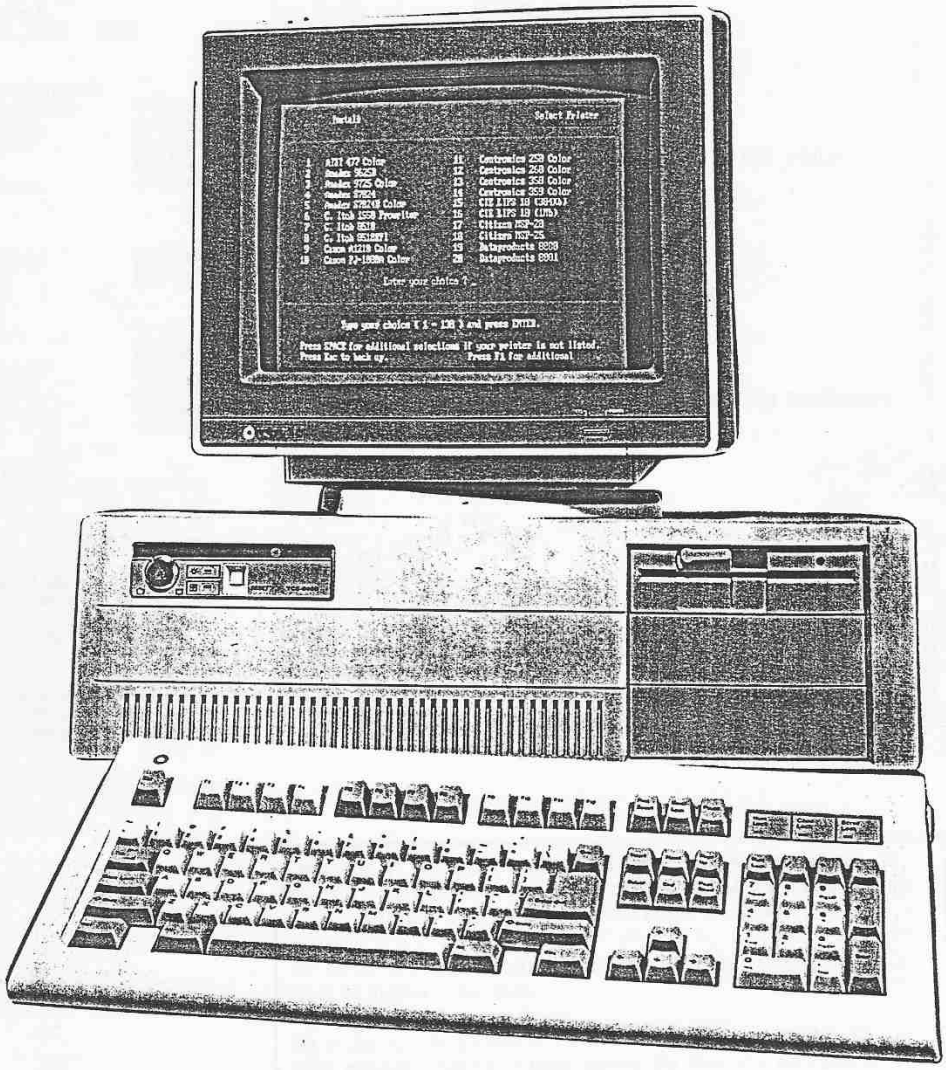
CompUser

INTERNATIONAL COMPUTING



Volume 1, Number 2, March 1988.

COMPUSER is a magazine issued by:
COMPUSER International computing;
Exchanging computer knowledge.
founded at Krimpen a.d. IJssel,
The Netherlands, by:
Willem L. van Pelt
Ir. Coen J. Boltjes.

Address all editorial correspondence to the editor COMPUSER:
Willem L. van Pelt
Jacob Jordaensstraat 15
NL - 2923 CK Krimpen a.d. IJssel.
The Netherlands.
Phone: (31) 01807 - 19881

CONTENTS VOLUME 1, NUMBER 2, MARCH 1988.

The editor is very interested in articles for the MS-DOS machines, to get more people involved with the activities around COMPUSER's issuing of this magazine, composed by all participants of COMPUSER.

The editor is also interested in articles for the DOS65 computer, based on Elektor's eurocards.

