
CTRL key combinations follow the AZERTY key setup, but function key commands remain in QWERTY setup.

Cannot be used in Hires mode.

ACCENT SET / ACCENT OFF

Redefines certain characters to show accents on screen. They have been chosen to reflect the ASCII characters used by most printers for the French character set accents.

ASCII	NORMAL	ACCENTED
40	@	à
5C	\	ç
7B	{	é
7C	¦	ù
7D	}	è
7E	■	ê

The command is not usable in Hires mode, but the character set will be preserved when selecting HIRES from TEXT mode. *ACCENT OFF* resets *ACCENT SET*.

FUNCTION KEY SET-UP

A function can be assigned to any key in two ways. The first is by the allocation of a pre-determined command using KEYDEF, the second the allocation of a command defined by the user utilising KEYDEF and KEYUSE.

KEYDEF NE

Permits the definition of function keys. The argument is the function code. After entering the command, press the key to which you wish to assign the function. If you press just the key, the function is accessible by pressing FUNCT and the key. If you press SHIFT and a key, the function is accessible by pressing FUNCT + SHIFT and the key.

The function codes are:
- Codes 0 to 15 for user-definable commands (with *KEYUSE*)
- Codes 16 to 31 for special keywords (see Appendices)
- Codes 32 to 127 for the DOS keywords (see Appendices)
- Codes 128 to 246 for the BASIC keywords (see the list in the Oric manual)
- Code 254 for FUNCT+DEL (delete keyboard buffer)
- Code 255 for FUNCT + RETURN (auto line numbering)

KEYDEF 0 assigns to the key you then press the command defined by *KEYUSE 0,"xxxx"*

KEYDEF #80 assigns to the key pressed the BASIC keyword END (#80 is the hex. code of END)

KEYUSE NE, AE

Only works on the Atmos. It permits the definition of the 16 user commands (with one or more commands each).

NE is the command code (0 to 15), AE is the command chain (the maximum number of characters is 16).

The chain can include all ASCII characters except 0 (null) and characters above 127. It can include CHR$(13) which simulates a RETURN.

KEYUSE 0, "PAPER 0:INK 7" + CHR$ (16) + CHR$ (13)

An example of setting the '1' key to list all the filenames with the extension ".COM" is set out in the three steps that follow.

1 Enter *KEYUSE 0, "DIR" + CHR$(34) + "*.COM" + CHR$(34) + CHR$(13)*

2 Enter *KEYDEF 0*

3 Press key '1'

Now if you press the FUNCT key with the '1' key, the files will be listed to the screen.

KEYSAVE FN

Saves the current function key configuration on disc. Equivalent to *SAVE FN, A#C800, E#C97F.* To recall a saved configuration, simply load it.

Note: KEYSAVE automatically behaves like SAVEU in that a backup file is created if you save a file to a disc having a file of the same name on it.

KEYSAVE "TEST.KEY"

VUSER

Displays the defined command chains for the 16 user commands.
Control characters appear in inverse video - e.g. M inverse for RETURN because CTRL M is equal to RETURN.

Note: This does not show to which keys the user commands are allocated. This would be suitable for an upgrade if it is required by users.

TKEN AV

Not usable in direct mode.
Encodes the string variable as a suite of BASIC instructions which can be executed by the use of *STRUN*.
The variable must contain the string to encode; after execution it contains the encoded string.
The string variable can contain up to 79 characters.

TKEN A$
TKEN "PING"

UNTKEN AV

The inverse of *TKEN*. Decodes the encoded chain into its original form.

STRUN AV

Executes a chain encoded with *TKEN* as if it were a BASIC line.
This allows the easy execution of formulae, or the creation of a command file.

All BASIC and DOS keywords except those normally not usable in direct mode may be included. Any string to be executed several times only needs to be tokenised once since it is stored in the nominated string variable.

INSTR AE_1 , AE_2 , NE

Searches for the first occurrence of the string AE_2 within the string AE_1 , starting at the character position specified in NE.
The position is returned in the system variable IN. If the string is not present, IN contains 0.
String variables may be used

10 INSTR "NICE DAY", "DAY", 1
20 PRINT IN
gives the result 6.

ERROR HANDLING ROUTINES

ERR SET or ERR OFF

So that the programmer can handle errors generated by the DOS the *ERR SET* command prevents the interruption of a program by a DOS error. Note that the handling of errors generated by the BASIC interpreter cannot be handled by this method.

ERR SET is used in conjunction with the command *ERR GOTO*. When the DOS detects an error, the program continues to operate from the line specified by *ERR GOTO*, or will stop if the program doesn't provide such an error routine. The *ERR SET* must appear in the program before the *ERR GOTO*.

The variable EN at #4FD will contain the error number, and EL at #4FE–#4FF the number of the line causing the error. In this way, for example, the programmer can trap DOS errors generated by having the incorrect disc in the drive when looking for a specified data file. The error number 01 would provide evidence of this so the programmer could use an IF statement to print instructions to insert the correct disc. If an error is generated in direct mode, the line number will be given as 65635.

ERR GOTO NE

Specifies the line number where DOS errors are handled provided an *ERR SET* has been executed.

N.B. – an *ERR SET* nullifies any previous *ERR GOTO*.

RESUME (NEXT)

Restarts execution where it was interrupted (*RESUME*) or at the following line (*RESUME NEXT*). This command is used within the program section that handles the error condition.

