CORTEX DISK OPERATING SYSTEM

VERSION 1.1Ø

USERS GUIDE

CORTEX DISK OPERATING SYSTEM

VERSION 1.1Ø

USERS GUIDE

SECTION

1. INSTALLATION

When the "BOOT" command is executed the disk in drive Ø is read into memory. The command is told what type of disk and disk drive by the DENSITY and SIZE jumper settings.

Jumper	In	Out
DENSITY	SINGLE	DOUBLE
SIZE	5¼"	8"

Note that these settings are used at "BOOT" time only and that the CDOS software reconfigures the drive size and density. (See CONFIG utility).

(For 5¼" drives that are 2 sided make sure that the signal from IC83 pin 8 is connected to pin 32 of the disk drive connector).

2. BOOT

The "BOOT" command reads the first sector of the first track on drive Ø. In this sector is a parameter block with a checksum. The command verifies the checksum before using the parameter block to read in the core of the disk operating system. The parameter block specifies the load address, entry point and length of the core. The core is always on track Ø of drive Ø. If "BOOT" fails to read the disk or the checksum fails then it will start again, looping indefinitely trying to read the disk. The only way to stop this loop is to press the reset button. If "BOOT" reads the parameter block successfully but then encounters an error while reading in the core of the disk operating system then the error message:-

" **** SYSTEM ERROR **** "

will be displayed. Once the core has been loaded successfully then it is executed as an assembly language program. The core tries to find and load the program file called "SYSTEMØ". This file contains extensions to the core system such as data file support and also re-configures the disk size/density. If the core fails to find the file then it will just display the "*Ready" banner and you will only have program load and save facilities available.

If the core is successful in loading the "SYSTEMØ" file then this is executed as an assembly language program and the banner message

"CORTEX DISK OPERATING SYSTEM 1.1Ø © 1984"

will be displayed. (Note that the revision number may change as improvements are made to the software.)

At this point it is strongly recommended that the user (with reference to the FORMAT and DISKCOPY utilities in Section 5) make a copy of the system disk.

Type in the following:-

LOAD Ø, "FORMAT"

this will load the disk format utility. Remove the system disk from drive Ø and insert a blank disk. In answer to the question which drive type in Ø. After the program has finished and asks again which drive just press "return". Now remove the initialised disk and insert the system disk. Type in the following:-

LOAD Ø, "DISKCOPY"

this will load the physical disk copy utility. The "Master" and "Copy" drive numbers are both Ø and press return in answer to the question "Number of Tracks". The program will read in several tracks from the master disk and then prompt you to remove the master and insert the copy disk. It then writes the data to the copy and then prompts you to put back the master. This is repeated until all of the master has been copied. After the program finishes you can now put your master system disk in a safe place and use your new system disk. Use the utility "LDIR" to list the contents of the system disk. You may want to use the "CONFIG" utility to optimise the system software for your disk size/type and configuration.

3. PROGRAM STORAGE

Once the operating system is installed then it alters the routines for loading and saving programs to allow both cassette and disk to be used.

3.1 SAVING BASIC PROGRAMS

The statement syntax for the "SAVE" command is changed to the following:-

(line number) SAVE < drive, "filename" > (,REP) (,EX) (,EX) (,REP)

The "SAVE" command can be executed as part of a program hence it can have a line number associated with it. The drive number can be either a digit Ø to 3 or a variable assigned a value between Ø and 3. The filename is either a text string enclosed in quotes or a string variable, but only the first 8 characters are used. Optional items are "REP" which specifies replace an existing file of the same name if it exists. Programs can be saved with the option of automatic execution upon load, this is specified with the "EX" keyword. Note that "EX" and "REP" can occur in any sequence. Some example commands are:-

```
    SAVE 1, "TEST", EX      ! save on drive 1, filename TEST with auto run option
1Ø  SAVE D, $N(Ø)           ! save on drive D, filename in $N(Ø)
```

3.2 LOADING PROGRAMS FROM BASIC

The statement syntax for the "LOAD" command is changed to the following:-

(line number) LOAD < drive, "filename" >

The "LOAD" command can be executed within a program to cause chaining from one program to another, hence the optional line number. The drive number can be either a constant or a variable assigned a value between Ø and 3. The file-name is either a string of characters enclosed in quotes or a string variable. Note that only the first eight characters are significant. If the program was saved with the automatic execution option specified then the program will start execution immediately after loading. You can load both BASIC and Assembly Language programs from the BASIC environment. An example of the command is:- LOAD 3, "MYPROG"

3.3 CASSETTE SUPPORT

The command syntax as described in the Cortex Users Guide for the "SAVE" command now applies to the "CSAVE" command and likewise the syntax for "LOAD" now applies to "CLOAD".