The editor will be pleased by receiving all other articles for all other machines, especially those written for Elektor's EC65 machine and for IBM's tools like SYMPHONY, dBASEIIIplus, WordPerfect, etc. Did you develop your own eprom programmer for your IBM or IBM-like computer? Send it to the editorial office, please.

LETTER TO THE EDITOR

In this colum participants may put their non-commercial announcements and questions to the other readers of COMPUSER.

IBM-PC : DISKCOPY also for RAMDISKs ?

I own an IBM compatible PC of the AT type. Who knows of a program that acts like DISKCOPY but also works with RAM-DISKs and Harddisks? A program which does the following full copy jobs:
Floppy drive a: ---> RAM drive d:
RAM drive d: ---> Floppy drive a:
Floppy drive a: ---> Harddisk c:\
Harddisk c:\ ---> Floppy drive a:
RAM drive d: ---> Harddisk c:\
Harddisk c:\ ---> RAM drive d:
An option that permits working with different formats (360K - 1.2M) between source and target would be appreciated, too.
Note that it is not a file copying program I am looking for: The public domain market offers a number of fast (selective) filecopy programs working also with hidden and write-protected files. It is the automatic inclusion of sub-directories that causes troubles.
Fred Behringer, Straßbergerstraße 9c/519, D - 8000 München 40.

WORDPERFECT

EASY WP DIRECTORY LIST.

Is it possible to make a printer-output of your chosen directory? Yes, it is possible, although your manual is not very clear at this point.
Enter F5 and go to the directory you want. Then enter Shift-F7. Now the directory or list of files, as you wish) will be saved in the printer control buffer. Leave the directory by 0, enter Shift-F7 again, choose option 4 to go to the printer-control menu.
Is your printer ready? Enter then 'G', and the directory will be listed on paper.

It will not be clear to everyone that the printspooler contains 512 bytes default (under MS-DOS 3.2), and that you can change it with PRINT /B: <size>.

IBM PRINTER ON ELEKTOR COMPUTER

If you want to connect an IBM printer to your own build system like an Octopus alias EC65, a DOS65 or a Junior, you might have a problem with the linefeed, because an IBM printer has a AUTO FEED XT line on the connector on pin 14.
When this pin is low the printer always gives a linefeed and you cannot change it with the dipswitch.
The solution is very simple. Just connect pin 14 with a resistant of 100 Ohm to the + 5 Volts on the connector (pin 18) and it will work.

Albert v.d. Beukel, Van Slingerlandtstr. 19, NL 2613 TT Delft, The Netherlands.

C-64 HINTS

Transl.: Cor Bergshoeff, Denmark.

How to simulate an <ESCAPE> ?

On most printers (not Commodore) the <Escape> key informs the printer that a special function is to take place (see user's manual of different printers). However, the Commodore 64 does not have an <Escape> key. This function can still be implemented in two different ways. Firstly with the instruction CHR$(27) in a program. Secondly, in the direct-mode with keying <SHIFT> and <:>. In either way an <Escape> is executed.

Cassette Load Problems.

Sometimes, after having 'rummaged' around with the computer, it can happen that you cannot load programs anylonger. The cassette recorder keeps going until the end of the tape. This malfunction can be restored by keying in the following instruction: POKE 0,47.
By doing this, bit 2 of address 0 is set to 1. Address 0 bit 2 is normally always 1.

Deleting of certain lines.

By keying in the following line, it is possible to delete a certain line.
POKE 781,R : SYS 59903
The character 'R' is the line number.
(0 up til & incl. 24)

Directory Hints.

If you want to see the number of available free blocks and not the whole directory, then enter the following:
LOAD "0:$$",8
Or if you want all the titels starting with 'k':
LOAD "0:$K*",8
It is for instance also possible to show all the programs titles consisting of 4 characters, of which the first character is a 'B' and the fourth character is a 'N'. You just type:
LOAD "0:$B??N",8

PC KEYBOARD

A free standing, low profile keyboard for applications within XT/AT environment, with ergonomic design featuring tactile feedback, real [illegible] rollover, very low acoustical noise, etc.
Info: Philips Kommunikations Industrie AG, Vertrieb Philips Peripherals, Eiserfelder Str. 316, D 5900 Siegen, Germany, Phone: 49 271 3850651.