ERROR NE

Generates an error condition with the specified number held at location #4FD. This allows the centralisation of errors, including user-created ones.
NE may range from 50 to 255. Numbers 1 to 49 are reserved for DOS errors.
If DOS error handling is enabled (by *ERR OFF*) then a 'USER XXX ERROR' message is generated, where XXX is the error number specified within the program.

LINPUT (@NE, NE,) (AE,) NE; AV (, **E**) (, **S**) (, **C**) (, **J**) (, **K**)
LINPUT (@X, Y,) (Character,) Length; AV...

An instruction to format text input – not numeric variables. It allows entry of text of the length specified, which is assigned to the string variable AV.

This is a very powerful instruction which needs to be used progressively to realise its full potential. The simplest instruction is the form *LINPUT (length); AV*, it is an INPUT where you can select the number of characters to be entered.

The command creates a window on screen, which utilises a full page editor within the window. When inputting text using *LINPUT* only certain control characters are valid: CTRL T, CTRL N, CTRL Z (equivalent to ESC for attributes) and DEL. The cursor is automatically enabled.

The maximum window length is 255 characters.
Input can be terminated by ESC or RETURN. The system variable OM (Out Mode) varies with the method of exit from the window:

0 = exit by RETURN
1 = ESC
2 = cursor left
3 = cursor right
4 = cursor down
5 = cursor up
6 = automatic exit (window full)

, E prevents erasure of the window before any text has been entered
, S permits exit by use of the cursor arrows
, C allows exit from the window when it is full without having to press RETURN. Without this option the cursor returns to the start of the window
, K justifies the text in the screen window by inserting spaces, the variable does not contain the spaces
, J the same as for , *K* except the variable is justified, but the screen unaffected.

The first character of AE fills the window if overwriting is required. If an empty chain is set, the character is "." by default.

@ *X, Y* permits you to fix the position of the window on screen, as for PLOT.

```
LINPUT 20; A$, S
LINPUT "-", 100; A$
LINPUT @10, 3, 100; A$, C, J
LINPUT @10, 3, "-", 2; A$
```

CREATEW (FN)

Creates a screen area of 25 lines by 40 characters, starting at screen line two, for use by the *WINDOW* command. The command is not usable in Hires. A full screen editor is used within the screen area.
CTRL S saves the screen page, use CTRL C to leave it.

Data fields are defined by CTRL W, appearing as a square on the screen. The number of fields is unlimited, the field length is limited to 255 characters. The screen area can be thought of as a proforma for the entry of data into programs.

CREATEW, "MASK.SCR"

WINDOW (FN)

Not usable in Hires. If no filename is used (as saved by *CREATEW*) then the current page is loaded, if possible.

Before using this command, you must correctly dimension the array WI$ with the number of fields. If not, the default of 11 fields is used.

WINDOW displays the page input to the screen along with the data fields and their values found in the array WI$. The data is correctly justified on screen.

During data entry, the user can only write in the fields, or move from one field to another with the cursor arrows or RETURN.

On typing CTRL C the fields are read into WI$, in the order right to left, top to bottom. The number of fields allowed are unlimited, but WI$ must be dimensioned correctly.

A field can be 255 characters long.

USING NE, AE (, AV)

This command formats numbers printed to the screen; the number is the numeric variable NE, its format is defined in the string variable AE. If an alphanumeric variable is specified (,AV), the number formatted is not displayed, but the variable specified is affected accordingly.

The formats are given below; all other characters are unaffected:

+	displays the sign of the number, + or –
–	displays a space or the – sign if a negative number
↑	displays the exponent in scientific notation (e.g. +12)
&a	is the character to replace 0's in front of a whole number, the default is spaces
%x	is from 0 to 9, displays x characters of the whole number
#x	is from 0 to 9, displays x characters of the decimal part
!x	is from 0 to 9, rounds to x characters of the whole part
@x	is from 0 to 9, rounds to x characters of the decimal part.

10 A=1000/6 : USING A, "&0Number:%9.#4 E↑"
displays Number: 000000166.6666 E+00

30 USING A, "@2+%5. Pounds and #2 pence."
displays 1666 Pounds and 67 pence

40 USING PI, "&%2 #5", A$: PRINT A$*
displays *3.14159

LUSING NE, AE (, AV)

As for *USING*, but outputs to the printer.

LUSING 12, "+–↑#3"

WIDTH (NE) or **WIDTH LPRINT** (NE)

Sets the usable screen width (*WIDTH*) or printer width. NE is the number of characters per line to be printed before a carriage return (CR) and line feed (LF) are automatically generated.

On the Oric–1 *WIDTH* controls both screen and printer. Default values are 40 screen/80 printer on the Atmos, and 53 screen/93 printer on the Oric–1.

WIDTH 20 gives a display 20 columns wide
WIDTH LPRINT 132 gives a printout 132 columns wide

PRINTER OUTPUT

OUT NE

Sends an ASCII code to the printer and is the equivalent of LPRINT CHR$(NE).

OUT 27: OUT 33: OUT 30 equals LPRINT CHR$(27);"!";CHR$(30)

PR SET or PR OFF

Switches the printer in or out. After a *PRSET* all screen output is directed to the printer.

Note: *PRSET* treats all PRINTs as LPRINTs. *PROFF* does not treat LPRINTs as PRINTs!

The printer must be disabled before executing a STOP or END in an Atmos program, or using CTRL-C. Otherwise a bug in the Atmos ROM sets the screen width incorrectly. A WIDTH command could be used to correct the fault. However this bug can be put to good use if you require double line spacing on your screen.

