For Superboard 600 and C1P

YE-OSI DOS SUPPLEMENT DOSSUP V5.4

1984 by TB

Revised 2024

```
*** DOSSUP V5.3 (DOS SUPPLEMENT) ***

***    WRITTEN IN 1984 BY TB     ***

***   UPDATE TO MATCH ROM V54    ***

************************************
```

To run YE-OSI DOS 3.54, it is mandatory to

- replace the OSI Boot ROM by EPROM1_V54.ROM ($F800..$FFFF)
- add 5,5k RAM memory to

      $E000-EFFF = (4k)

      $F200-F7FF = (1,5k)

- add a disk controller board from ELEKTOR or an OSI 610 Floppy board
- main memory requirements are min. 8k up to 40k with Hires Mode
- needs minor modifications on OSI 610 board to allow 3.5 & 5.25 inch drives
- Older YE-OSI DOS 3.54 versions required an inverted Write Enable (WE) to prevent data corruption for drives without Head Load mechanism.

Instead, with version 3.54, just remove Drive Select Line resistors R43 and R44 and the Write Enable Line resistor R41 at PB0 on the 610 board. There must be a pull-up resistor installed/enabled on one of the drives attached!

```
*****************************************************************

** Extended BASIC commands after YE-OSI DOS boot (file DOSSUP) **

*****************************************************************
```

With DOSSUP you will provide additional BASIC commands as:

DIR, SEL, DSAV, DFMT, DCHK, DREN for disk file management

EOF, SEQS, SEQW, VER for disk data management

CLG, GDIS, SCR for Text and Low-resolution graphics

PTR for general purpose

DOSSUP will be loaded into memory at location $E900 to 0xEFFF

Remark: Basic command parameters in **[]** are optional.

****

************** QUICK START GUIDE ****************

****

DOSSUP provides some of the following BASIC commands to manage disk drives

EXAMPLES:

Show file directory of currently selected disk drive

➔ **DIR**

Select the second disk drive 2, Boot drive is drive 0

➔ **SEL 2**

Save current Basic program to disk drive 0

➔ **DSAV "TEST",0,0,0**
will save under filename "TEST" onto drive 0 as a read write basic program.

Load back a Basic program from currently selected disk drive

➔ **DLOD "TEST"**
The BASIC program "TEST" will be loaded into memory

Rename the Basic program "TEST" from disk drive 0 into "HELLO"

➔ **DREN "TEST","HELLO",0**
The BASIC program "TEST" will be renamed to "HELLO" on disk 0

Delete a Basic program from disk drive 0

➔ **DREN "HELLO","",0**
The BASIC program "HELLO" will be deleted from disk 0

Writing Bootsector only to disk 0

➔ **DFMT 0,1,0**
The second parameter specifies “Bootsector only”
If you have double sided disk drives, you may update the Bootsector for these specific drives with (last parameter):

➔ **DFMT 0,1,1**

**** DIR ********** COMMAND:

DIR ["String"]

DIR will list the current active file directory. If no "String" is entered all files will be displayed. The length is shown in 256-byte sectors. The listing will pause after 9 file names. To continue press, ENTER, any other key will end the directory listing.

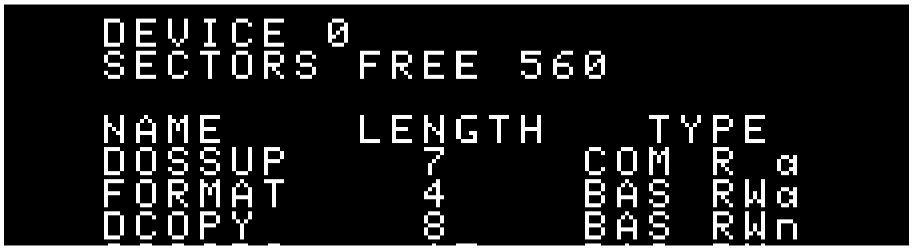
DIR "String" will list all files starting with the "String" letters. For example: DIR"DO" will list all files starting with letters "DO..".

The file type is specified as:

BAS	= BASIC Token Memory loads typically to $0300
COM	= MCODE Binary Code
SEQ	= SEQUENCIAL Data values separated by comma
VAR	= VARIABLE Other type of data

Protection status:

RWn	= Read/Write normal
RWa	= Read/Write autorun
R n	= Read Only normal
R a	= Read Only autorun



```
DEVICE 0
SECTORS FREE 560

NAME      LENGTH    TYPE
DOSSUP      7     COM R a
FORMAT      4     BAS RWa
DCOPY       8     BAS RWn
```

***** SEL ********* COMMAND:

SEL DRIVE

will select "DRIVE" number 0...3. If FAT was changed, FAT is saved before.

<u>IMPORTAT !</u>

Be careful removing and inserting disk into the drive during operation. YE-DOS will not automatically detect, when a new disk is inserted!!

