**RANDOS**

*Instructions for use of the operating system of micro disk Randos.*

We here confine us to indicate the differences with the DOS described in the manual of the micro drives Oric.

Randos offers a number of additional possibilities and expands the use of some functions of the back.

Immediately include tree directories. Suppose that three people: Luke, John, and Paul use the same disk. Each will create its own program list. Initially, the directory will contain LUC.DIR, JEAN.DIR and PAUL.DIR.

The appeal of LUC.DIR will display Luc programs only. It is more convenient. There are also management distinguish chapters and subchapters.

  **!BACKUP** works as with the back. To copy files from a DOS disk to a disk HIKES have features on the disk of a special programme TRANS.

!BACKUP with RANDOS copies only the areas that contain files. Thus, mirroring is done as soon as possible.

  **!MAKE** is used to create a subdirectory. The name of this subdirectory will be displayed with the suffix.DIR and must be different from the names of the other files. If you choose one extension other than DIR it will be ignored and automatically replaced by.DIR

!MAKE 'Games' will create a JEUX.DIR subdirectory. To enter in this subdirectory must be used!CHANGES.

!MAKE, S indicates that the subdirectory in service should be the root: it is one that will be used at the beginning, during the start-up of the system.

**!Exchange** is used to change directory. If it says nothing after Exchange, it gets the "root directory". If the named subdirectory does not exist in the current directory, an error message will be generated. Needless to give suffix, it will be ignored anyway and replaced by .DIR.

!FORMAT brings the service directory to become root.

**!DIR** displays all files in the directory service. If more than one screen is required, pressing a button displays the following page. Escape to exit. The used drive is selected by default.

!DIR 1 displays the directory of the No. 1 player.

!DIR "\*."DIR"display all sub directories.

!DIR "1-\*."DOWN"displays all BASIC files of the No. 1 player, extracted from the directory service.

!DIR 'XX.SUF"displays this single file.

!DIR, gives a detailed directory.

**!DEL** allows to delete files, but cannot delete directories unless they viennentt to be created and are empty. It must first unprotect by!PROT 'XX.DIR", N before deleting the directory by!DEL 'XX.DIR"

**!COPY** works as with the back. New: option; which makes the file invisible to the screen name when displaying the directory. Another option is, B. When you copy a file with this option, this causes the registration, (b) in the list of options for this file. If you modify this file, this indication is cleared. So, an update of the backups is facilitated. Just call another type programs and do will be recopied that those who do not have the B option in their description. The use of the 'carte blanche' characters is of course interesting here.

**!DRV**

!DRV 1 indicates that the recorder player number 1 was chosen as the priority unit.

!DRV? Returns the number of the priority unit.

If you choose a unit that does not exist, the unit will search desperately for it...

**!FORMAT**

!FORMAT 0 (don't forget the space between FORMAT and 0)

When you send this command, you are asked:

SYSTEM DISC? Enter Y if you want that the HIKES appears on the disk you are initializing. This is the most common case. Otherwise, enter N.

If you answer Y, you will see:

READING SYSTEM...

READING DOS PROGRAMS...

Then put the disk a boot in the reader that you have indicated at the outset and press RETURN.

Asked then:

No OF TRACKS? (number of tracks) Answer 40 or 80 followed by RETURN.

DISC NAME? (name of the disc) Enter it in 6 or fewer characters and then RETURN.

DUAL SIDED? (double sided) Type Y (Yes) or N (no)

Initialization begins.

Note: it is not possible to imply the number of Unity (therefore indicate!FORMAT 0 or!1 FORMAT, etc.)

This command destroys the contents of the memory and closes all open remaining files.

**!LOAD** is similar to that of the back.

It closes the files remained open.

Files with the suffix.DOWN or .COM can be recalled by! followed by their name without the suffix. They start automatically.

**!PROT** offers a new option:, A authorizing the extension of the file.

**!REN** can not be used for directory names.

**!SAVE** works as done for the back.

There is in addition to the O option which allows the backup of a file while the same already exists on the disk. In this case, the new replaces the old. It is written in its place.