GRAPHICS INSTRUCTIONS

These commands are constructed around the variable AN, representing the angle to be used. AN is expressed in degrees anticlockwise from EAST. Clockwise values may be expressed as a negative number of degrees.

LINE NE, FB code

Draw NE points in the current direction, i.e. the current value of AN. FB code is as usual,

FB=0 for current paper colour
FB=1 for current ink colour
FB=2 for inverse.

```
10 HIRES
20 CURSET 0,0
30 AN=-45
40 LINE 100,1
```

draws a line from the top left hand corner of the screen 100 pixels long and at an angle of 45 degrees below the top of the screen (i.e. 45 degrees below (minus) EAST).

BOX NE_1, NE_2, FB code

Draws a rectangle, with the current cursor position forming the top left corner. It will be angled to the current value of AN.

NE_1 and NE_2 are the sides of the rectangle in pixels, across then down. FB code is as for *LINE*.

LCUR

Returns the horizontal and vertical coordinates of the cursor in Text mode in the variables CX and CY.

HCUR

As above for the Hires screen.

USER x , DEF address (, **O**)

USER x (, **A** NE_1) (, **X** NE_2) (, **Y** NE_3) (, **P** NE_4)

x may be NV, a numeric variable.

Allows the call of assembly language sub-routines, with the ability to pass parameters.
x or NV is the number of the user routine and must be in the range 0 to 3.

USER x , DEF is used to define the routine execution address. If the *, O* option is specified, the routine will execute in RAM overlay.

USER x executes the routine at the specified address. The 6502 registers are loaded with the specified parameters (which are optional).

On returning from the routine, the variables RA, RX, RY and RP contain the values of the registers A, X, Y and P.

USER 2, DEF #CCB0
USER 1, DEF #4000,0
USER 2, X12, Y20
USER A, A100, X200, P#C1

] (right bracket)

As for the "!" in non-disc systems, used to call machine code routines from BASIC. DOKE addresses #2F9-#2FA with the call address

N.B. After a *QUIT*, the exclamation mark is obligatory to execute DOS commands.

QUIT

Resets the pointers used by the DOS, resetting the IRQ and NMI vectors. Disables the FUNCTion keys, and makes the use of "!" obligatory for DOS commands.

The instruction is necessary before running any program that uses Page 4 of memory or modifies the IRQ/NMI vectors itself.

RESET

A command having the same effect as the button on the disc drive.

RESTORE (No. of line)

Places the DATA pointer at the start of the program or at the specified line (which does not have to exist). Only RESTORE in capitals is treated by *RENUM*.

MOVE AE_{start} , AE_{end} , $AE_{target\ address}$

Moves a block of memory. The inclusive start address and end address+1, and target start address must be specified. Addresses above #C000 are in RAM overlay where the DOS is situated. Beware of corrupting this area.

MOVE #A000, #BF40, #1000 saves the HIRES screen at #1000.
MOVE #1000, #2F40, #A000 recalls it

OLD

Recovers a BASIC program lost by a BOOT or a NEW. Clears all variables.

RANDOM NE

Initialises the random number generator. If no argument is specified, the initialisation is left to chance.

RANDOM 15

SWAP Variable, Variable

Exchanges the contents of the specified variables (of the same type).

DATA FILES

SEDORIC DOS is capable of both sequential and random access to data files on disc. Each file has a logical number (LN) assigned to it to simplify the command syntax. The number may be from 0 to 63. It is necessary to open a file before accessing it, i.e. to allocate to it a logical number and associate it with a filename on disc. The DOS then loads all necessary information and creates the file on disc if it does not already exist. You can then use the file and access the data. On finishing with the file it is essential that it is closed.

N.B. if a 'WRITE-PROTECTED' or 'DISC FULL' error appears, it is advisable to close and then re-open the file for further access to ensure compatibility between disc and memory.

SEQUENTIAL FILES

In sequential access of files the data has to be read in the order in which it was written. Sequential operation is as if a pointer was moving along the file, advancing item by item. Reading or writing of data is carried out at the current pointer position. To enhance use of sequential files, instructions are provided to control the pointer position.

When writing to a file, data is immediately written to disc to minimise the risk of loss or corruption.

OPEN S, FN, LN

Opens a sequential file, reserves a memory buffer, and puts the pointer at the start of file.
OPEN S, "DISCS", 2 opens a file called "Discs" with the logical number 2.

If the file does not exist on disc, it will be created.

CLOSE (LN, LN, ...)

Frees the buffer reserved by *OPEN* and closes any open files.

CLOSE 34, 1 closes files 1 and 34
CLOSE closes all open files

PUT LN, list of variables

Writes the variable contents to a file.

PUT 2, A$, "TEST", 12, A% writes in file 2 the variables A$, TEST, 12 and the numeric variable A%

File size is only constrained by the free disc space. When writing to the middle of a file, the variable must be of the same type as that already written to the pointer position. With string variables, a string shorter than the previously-written string is packed with spaces to the right; a longer string is truncated from the right to the length of the previously-written string.

TAKE LN, list of variables

Reads from file LN the specified variables. The variables must be of the same type as those stored on the disc. If the end of file is reached during a *TAKE* operation an error message is generated.

APPEND LN

Places the file pointer at the end of the specified file.
APPEND 2 places the pointer at the end of file no. 2

REWIND LN

Places the file pointer at the start of the specified file.