TRACER FOR ACORN ATOM (and other 6502-computers).

By : Jan Krabbenbos, The Netherlands.

After I bought my Atom, I only extended it with a 6522 VIA and RAM. I didn't built a switchboard on #A000, so I didn't have all those facilities with special programs. Making machine language programs I faced the problem, that finding an error, while debugging, it was not always clear to me what went wrong. To step through the programs was (and still is) a helpfull way to debug machine language programs. The method I used for this is a common known method: generating an NMI (non-maskable interrupt) with help of the SYNC-signal on every instruction. A small circuit with one IC a switch and a resistance is all I needed to generate a NMI in the lower half of the memory (address #0000-#7FFFF). For the circuit see figure 1.
When, in figure 1, the switch is open there will be a NMI when addressline A15 is low and the SYNC-signal is high. When the switch is closed there is no interrupt possible.
After a NMI the 6502-microprocessor jumps to a vector which points to a program for doing those things which have to be done after an interrupt. In this case the program shows the instruction, address and the registers of the microprocessor. By pushing any key the next instruction will be traced.
The program is written in Atom Basic with the Atom assembler (@ = immediate, # = hex number, | = backslash). It is not a perfect program, it is just the basic idea of using the NMI for a tracer. Everyone can make the program better so it fits for everyone.

Remarks to the program:
- place it in the adress area above the Atom screen (#8Z00)
- only machine language programs on addresses lower than #8000 can be traced.
- watch out for subroutines in the monitor.

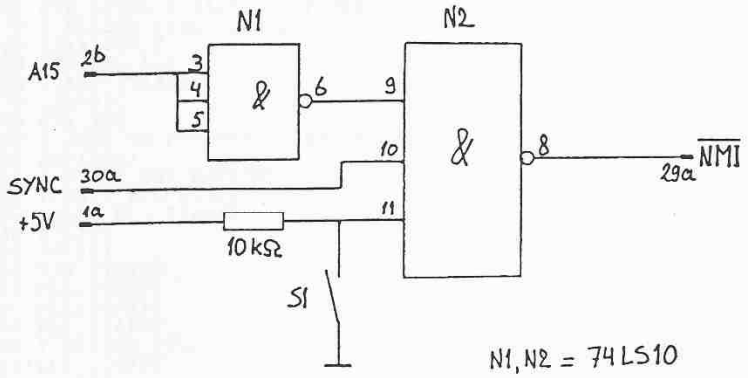


Fig. 1

USED MEMORY LOCATIONS:

Page Zero

```
# 80        Opcode
# 81        Operand
# 82        Operand
# 83        Program Counter High Byte
# 84        Program Counter Low Byte
# 85        Accumulator
# 86        X-register
# 87        Y-register
# 88        Programstatus
# 89        Stackpointer
# 8A        PCOLDL _ Old program counter low byte
# 8B        PCOLDH _ Old program counter high byte
# 8C        PCNEWL _ New program counter low byte
# 8D        PCNEWH _ New program counter high byte

# 9E/9F     Used

# DE        Screenpointer low byte
# DF        Screenpointer high byte

# E0        Screen index
# E1        Cursor
```

```
  10 DIM LL25;P.$12
  20 IN."STARTADDRESS"S;!#8A=S;?#83=?#8B;?#84=?#8A
  30 Q=#9F00
  40 P.$21;REM DISABLE OUTPUT DATA TO SCREEN
  45 REM ASSEMBLE PROGRAM, START ADDRESS 9000
  50 F.Z=0 TO 1;P=#9000;[
  60 ¦ POINT NMI VECTOR
  70 ¦ TO NMI ROUTINE
  80:LL0 LDA@#26;STA#200
  90 LDA@#90;STA#201
 100:LL2 JSR #FD69 \ CLEAR SCREEN
 110 LDY@$1F
 120 JSR #FE66 \ WAIT FOR FLYBACK
 130 ¦ HEADER ;:LL3 LDA Q,Y
 140 STA#8000,Y
 150 DEY;BPL LL3
 160 LDA@0;STA#E1
 170 LDA@#20
 180 STA #DE
 190 ¦ JUMP TO PROGRAM TO BE TRACED
 200 JMP(#8A)
 210 ¦ START NMI ROUTINE
 220 ¦ SAVE ALL REGISTERS
 230 :LL16 PLA;STA#85
 240 STX#86
 250 STY#87
 260 TSX;STX#89
 270 PLA;STA#88
 280 PLA;STA#8C
 290 PLA;STA#8D
 300 ¦ SCREEN FULL ?
 310:LL4 LDA#DE
 320 LDY#DF
 330 CPY@#81
 340 BCC LL9
 350 CMP@#E0
 360 BCC LL9
 370 ¦ SCROLL SCREEN
 380 ¦ LEAVE HEADER ON TOP
 390:LL5 LDY@#40
 400 JSR #FE66 \ WAIT FOR FLYBACK
 410:LL6 LDA#8000,Y
 420 STA#7FE0,Y
 430 INY;BNE LL6
 440 JSR #FE66 \ WAIT FOR FLYBACK
 450:LL7 LDA#8100,Y
 460 STA#80E0,Y
 470 INY;BNE LL7
 480 ¦ CLEAR LAST LINE
 490 LDA@#20;TAY
```