**!SYS** works similarly to that of the back.

**!BUILD** used to create a text file on the disk from characters entered at the keyboard. After entering!BUILD "TEST.TXT", the data entered on the keyboard will go into this file as soon as you have typed CTRL-C. If you tap on RETURN, a carriage return and a newline will be generated to facilitate page layout.

**!TYPE** recalls on-screen text files created by!BUILD or by a program that creates text files. The SPACEBAR is used as when displaying a listing BASIC and CTRL-C to interrupt the process. BASIC programs can not be thus recalled, this is due to the way in which they are stored on the disk.

*Error messages*

1277 or #4FD

0 error with the back forcing an interruption in BASIC with error display.

1 error with back causing no termination of the program, an error number is sent in 1279 (#4FF).

Normally, when an error occurs, an error message appears. You can suppress the display by POKE 1277,1 and restore it by POKE 1277,0. You can use the number contained in 1279 for error handling routines. Here is the list of error numbers and their meaning.

0 no error.

1 file not found

2 end of faulty

3 forgotten drive number

4 invalid drive number

5 invalid file names

6 error on the disk

7 invalid attribute

8 'carte blanche' unauthorized

9 file already created.

10 insufficient space on the disk

11 file already open

12 directory already in service

13 missing the ending address

14 greater than the ending address start address

15 'TO' forgotten

16. the files are not on the same disk

17 non-sized table

18 separate drives

19 disks of different types

20 bad disk. Formatting impossible.

21 already saved file

22 syntax

23. the name of file missing

24 service file number

25 type confusion

26 write-protected disk

27 Disc changed, unable to write

28 number of files not in use

29 end of file

30 not found directory

31 existing directory

32 open file in read only

33 too large table

34 too large registration number

35 invalid file number

36. too many open files

37 reserved

38 end of record

39 reserved

40 reserved

41. the file used

42 out of memory

*Data files.*

With the system walks there files for sequential access, direct access, possibility of access byte-by-byte, backups and reminders of data in tables.

Files, records, fields.

A file contains a number of records, each record is divided into parts called fields. The fields are usually designated by names of variables name$, $, ADR$...

a disk can have a number any file. However, we cannot exploit that 8 simultaneously. Each file can contain up to 32768 entries each 65536 bytes to the most. All this within the limits of the space available on the disk. In practice, a file contains hundreds of recordings of each a few hundred bytes at most.

Description of a record.

Here is an example of an address file fields.

HIKES must know the length of each field. There must therefore be five bytes here since there are five fields. Each record will be therefore 175 bytes. You can save thousand people (from 0 to 999). Suppose that our file has 300 names, you must book the disk space of 300 x 175 equal 52500 bytes.

To save space.

When the HIKES is launched, a part comes to settle into RAM. Some commands will remain on the disk and will be called as needed in OVERLAY. Here are the contents of 9 OVERLAY systems.

SYSTEM.OV0 Error Messages.

SYSTEM.OV1 - GET, OPEN, PUT, SET, CREATE, EXTEND, FILES

SYSTEM.OV2 - DIR, SAVE, OLD, DRV

SYSTEM.OV3 - FORMAT

SYSTEM.OV4 - BACKUP

SYSTEM.OV5 - STORE, RECALL, EXCHANGE, MAKE

SYSTEM.OV6 - DEL, PROT, REN, BUILD, TYPE

SYSTEM.OV7 - routines for access byte by byte RNDBYTE, WRNBYTE, SETRAN

SYSTEM.OV8 - COPY

You can remove a disk those that do not.

1. Unprotect. 2. delete.

Attention to OV7 called by various other.

As HIKES Initializes the disk micro marking sectors of 512 bytes with 9 sectors per track, for a disk single sided with 40 tracks the square available is around 184000 bytes.

Writing a batch file.