3.4 SAVING ASSEMBLY LANGUAGE PROGRAMS FROM THE MONITOR

The syntax for the "D" memory dump command has an added question as shown below:-

D < start address > < stop address > {entry point}

IDT = < filename >

Auto-run? (Y/N) {Y / N}

Device ? {C / drive}

The memory from start to stop is written either to a disk drive (Ø to 3) or to cassette (C). The optional entry point is used for auto-run files to execute immediately upon load. The IDT is the filename, only the first 8 characters are used. An example is given below:-

```
D   8000  9000   8100
IDT = MYPROG
Auto-run? (Y/N) Y
Device?  1
```

3.5 LOADING ASSEMBLY LANGUAGE PROGRAMS FROM THE MONITOR

The syntax for the "L" memory load command has an added question as shown below:-

```
L   IDT =     < filename >
Device ?   (C)
           (drive)
```

The specified file is loaded into the area of memory that it was saved from. Either a disk drive (0 to 3) or (C) cassette may be used to read the program from.

4. DATA STORAGE

The operating system supports both sequential access files and random access files. A sequential file allows reading or writing of consecutive elements. A random access file allows any element to be read or written in any order but requires that each element be of a fixed size. Sequential files support both variables and strings of any length mixed in a file. Strings are written in compressed form where a sequence of more than two spaces is shrunk to a control character followed by the number of spaces. The converse operation occurs on reading a string in a sequential file where the control character is detected and the spaces are restored.

4.1 CREATING A DATA FILE

All data files are created as part of the OPEN command. The syntax for the OPEN command is:-

(line number) OPEN < drive, "filename", filevariable > (,CRE (,record size))

The line number is optional.
The drive number is between Ø and 3 and can either be a variable or a constant.
The filename is either a text string enclosed in quotes or a string variable, but only the first eight characters are used.
The filevariable is a variable used to store a pointer for use by the GET, PUT and CLOSE commands.
Optional items are "CRE" which will create a data file if it does not exist.
If you do not specify a record size then a sequential file is created otherwise a random file is created.
EG:

```
1ØØ     OPEN     D,$NAM (Ø), F, CRE, 1ØØ
                    or
20       OPEN     1, "DATA", F
```

4.2 WRITING TO A DATA FILE

The "PUT" command writes to a data file. The syntax for sequential files is:-

NB: SEE BACK PAGE.

(line number) PUT < filevariable, variable > (, variable, ...)

and for random files the syntax is:-

(line number) PUT < filevariable, record, variable > (,variable, ...)

↓ can be variable or number but not calculation

The line number is optional. eg RN, 1 .NOT :- (L-P1)+2

The filevariable is the variable name used when the file was opened.

For random files you must specify which record to write to.

(Note that writing to only one record does not initialise the contents of lower record numbers, i.e. unless written to each record up to the end of the file will contain random (garbage) values).

The variable or variable list separated by commas will be written sequentially to disk.

For random files the information starting at the first byte of the specified variable up to the record length is transferred (see examples in Section 8).

Files that are written to must always be "CLOSE"d afterwards.

An example of the "PUT" command to a sequential file is:-

```
100 PUT F, $S (0)   !   WRITE STRING TO FILE
```

4.3 READING FROM A DATA FILE

The "GET" command reads data from a data file

The syntax for sequential files is:-

(line number) GET < filevariable, variable > (,variable .....)

and for random files the syntax is:-

NB: SEE BACK PAGE.

(line number) GET < filevariable, record, variable > (,variable.....)

↘ see GET OUT

The line number is optional.

The filevariable is the variable name used when the file was opened.

For random files you must specify which record to read from.

The variable or variable list separated by commas will be read sequentially from disk.

WARNING

Reading items that are larger than the space allocated to the variable specified will CORRUPT successive variables.

An example of the "GET" command from a random file is:-

```
1Ø  DIM  X(1Ø)          !  66 bytes of space
2Ø  GET  F,4, X(Ø)      !  read record 4 into array
```

4.4 TERMINATING DATA FILE ACCESS

All data files that are "OPEN" must be "CLOSE"d before the program stops to make sure that the directory entry is updated with the end of file position. The syntax for the "CLOSE" command is:-

(line number) CLOSE < filevariable >

The line number is optional.

The filevariable is the variable name used when the file was opened.

E.g.:-

```
500     CLOSE     F1     !     close file F1
```

5. UTILITY PROGRAMS

The CDOS system disk is provided with seven programs to enable the user to efficiently manage programs and data stored on disk.