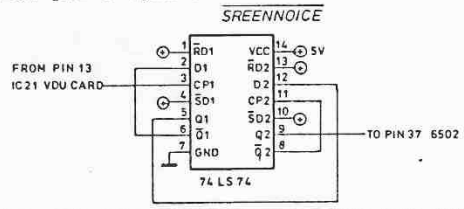
```
* dBASEIIIplus programming
*
* By : Fernando Lopes, Portugal.
*
*************************************************************************
*                          A G E S . P R G                              *
*************************************************************************
IF file->age=0.AND.DTOC(file->birth_dt)>=('  .  .  ')
  yearsage=(file->arrival-file->birth_dt)/365.25
  STORE mod((file->arrival-file->birth_dt),365.25) TO modulo
  years=INT(yearsage)
  REPLACE file->age WITH IIF(modulo>300,years+1,years)
ENDIF
IF file->age>0.AND.DTOC(file->birth_dt)='  .  .  '
  REPLACE file->birth_dt WITH CTOD(RIGHT(STR(YEAR(file->arrival)-file->age),2)+".01.01")
ENDIF

RETURN
*
*************************    u s a g e    ****************************
*
* You have a database "file.DBF" where there are, among others, 3 fields:
*   birth_dt (date), age (numeric), arrival (date)...
* You must, besides entering new elements to the file from whom you know
*   the exact date of birth, enter old records from which you know just the
*   age (at the time of arrival).
* So, you want, to ease data-entry, most of the times to type just the
*   date of birth, and expect the program to compute the age;
*   and, on the other cases, to type the age, and leave the computer to find
*   a presumible date of birth (here assumed to be the first of January).
* If someone is about to complete another year, (here assumed less of 65
*   days to go), he gets that year more.
*
* The program uses some of the DATE functions, the INT and MOD numeric
*   functions, and also the IIF (immediate if -in one line-) function.
* "File->birth_dt" means: the field "birth_dt" of database "file".
* The DATE format in use is the ANSI.
*
*************************************************************************
```

SCREEN-NOICE ON JUNIOR WITH VDU.

To avoid screen-noice, VDU-RAM, like any RAM, should not be written into, while phi-2 is low. To achieve this, processor and CRTC must be synchronized and RAM-select must be 'NANDed' with phi-2. To do this: 1) take processor out of socket, bend pin 37 (= phi 0) sideways, and connect with diagram. The VDU-clock is now system-clock. 2) take IC 8 (= N 37) of VDU-card out of socket, bend pin 1 sideways and connect with phi-2 (pin 27a of BUS).

Notes:
Do not use the 2 MHz-signal on pin 13 of IC 21, because this is the time-critical clock signal of the CRTC; capacitive wiring may affect screen.
Be sure, that the (16 MHz) X-tal-case is connected with ground.



APPLE COMPUTER INC. REPORTS RECORD QUARTERLY SALE. FOR THE FIRST TIME IN ITS TEN YEARS YOUNG HISTORY A SALES-LIMIT OF US$ 1 MILLIARD EXCEEDED.
Raise in profit with 108 percent.Cupertino/Zeist 1988
Transl.: Dirk Pickee, The Netherlands.