5!DEL "ADR.DTA"

10!FILES 1

20!OPEN 1, "ADR.DTA', W

30 INPUT N$

32 IF N$ = 'F' THEN 70

35 INPUT NAME$, ADR$, CPV$ AS$

40 PRINT

50!PUT 1, N$, NAME$, ADR$, CPV$ AS$

60 GOTO 30

70!CLOSE 1

In line 10, it says to open a single file. A portion of the RAM (usually 512 bytes) is reserved: it is a buffer. The statement!FILES 1 must precede all commands on the files. In line 20, opens the file "ADR.DTA' write (W = Write)

Online 30-35 it captures data at the keyboard. The procedure is here simplified to the maximum. In line 50, asked the disk write. It is sent online 70 to close the file by introduction of 'F'.

Reading of batch files.

100!FILES 1

110!OPEN 1, "ADR.DTA', R

130 INPUT K$

140 POKE 1277,1

150!GET 1, N$, NAME$, ADR$, CPV$ AS$

160 IF PEEK (1279) = 29 THEN PRINT "NOT FOUND":PRINT:GOTO 200

170 IF K$ <>N$ 150 THEN

180 PRINT N$, NAME$, ADR$, CPV$ AS$

190 POKE1277, 0

200!CLOSE 1

Online 110, it opens the file "ADR.DTA"read (R = Read)

Lines 130, 160 and 190 detect the end of file and act crazy warning. Line 150 reads the records. Line 180 proceeded to display as the correct sheet was found that checks the line 170. Finally the 200 line closes the file after use. Using a sequential file requires the reading of all records to the application. A direct access file is much faster to use, but a little trickier to create.

Initialization of a direct access file.

It starts by reserve disk space to the entire file. This is the command!CREATE that registered characters Dummies (CRC ASCII 0) to the location of the files thereby erasing any persistent data.

!CREATE "ADR.DTA", 175, 300

175 and 300 numbers correspond to the chosen example. (see description of a record).

Writing a file to direct access.

10!FILES 1

20!OPEN 1, "ADR.DTA', D, 175

30 INPUT N$

32 IF N$ = 'F' THEN 70

35 INPUT NAME$, ADR$, CPV$ AS$

40!SET 1, VAL(N$)

50!PUT 1, N$, NAME$, ADR$, CPV$ AS$

60 GOTO 30

70!CLOSE 1

Notice how, in line 40, asked the head of writing to stand for the plug VAL(N$).

Reading from a file with direct access

100!FILES 1

110!OPEN 1, "ADR.DTA', D, 175

130 INPUT K$: K = VAL(K$)

135 IF K < 0 OR K > 300 THEN 130

140!SET 1 K

150!GET 1, N$, NAME$, ADR$, CPV$ AS$

160 IF N$ = "" THEN PRINT "NOT FOUND":PRINT:GOTO 200

180 PRINT N$, NAME$, ADR$, CPV$ AS$

200!CLOSE 1

The 140 line place the playhead there where it takes to read the card number K. If the variable N$ is empty, it is that there is no such thing as plug.

Extension of a file to direct access.

A former file can be increased. You had planned 300 names and now need you 350.

!EXTEND "ADR.DTA", 175, 50

This command will include 175 x 50 bytes zero at the end of the address file. If physically, the place is not available immediately after the file, the system walks will look for space available: this does not affect your eyes, you do you to realize maybe even not.

Work byte-by-byte.

Sometimes the access to a particular byte may be useful. For example, to edit text. Here is an example where the spaces are removed. Reads the file text "ADR.DTA' and we fill in shoot.TXT, byte by byte, by not copying spaces (line 70).

10!FILES 2

20!OPEN 1, "ADR.DTA", RB

30!OPEN 2 "SHOOT.TXT", WB

40 POKE 1277,0

50!GET 1, A

60 IF PEEK (1279) = 29 THEN PRINT 'END': GOTO 90

70 IF A <>ASC("") THEN!PUT 2, A

80 GOTO 50

90 POKE 1277,0

100!CLOSE

It comes to batch files. You can also process files to direct access, in this case to write DB, but it's tricky because the length of the fields is part of the data on the disk. Care must be taken not to confuse these bytes with others...

Passing variables from one program to another.