5.1 "LDIR" - LIST DIRECTORY

The "LDIR" program lists the contents of the disk and gives information about the type and size of the files. To use the utility type in:-

LOAD Ø, "LDIR"

This loads the program from drive Ø, but any other drive containing the program could be used.
The program prompts for the drive number, enter a number between zero and three. The total number of allocated and free blocks is displayed, followed by the header:-

Filename Blks Rsize Recs Type

There then follows a list of the files on the disk. The full name is followed by the number of blocks allocated to the file, then the record size, number of records and then file type. The file type is one of the following types:-

auto-run program
program
auto-run basic
basic
sequential data
random data

The first pair are assembly language programs with the first having the auto-run option set.
The second pair are basic programs with the first having the execute option set.
The last pair are for sequential and random data files.

5.2 "DELETE" - DELETE FILE

The "DELETE" program allows the user to remove a file and free up its allocated space on a disk. To use the utility type in:-

LOAD Ø, "DELETE"

This loads the program from drive Ø, but any other drive containing the program could be used.
The program prompts for firstly the drive number, then the filename. It then searches for the file on the disk, if it is found then the program prompts "ARE YOU SURE?" Typing "Y" will delete the file, any other answer will not. Delete will release the space allocated to the file for future use and remove the file entry in the directory.

5.3 "FORMAT" - DISK INITIALISE

The "FORMAT" program takes a virgin disk and initialises the soft sector format required by the disk controller. To use the utility type in:-

LOAD Ø, "FORMAT"

This loads the program from drive Ø, but any other drive containing the program could be used
The program prompts for the drive number. If you just hit the "Return" key

then the program will stop and return to basic. Type in a number between zero and three and then the program will start formatting.

(Note that the disk size, density, number of sides and number of tracks is set up by the "CONFIG" utility)

5.4 "DISKCOPY" - DISK PHYSICAL COPY

The "DISKCOPY" program copies byte for byte from one disk to another with no checks on file types etc. To use the utility type in:-

LOAD ø, "DISKCOPY"

This loads the program from drive ø, but any other drive containing the program could be used.
The program prompts for the master (source) drive number, followed by the copy (destination) drive number. (Note that single drive operation is possible as the program will then prompt you to swap disks in and out of the drive). The last prompt is for the number of tracks, to copy the whole disk just press the "Return" key.

WARNING This utility assumes that the copy drive is of the same size, density and number of tracks. Use "FILECOPY" for dissimilar drives.

To make a disk which will BOOT and load in the kernel you can just transfer the first track between any equal sized drive, and from single density to either double or single density and from double to double density disks. You must then use "CONFIG" to create the "SYSTEMø" file on your new disk with the correct size, density etc.

5.5 "FILECOPY" - COPY FILE BY FILE

The "FILECOPY" program copies each file from one disk to another. It can be used between drives of different sizes, or with different densities or number of tracks/sides. To use the utility type in:-

```
LOAD  Ø, "FILECOPY"
```

This loads the program from drive Ø, but any other drive containing the program could be used.
The program prompts for the source drive, followed by the destination drive number. (Note that single drive operation is possible as the program will then prompt you to swap disks in and out of the drive).
The program reads the source directory, when a file is found it prints:-

```
Found  FILENAME
```

It then creates (if it doesn't exist) a file of the same name on the destination drive and copies the file block by block. This continues for each file until the message:-

```
END OF DIRECTORY
```

is displayed.

Note that files which have more blocks allocated than used are compressed down to the actual size required on the destination drive.

5.6 "RENAME" - CHANGE FILENAME

The "RENAME" program allows the user to alter the name of any file. To use the utility type in:-

LOAD Ø, "RENAME"

This loads the program from drive Ø, but any other drive containing the program could be used.
The program prompts for the drive number, then present filename. It then searches for the file, if it is found then the program prompts for the new filename. The program checks if that filename has already been used before renaming the file.

5.7 "CONFIG" - SYSTEM CONFIGURATION

The "CONFIG" utility program gives the user the ability to alter disk drive parameters to suit his hardware configuration. Note that the CDOS is shipped configured for ; (a) 5¼", 4Ø track, single sided, (b) 5¼", 8Ø track double sided, or (C) 8" single sided. The 5¼" options are for TEAC 5ØA and 5ØF drives respectively and the 8" option is for SHUGART SA8ØØ drives.

To run this utility type in:-

LOAD Ø, "CONFIG"

The program reads the "SYSTEMØ" file into memory and then extracts the information it requires to tabulate against each drive the characteristics as last set up. The table lists the drive size (8 or 5), the number of sides (1 or 2), the number of tracks (40 or 80 for 5¼" or 77 for 8"), the recording density (single

or double) and the drive stepping rates (A or B). Under this table is smaller table of stepping rates which lists the time delay in milliseconds that the disk controller uses for any disk access. The three parameters are, (a) head stepping speed, (b) head settling time after stepping and (c) head load delay after settling time. Note that with 5¼" drives head load is activated directly from the Motor On signal and therefore doesn't need a time delay.