Apple Computer Inc. has reported an historic event. For the first time in its existence the company has exceeded the limit of $ 1 milliard in one financial quarter. The first quarter of the financial year 1988 (oktober-december 1987) showed sales for $ 1.042 milliard. This implies a raise of 57 percent compared with the $ 662.3 million in the same quarter of the previous financial year. The profit for taxes amounts to $ 121.4 million over the first quarter of 1988, a raise of 108 percent compared to the $ 58.5 million of last year.
Profit per share amounts this quarter to $ 0,92, a raise of 104 percent compared with the $ 0,45 noted for the same quarter last year. The remarkable results over the first quarter of the financial year 1988, confirm anew the succesful penetration which Apple Computer has achieved on the business and governmental markets.
This development has been supported by a recent announcement of mutual activities of Apple Computer and Digital Equipment Corporation.

A DESCRIPTION OF THE DOS65 DISC FORMAT.

INTRODUCTION.

The DOS65 operating has an efficient file manager which has a lot of nice features. In particular, it allows filenames of up to fourteen letters in length, it draws distinction between ASCII files and binary files and between command and data files, it allows directories of almost unlimited size and it re-uses the space occupied by deleted files so that discs never require compacting. These features require a disc format more complicated than is usual on an eight bit machine. This is the subject of this article.

DOS65 uses sectors of 256 bytes on single density (also called FM, Frequency modulated) or double density (MFM, Modified FM) discs. Single density discs have ten sectors per track on each side. Double density discs have sixteen or eighteen, the latter being known as `extra density' on DOS65 machines. The sectors are labelled from one upwards. The numbering covers both sides, so for example sector 17 on a double sided double density disc is actually sector 1 on the second side. Tracks are labelled from 0 to 39 or 79, depending on whether the disc has 40 or 80 tracks. An individual sector is referred to by it's track-sector address or `tsa'. The DOS65 sector read and write routines use what is termed `physical addressing' because the actual sector specified is read or written to. But the file read and write routines use what is termed `logical addressing'. The physical sector is obtained from the logical sector in a look-up table which has been constructed so that the number of physical sectors between adjacent logical sectors is a constant for that disc known as the skew. This speeds up the disc access time by giving DOS65 time to prepare itself to read a sector after reading the previous one without the disc having to make a complete revolution.

There are several types of sector on the disc. Each will be considered individually.

THE SYSTEM SECTOR.

The system sector resides at track 0 sector 1 side 0 on every disc. Amongst other things it contains the look-up table which relates logical and physical addressing and it tells DOS65 which sectors on the disc have been used. The names and functions of the various locations are listed below. You will find their addresses in your dvar.MAC file.

s.stab - (32 bytes). The look-up which relates logical and physical addressing. Each entry is greater then the previous one by the `skew' with which the disc is formatted. (Usually 2 or 3).

s.mode - (1 byte). This gives information about the contents of the look-up table. Bit 7 is `1' if there is no table, in which case logical and physical addressing are the same. This would probably be so for a silicon disc. Bit 6 is `1' if there is a side offset, meaning that the table only covers one side. In this event when finding a physical sector on the second side the number of sectors per side must be subtracted before the look-up and then added on afterwards. The remaining bits are unused.

s.mtrk - (1 byte). The number of tracks formatted. (40 or 80).

s.mcil - (1 byte). The number of sectors per track. (Total of both sides).

s.msec - (1 byte). The number of sectors per track per side. If this is half of the value contained in s.mcil the disc is double sided, if it is equal the disc is single sided.

s.bpat - (4 bytes). A bitmap pattern. Always $ff,$ff,$ff,$ff.

s.acnt - (1 byte). The allocation count, the number of sectors represented by each bit in the bit map table, s.bmap.

s.sht - (1 byte). The shift factor map per physical track. It contains the number of times a track number has to be multiplied by two to find the corresponding address in the bit map table, s.bmap.

s.tbas - (1 byte). Lowest track number on which tsl sectors can be written.

s.tbam - (1 byte). The tsl bit map of the first bitmap byte.

s.dbas - (1 byte). Lowest track number on which data sectors can be written.

s.boot - (2 bytes). Logical tsl sector of the boot file. (Found by I/O65 during booting).

s.dir - (14 bytes). Track sector addresses of sub-directories. They are zero for sub-directories which have not been created.

s.name - (24 bytes). Disc name.

s.cdat - (4 bytes). Creation date of the disc.

s.mdat - (4 bytes). Date of last modification of the disc.

s.bmap - (Rest of sector) . A `bitmap' of the disc usage. Each bit represents s.acnt sectors. If a bit is a `1' then the sectors which it represents are unallocated.