REWIND 3

JUMP LN, number of data items

Moves the file pointer the number of items specified. If the end of file is reached, acts as *APPEND*.

JUMP 2, 200
*JUMP 12, A*12*

BUILD LN

Enables you to build a sequential file by keyboard entry. The characters are entered on file as strings of 200 characters. After each 200 characters the string is saved to disc automatically.
Exit with CTRL C, when the characters input to the current 200 character string are saved to disc.
Code ASCII 13 (CR or RETURN) is automatically replaced by the code 13/10 (i.e. a line feed is added) to improve legibility when read with the TYPE command.
All characters are faithfully reproduced on file, which can be useful for animation on the text screen.

BUILD 2

TYPE LN

Allows you to display the contents of a sequential file, starting at the current position of the file pointer.

Pressing a key will stop the listing, Space to restart, CTRL C to exit.

Can be used with *BUILD* for animated sequences.

```
TYPE 2
```

LTYPE LN

As for *TYPE*, but on the printer.

```
LTYPE 2
```

&(LN)

Returns 0 (false) if at the end of file, and -1 (true) otherwise.

N.B. the brackets are part of the command syntax, not an option.

```
1000 A=&(2)
1010 PRINT A
```

&(-LN)

Returns the type of the next data item to read.
The code returned is:

0 for a numeric variable
128 for a string variable
-1 for end of file.

```
A=&(-2)
```

RANDOM ACCESS FILES

Random access is very different from sequential access in that the data is organised within the FILE into RECORDS. Each record contains all the relevant information on the item recorded, and each has its own identity so that it can rapidly be accessed on disc.

Each record contains all relevant information grouped in FIELDS.

The record is the entity that is transferred from the disc to memory and vice-versa. To facilitate file operations, the commands work on a memory buffer holding the whole of the relevant record.

There are two types of instructions, those concerned with transfer between disc and buffer, and those with transfer of variables to and from the buffer.

OPEN R, FN, LN (, record length, number of records reserved)

Opens a file. It is essential that you specify the record length and number of records when first opening a file.

If the number of records exceeds the initial setting, extra space is automatically created for extra records.

The space occupied by a random access file is indicated in the number of sectors used on disc (i.e. number of records x record length / 256). A file of 300 records of 50 characters each would occupy about 60 sectors.

OPEN R, "DIRECT", 2, 183, 100

opens a file called "Direct", logical number 2, with each record a maximum 183 characters, and with space for 100 records reserved.

CLOSE (LN, LN ...)

Closes the file numbered LN (or all files) and frees the buffer.

CLOSE

TRANSFERS BETWEEN BUFFER AND DISC

TAKE LN, Record no.

Loads the specified record from disc into the buffer.

TAKE 2, 120 (i.e. No. 120 of logical file number 2)
*TAKE F, JP*12*

If the record does not exist a 'BAD RECORD NUMBER' error will be generated.

PUT LN, Record no.

Transfers the specified record from the buffer to the file on disc.

If the record does not exist, it will be created.

PUT 2, 120

WORKING ON THE BUFFER

To facilitate the classification of data, the record can be further divided into fields, enabling the place within the record where the data will be held to be defined. The fields are rather like variables; you may give them a value and read them as with variables. However be warned, they are not variables!

The conventions for field names (FLN) are :

- 5 significant characters
- can contain a pseudo array by indexing the field
- the field name with index 0 is equivalent to the name without an index, i.e. if a field is called NAME, NAME(0) is the same as NAME
- must not include a BASIC keyword or start with a DOS keyword
- the total number of fields is only limited by the available memory.

E.g.: *NAME*
INDEX (1)
ADDRESS1 (0) equals *ADDRE (0)*, only the first five characters are significant.

FIELD LN, FLN TO type (,....)

Defines fields within the each record of the file. The type of field is defined by:

$NE	for an alphanumeric field, NE defines its length
%	for an integer field
!	for a numerical field
letter **O**	for a field of one byte

If the command ends with a comma, the next definition starts at the current position. The length of a field must not exceed the total length of the record.

An integer field is 2 bytes long, a numerical 5 bytes, and an alphanumeric the length specified. To that is added 2 bytes for internal working space.

FIELD 2, NAME TO $8, SURNAME TO $12, AGETO %, SALARY TO !, SEX

opens in file number 2 a field for the name of 8 characters, one of 12 characters for the surname, an integer field for the age, a numeric field for salary, and a single byte field for the sex.

Total length of record: 2+8 + 2+12 + 2+2 + 2+5 + 2+1 = 38 bytes per record

BUG: Note the bug and solution dealt with on page 49.

TRANSFERS BETWEEN FIELDS AND VARIABLES

RSET FLN < Expression

Writes the given value to a field – both must be of the same type.
String variables will be justified to the right, and stripped at the left if the chain is too long for the field, any gap to the left is filled with spaces

RSET NAME(2) < "EUREKA" (NAME(2) is an alphanumeric field)
RSET TEST < 123 (TEST is an integer field)

The '<' is part of the syntax of this command.

If the field CHAIN has five characters:

RSET CHAIN < "123456" stores "23456"
RSET CHAIN < "TOTO" stores " TOTO"

LSET FLN < Expression (,.....)

As *RSET* for numeric and integer fields.

String variables are justified to the left and stripped at the right if too long, or completed at the right by spaces if too short.
*LSET TEST < 2000*AS*

If the field CHAIN has five characters:

LSET CHAIN < "TOTO" stores "TOTO "
LSET CHAIN < "123456" stores "12345"

FLN > Variable

Read field of name FLN and put into the specified variable, which must be of the same type.