To reload the Disk FAT directory, always type SEL 0..3, to get the current disk content. Otherwise, the disk content may get corrupted.

Only inserted disk will be detected as valid drives. Use DISK command to refresh the drive valid information.

Physical Drive 1

    Side A: >Drive number 0

    Side B: >Drive number 1

Physical Drive 2

    Side A: >Drive number 2

    Side B: >Drive number 3

Remark: Emulation supports drive number 0 and 2 only.

**** DSAV ********** Command:

DSAV "FILNAME", DRIVE, TYPE, PROTECTION

> 1st VERSION <

DSAV stands for Disk Save File and will write a BASIC program, Binary data or other data to disk. "FILENAME" are max 6 characters, longer names are ignored.

When DSAV has been executed, active drive will change to the "DRIVE" number. Check the variable ERR, if any Error occurred.

For the file attributes Type and Protection, see the following valid codes.

***** TYPE codes:**

BAS=0, COM=1, SEQ=2, VAR=3

***** PROTECTION codes:**

RWn=0 RWa=1 ROn=2 ROa=3

Example: DSAV "TEST",2,0,0 will save a BASIC program in memory with the filename "TEST". BASIC programs require no address range information.

If a file already exists and file protection is "Read Only", like 2 or 3, DSAV will fail. In such a case, you have to remove the file protection with DREN (Disk File Rename) first. For example, DREN "TEST","TEST",0,0,0

***** DSAV ********* Command:

DSAV "FILNAME", DRIVE, TYPE, PROTECTION, START, END

> 2nd VERSION <

Types COM, SEQ and VAR are saved in the same way. These file types (Binary Code, Sequential or Variable) are written to the disk like binary data, but with its specific Type identification.

Example: DSAV "TEST", 0, 1, 1, 32768, 32768+1023 will write 1kb binary data to drive 0 as an autorun RW file (RWa). Execution will start after loading the file back at address 32768 in this example.

**** SEQS ********** COMMAND:

SEQS Address

SEQS or Sequence Set Read file pointer will set the pointer to the "Address" in memory.

The purpose is to READ strings or numbers from a given memory location.

These data elements have to be “comma” separated.

The next READ operation will get the stored strings and numbers in a typical DATA read operation.

***** SEQW ********* COMMAND:

NewAddress = SEQW Address, Parameter1, P2, ..

SEQW or Sequence Write data, will put strings or numbers to the Address Pointer. The Command will return the new Address pointing to the next Input.

SEQS and SEQW are used to store string or variables in memory that can be saved or loaded to or back from disk. Memory space selection and pointer control has to be done by software.

***** EOF ********* COMMAND:

[Value=] EOF

EOF will return TRUE after a READ operation, if more data is available.

Example in BASIC:

```
10 AN=61952:EN=AN+256-20

20 RESTORE:PAGE

25 A$="QWERTY"

30 LE=SEQW AN:REM SET START POINTER

35 PRINT"GENERATE SEQ DATA AT $F200"

40 LE=SEQW LE,A$,LE,-1

50 IFLE<ENTHEN40

60 LE=SEQW LE:REM GET END POINTER

70 SL=LE:LE=AN

80 SEQS LE

90 READB$,AD,F

110 PRINT B$;AD;F

120 IF EOF THEN90

130 PRINT

140 PRINT"SEQ DATA SIZE";SL-AN;" BYTES"

150 F$="DATA":DSAV F$,0,2,0,AN,SL

155 IF ERR<>0 THEN PRINT"ERROR";ERR:STOP

160 PRINT"DATA SAVED"

170 DLOD F$

180 IF ERR<>0 THEN PRINT"ERROR";ERR:STOP

190 SEQS AN:REM READ POINTER TO START

200 READB$,AD,F

210 PRINT B$;AD;F

220 IF EOF THEN200

230 PRINT"DATA LOADED BACK"

240 DREN F$,"",0:REM DELETE FILE

250 IF ERR<>0 THEN PRINT"ERROR";ERR:STOP
```

This program will generate a data parameter stream of a string and two numbers at $F200 (Line 25..50). The sequential data stream is than stored as "DATA" SEQ file to disk. Afterwards read back and displayed using the BASIC READ statement (Line 190..220)

***** VER ********* COMMAND:

VER

VER will return the disk DOS version of the currently selected drive
For Example, VER will return **"YE-OSI DOS 3.54"** on the current revision.

****** DFMT ******** COMMAND:

DFMT DRIVE, SECTION, DISKTYPE

DFMT stands for Disk Format. The Command will format a disk "DRIVE".

DFMT will be executed without further prompt or question.

Please make sure, you have no valuable data on the disk to format.

"DRIVE" number has to be between 0...3.

"SECTION" defines, if the whole disk (0) or only the disk BOOT sector (1) has to be formatted.