You may wish to initiate one new program from another. It's easy with LOAD and SAVE using the AUTO option. However all variables of the first programme is lost when loading the second program. It may be that the first variables is necessary for the second. How is this passage of variables possible with the HIKES?

STORE and RECALL have a M option that saves and recalls an array using an area of RAM instead of using the drive. This convenience can be used for the passage of variables of a program other:

-Save the second program with the AUTO option

-in the first programme, put the variables in table. Book it in RAM by STORE with M option and call the second program by LOAD.

-you have written in the second programme a RECALL with the M option not forgetting to properly size the table. The variables are then available.

This technique is very powerful, the only restriction is that the table is not too large, the space available in RAM should suffice it.

*Commands in file of the HIKES.*

**!CLOSE** to close the file.

!CLOSE [< n >]

< n > is a number from 1 to 8 corresponding to that used with OPEN. If no number is specified, all files are closed.

Note:

1. If a file has been opened and not closed, an error message is generated when a logon attempt. It is customary to open files at the beginning of program and close them at the end.

2. an open left file may be damaged.

3. it is usual to close each file separately to better detect errors.

**!CREATE** reserves space on the disk for a file for direct access and put his name in the directory service.

Note:

1. This command must precede OPEN containing the D or DB options. The file name must be different from existing ones.

2. in the event of need, use EXTEND to increase the number of sheets.

**!EXTEND** is used to increase the number of sheets of a direct access file.

!EXTEND < file name >, < length >, < number of sheets to add >

**!FILES** used to open buffers for each file

!< Number of files > FILES

The number of file can go from 1 to 8

Note:

1. This number corresponds to the files that are open by OPEN and close by CLOSE

2. This command must precede the others (except HIMEM)

3!FILES 1 is optional, a buffer is always available with the HIKES

4. This command closes all files that would be remained open

**!GET** is used to read data from a file and assign it to the selected variable

!GET < n >, < list of variables >

< n > is a number from 1 to 8 corresponding to that used with OPEN

< list of variables > a variable or more comma separated

Note:

1. the reading is done in sequence and the variables are fed in order

2. If a variable is not in conformity with the requirements of the BASIC error message is displayed

3. the variables are of type string but for work byte-by-byte are the actual type

**!OPEN** opens a file for reading or writing

!OPEN < n >, < file name >, < option 1 > [, < option 2 >]

< n > is a number from 1 to 8. It can have a maximum of 8 files simultaneously. Each file has its own number, it is the one of the buffer where will transit data.

< option 1 >

R for reading

W for writing

D to read or write to a file in direct access

To write at the end of a batch file

To access the byte-by-byte, respectively are used RB, WB, DB, AB

< option 2 > length of a sheet, in bytes, if option 1 is D or DB

Note:

1. If you use R, D or as an option, it is necessary that the file exists in the directory service.

2. If you use the W option, on the contrary, the file must not exist, it will be created

3. This command must precede GET and PUT

**!PUT** is used to write data to a file

!PUT < n >, < list of variables >

< n > is a number from 1 to 8 corresponding to that used with OPEN

< list of variables > a variable or more comma separated

Same comments as for GET

**!RECALL** and **!STORE**

same use as with DOS v1.1 or V1.0

In addition, M option explained in the passage of variables

**!SET** used to place the head of read/write in the right place to read or write to a file in direct access

!SET < n >, < number plug >

< n > is a number from 1 to 8 consistent with that used in OPEN

< number of the sheet > that you want to read or write

Note:

1. an error message will be displayed if you are looking to use SET without choosing the option D or DB with OPEN

2. an error will be displayed if you are looking to read a record whose number is not compatible with the statements made by CREATE or EXTEND

General remark

For FILES, OPEN, PUT, GET, OPEN, CLOSE, SET must be a space before writing the following number.

This document is provisional and intended to assist those do not have sufficient knowledge of English

!1 2 3 HELP... 30 displayed all the examples.



**Original**

Si vous choisissez une extension autre que DIR elle sera ignorée et automatiquement remplacée par .