NAME(1) > A$
TEST > A
PROPERTY > A is invalid because *PR* is a DOS keyword.

& (LN)

Display the number of records in file number LN

& (12)

DISC ACCESS

There is a special kind of file which allows you to modify the disc, sector by sector, directly from BASIC. It enables the experienced programmer to create a filing system particular to the needs of the application. The idea is similar to the instructions for working on the buffer files, but the fields are here defined without the 2 bytes working space.

Instead of the record being in the buffer, the buffer now contains the contents of the disc sector. The buffer is 256 bytes long.

To ease things, instructions are included to search for empty sectors on the disc.

OPEN D, FLN (,Drive)

Reserves the buffer of 256 bytes and opens the file FLN. Note that the space between OPEN and the D is obligatory to avoid the BASIC keyword END.

OPEN D, 1
OPEN D, 1, B

CLOSE (LN (, LN.....))

Closes the file(s) specified and frees the buffer.

TAKE LN, Track, Sector (, Drive)

Reads the specified sector into the buffer. No test is carried out to see if the specified sector exists. If not, a 'DISC I/O (ERROR number 04)' message is generated.

TAKE 2, 14, 3, A
TAKE 0, 20, 1

PUT LN, Track, Sector (, Drive)

As TAKE, but used to write, not read the sector.
For a double-sided drive, add 128 to the track number to access side 2 of the disc:

PUT 1, 131, 2 writes the buffer in sector 2 of track 3 on Side B

SECTOR RESERVATION

It is possible to search for free sectors on the disc, as well as to free certain sectors. All this information is stored in special sectors (Track 20, Sectors 2 & 3) called **BITMAP**. To reserve a sector it is necessary to read BITMAP, determine the free the sector using *CRESEC*, and rewrite BITMAP.

PMAP Drive

Reads BITMAP into memory.

PMAP
PMAP A

SMAP Drive

Writes to disc the BITMAP in memory.

Between a *PMAP* and an *SMAP* you must not do a *LOAD*, *SAVE* or *DIR*, or make direct or sequential access to the disc, otherwise BITMAP may be corrupted.

CRESEC

Returns in the system variables FP and FS the track and sector address of a free sector. If the disc is full, it says so.

It is only usable after a *PMAP*.
Automatically decrements the number of free sectors.

FRSEC NE_{track}, NE_{sector}

Frees the specified sector and increments the number of free sectors. If already free, it has no effect.

Only usable after a *PMAP*.

FRSEC 20, 10

INDEX OF DOS KEYWORDS

APPENDIX – DOS Keyword Addresses and Codes

Command	Code	Entry Address	Block	Command	Code	Entry Address	Block
ACCENT	35	EB91		LUSING	80	F036	
APPEND	33	FE07		MERGE	88	F13C	3
AZERTY	34	EBDE		MOVE	87	F136	1
BACKUP	37	F151	2	NUM	89	EB25	
BOX	36	F0DE		OLD	91	E0AF	
BUILD	38	FEE0		OPEN	92	FA50	
CHANGE	39	F148	3	OUT	90	E71F	
CLOSE	40	FB8D		PMAP	96	F990	
COPY	41	F157	4	PR	95	E7C0	
CREATEW	42	DE4D		PROT	94	E6D0	
CRESEC	43	F9BC		PUT	93	F9CB	
DEL	48	E446		QUIT	97	E7F5	
DELBAK	47	E437		QWERTY	98	EBE1	
DELETE	45	F142	1	RANDOM	105	E796	
DESTROY	46	E444		REN	103	E537	
DIR	49	E344		RENUM	102	F14E	1
DKEY	53	F124	5	RESET	100	E7B8	
DNAME	52	F145	5	RESTORE	106	E7D9	
DNUM	51	F12A	5	RESUME	99	E9BB	
DSYS	54	F127	5	REWIND	101	FABB	
DTRACK	55	F139	5	RSET	100	FC75	
ERR	60	E97F		SAVE	117	DD50	
ERRGOTO	57	E999		SAVEM	115	DD4A	
ERROR	59	E9B0		SAVEO	116	DD53	
ESAVE	61	DDE0		SAVEU	114	DD4D	
EXT	62	E9ED		SEARCH	118	E5FC	
FIELD	63	FBBF		SEEK	109	F154	3
FRSEC	64	F99C		SMAP	120	F996	
HCUR	65	EBF5		STATUS	113	E62E	
INIST	68	F12D		STRUN	111	E853	
INIT	66	F169	6	SWAP	108	EA3B	
INSTR	67	EC2E		SYS	119	F159	5
JUMP	69	FE12		SYSTEM	112	E702	
KEY	76	E70B		TAKE	122	F8DF	
KEYDEF	73	D9FD		TKEN	121	E89D	
KEYIF	71	DA20		TRACK	124	F130	5
KEYSAVE	75	DDCD		TYPE	123	FE98	
KEYUSE	72	D9B0		UNPROT		E6D3	
LCUR	86	EBEC		UNTKEN	126	E8E1	
LDIR	84	E7D0		USER	125	EA7F	
LINE	77	F079		USING	127	EE99	
LINPUT	82	EC94		VUSER	129	F121	5
LOAD	83	DFF7		WIDTH	130	E740	
LSET	78	FC73		WINDOW	131	F21C	
LTYPE	85	FE95		]		EC04	

The block reference refers to the external blocks of the DOS that have to be loaded when the command is used for the first time.
The entry point addresses (in hexadecimal notation) can be used in conjunction with the *MOVE* command to examine the DOS and customise it if necessary.