"DISKTYPE" defines, if we have a single (0) or double-sided disk (1).

IMPORTANT !

DFMT will only create “blank” diskettes, without content. Use the Basic program FORMAT.BAS, to create fully bootable diskettes. DFMT will immediately start, there will be no warning. Both sides on double-sided will be formatted.

EXAMPLE: DFMT 2,0,0 will format disk 2 as single sided.

FORMAT.BAS program example:

```
10 REM DISK FORMAT UTILITY

20 TA=64768:PAGE:PRINT"UTILITY FOR 40/80 TRACK DRV":PRINT

25 PRINT"FORMAT DRIVE NUMBER ?":T=CALLTA,0:IFT<48 OR T>51THEN END

30 PRINT:PRINT"INSERT DISK IN DRIVE ";CHR$(T)

35 DR=T-48:GOSUB800:PRINT

40 PRINT"PRESS(Y),WHEN READY:":T=CALLTA,0:IFT<>89THEN END

50 PRINT:PRINT"FORMATTING DISK";DR

60 DFMT DR,0,SS:REM FULL FORMAT DISK

80 IF ERR>0 THEN PRINT"FORMAT FAILED, ERROR NUMBER ";ERR

90 REM ---------------------

100 PRINT"TRANSFER DOS TOOLS TO DRIVE";DR

105 PRINT"PRESS(Y),WHEN READY:":T=CALLTA,0:IFT<>89THEN160

110 SELDR:PRINT:PRINT "TRANSFER DOS EXTENSION TO DRV";DR

115 DSAV"DOSSUP",DR,1,3,233*256,240*256-1: REM TYPE MCODE,AUTORUN

120 IF ERR>0 THEN PRINT "TRANSFER FAILED WITH ERROR";ERR

130 PRINT:PRINT "TRANSFER FORMAT.BAS TO DRV";DR

140 DSAV "FORMAT",DR,0,1:PRINT:REM TYPE BASIC,R/W AUTORUN

150 IF ERR>0 THEN PRINT "TRANSFER FAILED WITH ERROR";ERR

160 DIR:IF ERR>0THEN PRINT"DOS FORMAT FAILED"

170 SEL0:IF ERR>0THEN PRINT"DOS DRIVE 0 FAILURE":END

200 END

800 REM CHECK FOR SS OR DS DRIVES

810 SS=0:IF DR<2 THEN 840

820 IF PEEK(57364+3)<>255 THEN SS=1

830 GOTO 850

840 IF PEEK(57364+1)<>255 THEN SS=1

850 IF SS<>1 THEN RETURN

860 PRINT:PRINT"SINGLE(S) OR DOUBLE(D) SIDED DRIVES ?":T=CALLTA,0

870 IFT=68 OR T=83THEN880

875 GOTO860

880 IFT=83 THEN SS=0

885 RETURN
```

***** DCHK ********* COMMAND:

DCHK "FILENAME"

DCHK stands for Disk Check/Verify. The Command will Verify a saved file "FILENAME" with its original location in memory.

It will also check, if the file exists on the disk.

If the filename is not on the disk, ERR number 9 is returned.

In case the filename exists, the file content is compared to memory.

If this verification fails, ERR number 11 is returned. Elsewhere ERR 0 is returned.

Verification or Check is done on the current selected drive only.

**** DREN ********** COMMAND:

DREN "FILENAME", "NEW FILENAME", DRIVE [, TYPE, PROTECTION]

DREN stands for Disk File Rename or Delete. The Command will rename a saved file "FILENAME", change its type or protection status. With an empty new filename, the file will be deleted.

Keep in mind, that Read Only protected files cannot be deleted, before the protection has been changed.

DREN Version 1 (5 parameters):

DREN "FILNAME", "NEW FILENAME", DRIVE, TYPE, PROTECTION

This will change Filename and/or file attributes. Using the same filename will only change attributes. For example, changing DOSSUP from RO to RW:

DREN"DOSSUP","DOSSUP",0,1,0 (Drive 0, COM Type, RW Protection)

DREN Version 2 (3 parameters):

DREN "FILENAME", "", DRIVE

This will delete the file "FILENAME" on drive "DRIVE". Data is still on the disk, but the directory entry filename is cleared.

File recovery is possible.

**** SCR ********** COMMAND:

SCR X (0...31/63), Y(0...15/31), DATA, [DATA,...]

SCR will write DATA (Strings or Variables) to the screen at position X,Y.

The left bottom corner of the screen is at SCR 0,0,"x". Range depends on machine graphic capabilities like 32x32, 64x16 or 64x32 characters.

***** PTR ********* COMMAND:

VAL=PTR(VARIABLE)

PTR will return the Pointer to the variable content as a 16bit address value. VARIABLES can be a numeric variable like AD=PTR(A) or a string variable AD=PTR(A$) or a pointer to an array like AD=PTR(M(0)). PTR(M(0)) will return a pointer direction to the first byte of array M().