The program prompts you for the following parameters:-

"Drive ?"	answer Ø to 3
"Size ?"	answer 5 or 8
"Sides ?"	answer 1 or 2
"Tracks ?"	this is only asked for 5¼" drives. 8" drives are fixed at 77 answer 4Ø or 8Ø
"Density ?"	answer S or D for single or double density respectively.
"Timing ?"	answer A or B timing parameters.
"Rate A Timing"	- title then appears
"Head step?"	- enter stepping period in milliseconds
"Head Settle?"	- enter settling period in milliseconds
"Head load?"	- enter head load delay period in milliseconds
"Rate B Timing"	- title appears and the same three questions about step, settle and load occur
"Save (Y/N)"	answer Y if you have finished specifying all the changes or answer N to go back to ask for drive once again.

Upon typing "Y" in response to the "Save" prompt the program writes back the modified "SYSTEM$Ø" file and displays the message "SYSTEM$ updated" before stopping

WARNING always maintain a backup copy of the "SYSTEMS" file as inadvertent changes using this program can make the system inoperative!
It is recommended that drive timing parameters are not changed without careful thought and always try out your changes on only 1 drive at a time. The newly configured system will not come into effect until you "BOOT" the system again.

The sequence to change drive Ø to a different density or size is as follows:

(a) Re-configure drive 1 to the required characteristics for drive Ø.

(b) Re-boot the system.

(c) Format a disk in drive 1.

(d) Re-configure drive Ø to the required characteristics and restore if necessary the characteristics of drive 1.

(e) Use "DCOPY" to copy from drive Ø to drive 1 just ONE track.

(f) Use "FILECOPY" to transfer all the files to the new system disk.

(g) You now have a re-configured system disk.

6. INTERNAL ROUTINES

There is one common routine which can be used with assembly language to access a disk. The routine is called BYTEIO. It allows the caller to read or write to any byte on any disk. The calling sequence and parameters are:-

call:- BLWP @ >618Ø

parameters are passed in the callers workspace:-

RØ - high byte is status returned by the routine, the low byte is the read/write flag. If the flag is non-zero then a disk write is performed.

R1 - high byte contains drive number Ø to 3.
low byte contains high order disk linear address.

R2 - low order disk linear address. Each byte on the disk is accessed by its index starting at Ø which reads (writes) to sector Ø, track Ø.

R3 - buffer address (source or destination for data).

R4 - transfer byte count

The routine returns to the caller with an RTWP.

The status is zero for successful transfers or it contains the code generated by the disk controller. A standard error message can be generated by branching to location >655Ø with the controller error code still in RØ.

7. ERROR CODES

There are additional error messages provided to cover the error conditions that arise during file accesses.

Error Number	Text message
50	Drive error
51	Disk write protected
52	Controller error
53	File not found
54	File not replaced
55	Directory full
56	Disk full
57	File too fragmented
58	File type mismatch

The conditions which cause these errors are:-

Drive error - Disk not in drive, drive timing incorrect, drive hardware error.

Disk write protected - For 8" disks the write protection is caused by the lack of a label over the notch. For 5¼" disks the notch must be covered to stop accidental writing to the disk.

Controller error - Damaged disk, bad format, illegal disk address, or timeout on access.

File not found - The filename specified was not found on the disk check your spelling.

File not replaced - During the SAVE command a check is made to avoid accidentally over-writing a wanted program. You must specify the REP option to update the file contents.

Directory full - There is a limited quantity of disk space reserved for the directory (list of contents) of the disk. A single density, 5¼", 40 track disk allows only 30 files. Most others allow more than 60 files.

Disk full - Files are allocated space in units of one sector, so your disk has no free sectors left. (FILECOPY may recover disk space for you).

File too fragmented - Files are allocated space in blocks on the disk. After frequent use all large free blocks are used and the system has to piece together small blocks to get sufficient space. Up to eight small blocks are allowed before the disk space gets too fragmented to allocate space for the file.

File type mismatch - If you try to load a data file as a program then you will get this error. Also when opening a data file with the "CRE" create option where the filename already exists but the file is of a different type or record size will generate this error.

RANDOM ACCESS FILES

WHEN THE OPEN COMMAND IS USED, 96 BYTES IN RAM ARE RESERVED AS A WORKSPACE FOR FILE HANDLING.

IF RECORD SIZE IS, SAY 40, THEN WHILST

PUT FILEVARIABLE, R.NUMBER, VAR IS OK,

GET " " VAR IS NOT,

SINCE 40 BYTES WILL BE TRANSFERED INTO VAR, THUS CORRUPTING SUBSEQUENT VARIABLES,

SOLUTION.

DIM VAR(6) (VAR = 42 BYTES)

'GET FILEVARIABLE, R. NUMBER, VAR(0)