APPENDIX – Error Messages

01 – FILE NOT FOUND
The file has not been found on the disc (the file may not exist, the filename may be mis–spelt or a wildcard misplaced).

02 – DRIVE NOT IN LINE
A specified drive is not connected. Check the drive number. To connect it use the *SYS* command.

03 – INVALID FILE NAME
The filename contains invalid characters or is too long (9 characters maximum).

04 – DISC I/O
The disc is corrupted. The DOS displays the sector and track where the error was detected to assist experienced users to recover the disc (e.g. using the NIBBLE disc sector editor). Note: The sector number is not significant, nor is it displayed, if an error occurs during formatting of a disc.

05 – WRITE PROTECTED
Indicates an attempt to write to a disc which has the write–protect tab set.

06 – WILDCARD(S) NOT ALLOWED
A wildcard has been used in a filename in connection with a command that does not accept wildcards (e.g. *LOAD*).

07 – FILE ALREADY EXISTS
An attempt has been made to save a file on a disc which already contains a file of that name.

08 – DISC FULL
The file is too long to be saved in the available disc space.

09 – ILLEGAL QUANTITY
A parameter has been specified that is too large or too small.

10 – SYNTAX ERROR
A mistake has been made in typing an instruction.

11 – UNKNOWN FORMAT
An attempt has been made to use a disc that has not been formatted by SEDORIC DOS.

12 – TYPE MISMATCH ERROR
An attempt has been made to match one type of variable or field against another, such as a numeric against a string variable.

13 – FILE TYPE MISMATCH
A file of one type has been specified with a name associated with a file of another type (e.g. attempting to load data with the *LOAD* command).

14 – FILE NOT OPEN
An attempt has been made to access a file that has not been opened.

15 – FILE ALREADY OPEN
An attempt has been made to assign a logical number to a file when that number has already been assigned to another file.

16 – END OF FILE
An attempt has been made to read data when the end of the file has been reached.

17 - BAD RECORD NUMBER
The record number entered exceeds the capacity set for the file.

18 - FIELD OVERFLOW
The field definitions have produced an overall length that is greater than the length specified for the record.

19 - STRING TOO LONG
The string entered is too long (see *TKEN*, etc.).

20 - UNKNOWN FIELD NAME
An attempt has been made to read from or write to a field not previously defined with the FIELD command.

APPENDIX - Function Key Codes

By means of the command *KEYDEF*, these codes allow any of these command sequences to be assigned to any key on the keyboard. The number of each sequence is that used as the argument (NE) for *KEYDEF*.

Code	Definition
000 to 15	User defined
016	HIRES + RETURN
017	TEXT + RETURN
018	LIST + RETURN
019	RUN + RETURN
020	LPRINT + RETURN
021	FOR I = 1 TO
022	CURSET 120,100,1 + RETURN
023	CTRL-A six times
024	EXT + RETURN
025	NUM END + RETURN
026	SEEK + RETURN
027	RENUM + RETURN
028	OLD + RETURN
029	DIR + RETURN
030	CTRL-T or CHR$(20)
031	© or CHR$(96)
032 to 127	DOS keywords (see page 38)
128 to 253	BASIC keywords (see Atmos manual)
254	DEL
255	Generate line numbers.

APPENDIX - System Variables

These variables are used by the DOS in conjunction with the relevant command or error condition. They can be read from BASIC programs, but should not be assigned new values from within the program if the associated DOS command is to be used, except for the AN variable.

Used by SEARCH:

EF (Existing File) EF=1 if the file exists, 0 if it doesn't.

Used by LOAD or direct loading:

ST (Start address) contains the start address of program
ED (EnD address) contains the end address
FT (File Type) contains the program type
EX (Execution address) of the program.

Note that these variables will be cleared if a BASIC program is allowed to run on loading. they may also be corrupted if a file is loaded into the area of memory occupied by the variables. To preserve them, use the *,V* option which will only load in the status byte and not the whole file.

Used by SEEK:

SK (SeeK) contains the number of string occurrences sought.

Used by INSTR:

IN (INstring) contains the sub-string position in the string

Used by the error handling routines:

EN (Error Number) displays the DOS error number
EL (Line number) displays the line where the error is.

```
100 ERRGOTO 1000
.....
1000 PRINT "Error"; EN, "in line"; EL
```

Used by LINPUT:

OM (Out Mode) returns the mode of exit from entering text into the window buffer.

Used by WINDOW:

WI$ is an array containing the number of fields used within the specified window. Its default is 11 as in BASIC.

Used by LINE and BOX:

AN (ANgle) contains the angle in degrees used in the commands LINE or BOX. Default is 0 degrees. This must be written to from a BASIC program.

Used by HCUR and LCUR:

CX contains X coordinate of text or graphic cursor
CY contains Y coordinate of cursor.

Used by USER:

RA (Register Accumulator)
contains the value of the A register of the 6502
RX (X Register)
contains the value of the X register
RY (Y Register)
contains the value of the Y register
RP (Processor status Register)
contains the value of the Status register

Used by CRESEC:

FP (Free Piste [Track])
contains the track number of the freed sector
FS (Free Sector)
contains the sector number.

APPENDIX – Disc Structure

The first sectors of a Master disc (from Sector 1 of Track 0) are occupied by the boot routines and then the DOS.