THE DIRECTORY SECTORS.

The first root (@) directory sector is found at track 0 sector 3. The track sector addresses of the sub-directories A to G are given at the location s.dir in the system sector. If they are zero the sub-directory does not exist. Each directory sector contains 15 entries. Starting at the first byte of the sector each has a 14 letter filename (with unused letters filled with zeros) followed by the logical track sector address (tsa) of the first track sector list sector of that file. Bytes 240 and 241 of the sector contain the logical track sector address of the next directory sector if there is one, otherwise both bytes are zero.
If bit 7 of the first byte of a filename is a '1' then the file has been deleted.

THE TRACK SECTOR LIST SECTORS.

Every file has at least one track sector list (tsl) sector associated with it. The tsl sectors list the logical track sector addresses of the data sectors in the correct order. In addition, the first tsl sector gives information about the file. The structure of the first sector is as follows:

ftsl - (2 bytes). logical tsa of next tsl sector. (zero if none).

btsl - (2 bytes). Backward link to previous logical tsl sector. Both zero for first tsl.

flmo - (1 byte). File mode. As shown by the `CAT' command.

```
Bit 7 - `1' if file can be read.
Bit 6 - `1' if file can be overwritten.
Bit 5 - `1' if file can be deleted.
Bit 4 - `1' if mode x (not in use yet).
Bit 3 - `1' if a command file.
Bit 2 - `1' if file executable.
Bit 1 and 0. File type: %00 Default.   ` '
                        %01 Ascii     `a'
                        %10 Binary    `b'
                        %11 Tokens    `c'
```

dblk - (1 byte). Directory sector number.

dpos - (1 byte). Position in sector. (0 to 15).

stad - (2 bytes). Load address of a file of default type. (BOOT for example). Note the load addresses of binary files are given in the data sectors.

rnad - (2 bytes). Run address of a file of default type. Note the run address of binary files is given in the data sectors.

flln - (2 bytes). Length of file.

vers - (1 byte). Version number of file. (Incremented each time file is updated).

credat - (3 bytes). Creation date of file.

moddat - (3 bytes). Date of last modification of file.

The remaining bytes ($20 onwards) are logical tsa's of data sectors. Subsequent tsl sectors contain:

ftsl - logical tsa of next tsl sector. Zero if none.

btsl - logical tsa of previous tsl sector.

All the following bytes contain the tsa's of data sectors.

DATA SECTORS.

Data sectors are entirely filled by data, they contain no information about the structure of the disc. When a file of type default, a or c is loaded into memory each of the sectors is loaded one after another in the order given by the TSL sectors.

A binary file can be loaded in two different ways. If it is loaded with the command 'load filename xxxx' where the load address xxxx is given it will load sector by sector in exactly the same way as the other file types. But data loaded in this way is not executable because mixed with the 6502 code are what I shall call 'information blocks' of five or eight bytes which give the load and execute addresses. There is always one at the beginning of a file preceding the data. A hexadecimal listing is:

02 rL rH 01 lL lH nL nH xx xx xx ...

Where xx xx xx ... are bytes of actual binary data. The 02 is a code which indicates that the following bytes are the low and high run address. It begins one of the information blocks in a 6502 code file. The 01 code occurs in every complete information block. Following it are four bytes. The first two are the load address of the next part of the file and the second two are the number of bytes to be loaded before the next information block is encountered. A file may be load at many different places in memory as a consequence of this scheme. The information block does not define the length of a file, so it is not uncommon for a file to end:

02 rL rH

When binary files are loaded with the load command without an address being specified or when they are loaded as commands the information blocks will cause the different parts of the file to be loaded at the correct places in memory.

TAPE BACKUP AND RESTORE UTILITY FOR OCTOPUS/EC-65 COMPUTERS

By : Marc Lachaert
Vaartstraat 38
B-9288 Kalken
Belgium

Preserving precious software from loss is one of the most critical things in computer matters. How many times did it happen to you too, that a diskette forgot it's contents? Or how many times it happened that due to a stupid mistake, you overwrote that program you worked on during the last ten weeks? Backuping your diskettes is the only way to be sure nothing (or almost...) can go wrong.