The sectors identified below have a fixed position and are reserved at the time of the disc initialisation.

Track	Sector	Use
20	01	Disc name
20	02	Bit map
20	03	Bit map
20	04	Directory 1
20	07	Directory 2
20	10	Directory 3
20	13	Directory 4
20	16	Directory 5

The remaining sectors are free for the storage of files or data.

APPENDIX – File Status Byte Coding

The status byte of a file is binary coded and the significance of each bit is given below.
The byte follows the standard bit format, i.e. b7 to b0. The bit is active if it is set to a '1' and inactive if it equals '0'.

b0 : automatic execution
b1 : unused
b2 : unused
b3 : direct access
b4 : sequential access
b5 : window (note – b6 = 1 also)
b6 : data block
b7 : BASIC file

APPENDIX – Keyboard Codes for **KEYIF** Instruction

Key	Code	Key	Code	Key	Code
1	#A8	2	#B2	3	#B8
4	#9A	5	#90	6	#8A
7	#80	8	#87	9	#8B
0	#97	–	#9B	=	#BF
\	#B3	ESC	#A9	Q	#B1
W	#BE	E	#9E	R	#91
T	#89	Y	#86	U	#85
I	#8D	O	#95	P	#9D
[	#BD	]	#B5	DEL	#AD
CTRL	#A2	A	#AE	S	#B6
D	#B9	F	#99	G	#96
H	#8E	J	#81	K	#83
L	#8F	;	#93	"	#BB
RETURN	#AF	SHIFT L	#A4	Z	#AA
X	#B0	C	#BA	V	#98
B	#92	N	#88	M	#82
,	#8C	.	#94	/	#9F
SHIFT R	#A7	→	#AC	↓	#B4
SPACE	#84	↑	#9C	←	#BC
FUNCT	#A5				

APPENDIX – Switching between RAM and ROM

The DOS is contained in RAM overlay, which 'shadows' the BASIC ROM.

When in ROM, **JSR #04F2** takes you to RAM overlay. Another **JSR #04F2** returns control back to the ROM.

When using this routine there is a risk of 'hang-up' when returning to BASIC so care must be taken. Switching between ROM and RAM overlay does not affect any of the 6502 registers.

APPENDIX – Utility Files on SEDORIC Master Disc

The SEDORIC master disc contains several programs and utilities.

CONVERT

Easily transfers your files created under ORIC DOS format to SEDORIC. Load by typing CONVERT. The program is menu-driven – move between choices with the space bar or cursor arrows, and validate a highlighted choice with RETURN.

ESC returns you immediately to the menu, and ESC at the menu returns you to BASIC. There are 4 menu options:

Selection of drives
Drive selection – enables you to select which drive will hold the old format disc and which the new – they can of course both be the same drive, as with the COPY command.
Select the required drive with the space bar and press RETURN to toggle between source and target drives.
ESC to return to the menu.

Initialisation of target disc
Enables you to format and initialise the target disc, equivalent to the command INIT. A slave disc is formatted. Since the track and sector format is read from memory the values default to those read from the bootup DOS disc. Therefore double-sided formatting of discs of 80 tracks is supported.

Selection of the DOS version
Always leave this on ORIC DOS – the others are French DOS's.

Conversion of files
The heart of the utility. Insert the source disc in the relevant drive. Press the DEL key to produce a catalogue of files on the source disc – only the first 80 (!) files are copiable.
To select files to transfer/convert, move the highlight bar with the cursor arrows to each file that you wish to transfer in turn, and at each file press SPACE. A star will appear by each file you select. When you have finished selecting files, press RETURN to start the file conversion. If one drive is being used, the usual prompts for loading source and target discs will appear. As each file is transferred a '+' replaces the '*'.
If an error occurs, press ESC to move to the next file, or any other key to retry the file that produced the error.

Note: CONVERT will copy ORIC DOS data files across onto the target disc, but they will not be converted for use by SEDORIC DOS.

GAMEINIT

To initialise a games disc containing 'SHORTSED' – a DOS of only three instructions, *!LOAD, !DIR* and direct file loading (i.e. *! (name of file)*), each with its usual syntax. This disc will have only 17 sectors occupied by the DOS, as against 100 plus for a normal Master disc.

The syntax is close to that of INIT, except that you put a colon after the command, e.g. *GAMEINIT:A,17,42*. If no parameters are entered, the default values of 42 tracks, 17 sectors, drive A are used.

NOTE: *GAMEINIT* does not permit more than 1920 sectors on a double-sided disc.

ROMORIC1

This loads the V1.0 ROM into RAM overlay, giving you the Oric–1 ROM on an Atmos. Note of course that you no longer have the DOS in memory, and any programs must be loaded from/saved to cassette.

ROMATMOS

The identical facility to give the Oric–1 a V1.1 ROM in RAM overlay.

ADDRESS

This is an example of a direct access file. Type ADDRESS and press RETURN to run the program. There are two associated files – ADDRESS.WIN and ADDRESS.DAT

ALPHA

ALPHA (RETURN) enables you to sort a disc's directory in alphabetical order. It allows a disc to be sorted in any drive.

STAT

STAT enables you to produce statistics on the number of times instructions are used in a BASIC program. Type *STAT, J*, then RUN 64000. The program only lists those commands actually encountered.

VERSION

Gives you the version of SEDORIC you are using. It allows a disc to be inserted in any drive.

SECTOR MAP

This utility displays the state of all sectors on the disc, whether they are (O)ccupied or free (.). Output can be to the screen or printer. To run type SECTMAP and press RETURN. Discs formatted to 80 tracks double–sided are supported.

APPENDIX – Known bugs and limitations

Version 1.006:

1. TAKE command – variables were corrupted

2. LDIR does not filter out screen control codes

3. FIELD command only operated on one random access file at a time

4. Formatting a double–sided disc did not give the correct indication when displaying the directory

5. In formatting a double–sided 80–track disc, the number of sectors was limited by the BITMAP to 1920. INIT was therefore limited to the following formats for 16 or 17 sectors per track: *INIT A, 16, 59, D* *INIT B, 17, 56, D*

6. Format failures can occur when using INIT or BACKUP for formatting.

7. All utilities were written in French.

APPENDIX – Developments and solutions

Version 1.007:

1. TAKE command has been fully corrected

2. FIELD bug: When opening more than one random access file at once, include the FIELD definition in a sub–routine and call it prior to accessing a particular file

3. Double–sided directory display bug: After the first format return to command level, re–enter INIT, do not format, but complete the initialisation as required

4. If a format failure occurs, there is no option but to run the command again

5. All utilities are translated into English

Version 2.0:

1. The restriction of the total number of sectors permissable (formerly 1920) has been removed.

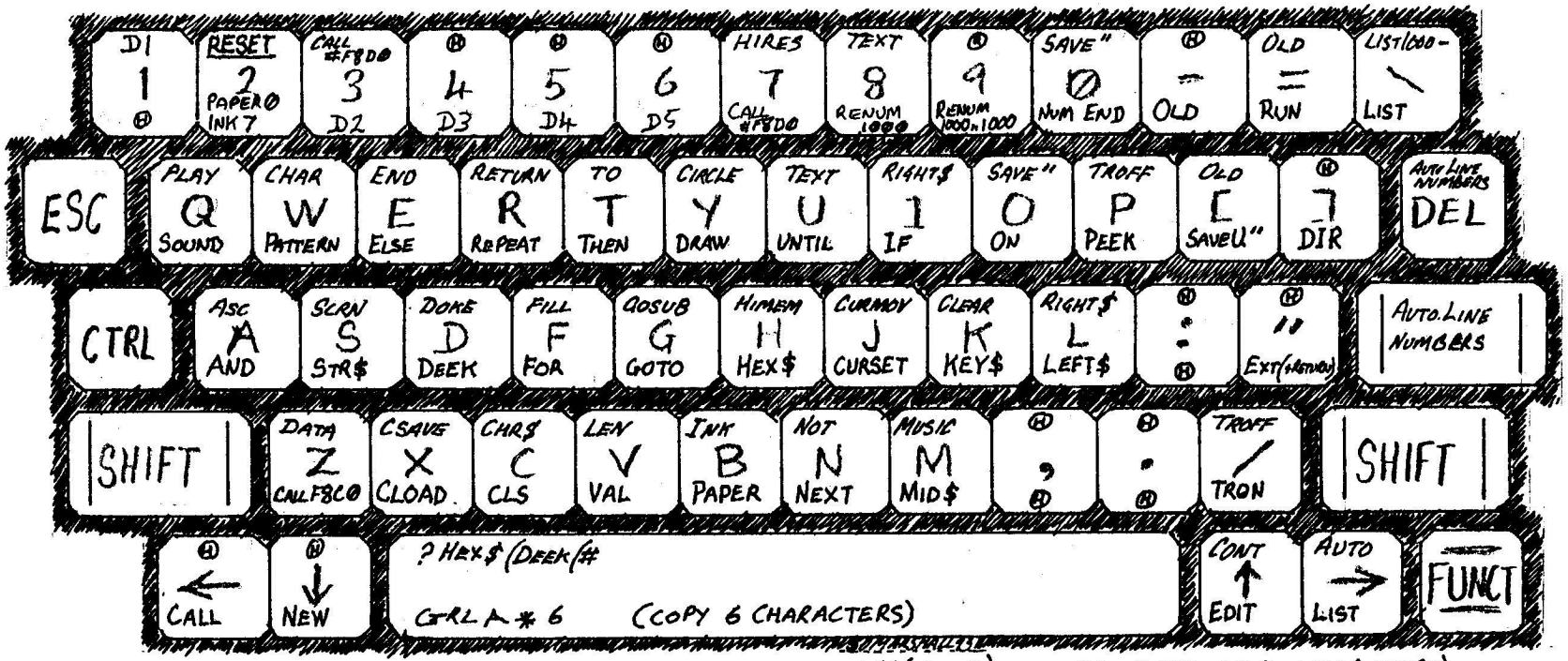
2. The directory indication for a double–sided disc is now correct following the first format.

3. The *SECTMAP, ALPHA* and *VERSION* utilities on the master disc have been updated.

4. The packaged game has been changed from M.A.R.C. to Krillys.

5. The master disc contains a file **NIBFIX** which will amend the disc sector editor 'NIBBLE' to work with the updated DOS. This is the only known program that needs amending to be compatible with V2.0.

Other bugs and limitations remain for the present



KEY Ⓗ = ?HEX$(DEEK(#

D1 = DOKE #2F5, #463 (+RETURN)

D2 = DOKE #24E, #108 (+RETURN)

D3 = DOKE #245, #EE22 (+RETURN)

D4 = DOKE #245, #484 (+RETURN)

D5 = DOKE #23C, #EB78 (+RETURN)