The most simple way to backup is naturally to make one (or several...) copies on an other diskette. However, I cannot always easely make difference between original and copy-diskettes, so sometimes fatal errors occured in the past.

After I wrote the Octofate Tape Utilities, Coen Boltjes produced a Tape Backup program for Octofate sources and text files. Since several months, I backup now my sources on tape. Tapes are in my opinion much more reliable than diskettes, and I can now very easely differentiate originals and copies (a tape does not very look like a diskette, you know...). One big handicap however, Coen's utility only works on Fate files. I wished to backup all my diskettes, sources, machine-code, Basic, all of them.

So, I decided to write a 'big' utility which was able to backup a whole disk (from track 00 to track 39) or only one or a few tracks. I wished also to restore disks from tape in a simple and comfortable way.

The 'Backup/Restore utility V1.0' is a menu-driven program, which needs only a 'standard' EC-65 and an Elektor Cassette Interface card.

The program occupies from $0200 to $1809 in the computer's memory.

When you enter the program (through the main menu on my system disk), the menu asks if you want to backup or restore and if you want to treat a whole disk or only a part of it. The fifth option is quite simple : return to DOS...

Depending on your choise, the program asks for a certain amount of parameters.

Those are:

The recording date for backup (that can be very interesting at a crash several months later!).

The disk identity. You can enter here a string of maximum 10 characters which will be recorded on the tape. So, you can give a 'name' to all of your disks and backups. The recording date and the identity will preceed the recording of every individual track on the tape.

The source drive (while backuping) or the destination drive (while restoring).

The starting- end ending tracks for backup or restore.

A query if you want (Y)es or (N)o a hardcopy of the report that will produced.

A query if (Y)es or (N)o the introduced parameters are correct.

A query to install properly disk and tape and to give an <ENTER> as aknowledge.

After this handlings, a very nice 'Report' screen is dressed which will give all the information about the backuped/restored tracks with their different sector numbers and sector lengths. It is this 'Report' which can be hardcopied if decided.

The way the information is written on tape is quite similar to the Octofate tape utility way. Information is written down in blocks of 253 chars. Each track will have its own record. Doing so, any individual track can be reloaded from tape without any problem.

A track record on tape consist of:

Recording date		3 bytes
Disk Identity		10 bytes
Track header	: - $43 - $57 - Tracknumber in BCD - $58	4 bytes
Pro sector	:	
Sector header	: - $76 - Sector number in binary - Sector length in pages	4 bytes
Sector contents	:	256 bytes/page
Sector trailer	: - $47 - $53	2 bytes

Track zero is treated a bit differently. Here, no sector information will be recorded. The contents of the track zero record on tape is as follows:

Recording date		3 bytes
Disk Identity		10 bytes
Track header	: - $2A - $2A - Loadvector LO/HI - Track length in pages	5 bytes

Every tape block consists of :

Synchro characters $16	50 bytes
Start-of-data character $23	1 byte
Blocknumber 0 ... 255	1 byte
Databytes (506 ASCII characters)	253 bytes
Checksum	2 bytes

The tape information is transiting through a tape buffer of 1/4 Kbytes residing on $CF00. Disk information transites through a disk buffer of 2 Kbytes residing on $C000.

The program can work on either 1 or 2 Mc clock frequency. On 2 Mc, only 5 consecutive memory locations have to be changed:

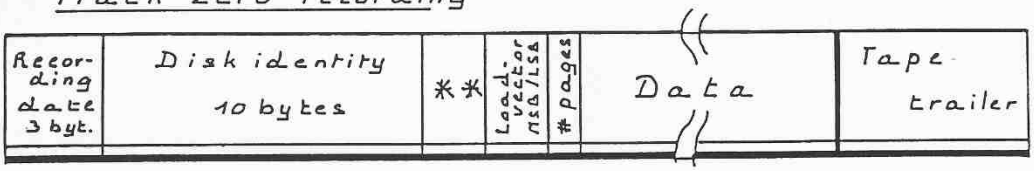
Function	Mem.loc.	1 Mc	2 Mc
High tape delay	$17B0	74	FF
	$17B1	00	00
Low tape delay	$17B2	BA	8A
	$17B3	00	01
Disk write rate	$17B4	12	24

The program normally interfaces to C. Boltjes' VIDEOTEX character generator in order to generate inverse video patterns. If your computer has the normal Octopus character generator Rom, then the location $1809 has to be set on $00. For the Videotex character generator, this value has to be $80.

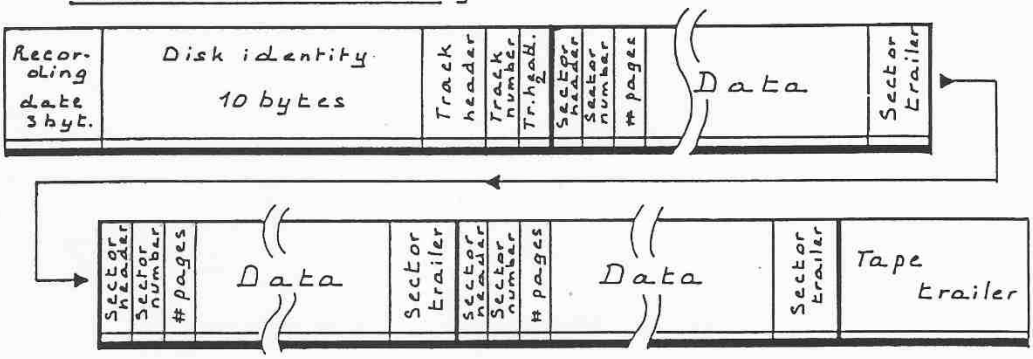
The program is normally intended to run on 40-tracks (single or double-sided) diskettes. If you want to treat 80-track disks, you must first change both locations $1807 and $1808 into $79 and $80. However, the screen layout of the 'Report' will be seriously perturbed, so the report section routines will have to be rewritten.

Information on availabilty of the program can be obtained at the 'COMPUSER' editing office.

Track zero recording



Other tracks recording



M.L. 21/12/87

EC65 Tape backup format

OHIO DOS V3.X SPECIAL FOR THE JUNIOR

Transl.: Dirk Pickee, The Netherlands.

There are some commands which start with the escape-key, followed by an other character. (e.g. ESC 1 = clear screen)
The OHIO - BASIC ignores the escape-key. This has been solved by using PRINT !(n).
[n is the decimal value of the character coming after the ESC - key.]
So, PRINT !(49) equals PRINT CHR$(27);CHR$(49).

By using CTRL 'X', there is a fat chance that the system 'hangs up'. To prevent this, include in the BEXEC the following commands:

```
POKE 9593,234:POKE 9594,234
```

Returning to the 'old' situation can been done by:

```
POKE 9593,20:POKE 9594,24
```

After excecution of the following instructions in the BEXEC, the printer can be switched on and off by using CTRL 'G' [io is set to ,09].

```
POKE 9610,201:POKE 9611,7:POKE 9618,9
```

For those who own DOS V3.X, and are wanting to use the editor; it is located on track 15. Type DISK!"lo 15", excecute it and when the program prompts with 'EDIT ENABLED', type "HALLO" followed by CTRL 'H' and CTRL 'P'. Now the editor has been initialised.

The same with the RUBOUT - function in DOS V3.X, this can be used after excecution of the following instructions:

```
POKE 1394,127:POKE 1419,127
POKE 2820,127:POKE 1386,128
```

Following, some sub-routines in DOS.

$2D73 STRING OUT	JSR $2D73 followed by a textstring, then the string will be printed on the default device. The string must end with $00.
$2761 UNLOAD HEAD	Easy for use, when the system is hanging up, and the head is still loaded. (Some drives won't even open then.)
$267C 1 mSec COUNTER	
$2678 10 mSec COUNTER	

Important addresses:

$2300 (8960D) HIMEM	When at the end of the available RAM some space has to be reserved (e.g. in BASIC), this address contains the HI-address of the last available RAM - area.
$2322 (8994D) OUTPUT IO DEVICE	
$2DC4 (12716D) DIR TRACK	This address contains the track, where to find the directory.
$26A3 (9891D) TRACKACC	Track-to-track acces time in msec, decimal notated.(default 28).
$2E8C	Contents of $31. This is, with 2 processors, $62.