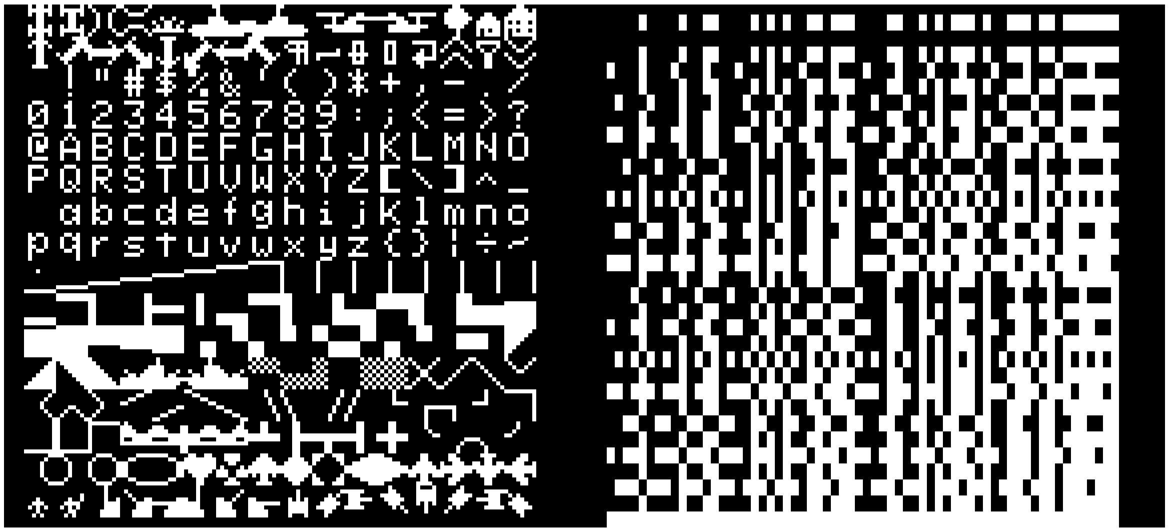
This can be used to reserve memory space and to place code or data into the array to peek or poke or read or write to the disk.

REMARK: Each array element occupies 4 bytes. DIM M(255) will reserve 1kB.

***** CLG ********* COMMAND:

CLG NUMBER

CLG will clear text or low-resolution graphics or enable/disable text/low resolution mode. Low resolution mode is a 128x32 pixel graphic displayed in the upper half of the text screen as a kind of split screen.
This was done by a modified character ROM of 4kB instead of 2kB. The graphic part of the ROM is enabled by the ACIA RTS line and the upper half display interval.



C1P lower 2kB character ROM   Upper Low-Res 2kB character ROM

placed into a pin compatible 4kB EPROM with one gate logic chip.

CLG 0: DISABLE LOW-RES MODE (same as CLG without parameter)

- Standard text mode (RESET (F12) will do the same)

CLG 1: ENABLE LOW-RES MODE

- TOP part of the low-res display (128x32 or 128x64 pixel)
  in half screen mode

CLG 2: Clear BOT half with "20"

- Clear text part

CLG 3: Clear TOP half with "00"

- Clear graphic part

also see PAGE command

- Clear entire text screen, when back in text mode

***** GDIS ********* COMMAND:

GDIS X (0...127), Y(0...16/32/63), MODE (depending on display mode)

When the low-resolution graphics mode is enabled, GDIS will plot dots or lines on the screen or will clear the same if required.

The Y coordinate of low-res section starts at Y=0 or Y=32 (upper half of screen, depending on display mode).

A graphic section of 128x32 or 128x64 is not much, but it is a simple add-on to allow fast pixel graphics in combination with text output on the C1P machine.

And it uses only the standard screen memory.

The pixel origin is at the left bottom corner of the upper low-res screen.

GDIS X, Y, 1	- Plot at X,Y a white dot
GDIS X, Y, 2	- Draw a line from the last coordinate to this one.
GDIS X, Y, 0	- Plot at X,Y a black dot (clear)
GDIS X, Y, 3	- Draw a black line (clear) to the new coordinate.

See GRDEMO.BAS program example:

```
5 OY=32:IFPEEK(65506)<>0THENOY=0

6 F=1:IFPEEK(65506)<>0THENF=2

7 OM=OY+16*F:OX=OY+32*F-1

8 CLG1:CLG3

9 CLG2:SCR5,5,"LINE SET&RESET"

10 GDIS0,OM-1,1:GDIS127,OM-1,2

11 FOR L=0 TO 1

12 FOR R=0 TO 127 STEP 2

15 IFL=0 THEN GDISR,OX,1:GDISR,OY,2

16 IFL=1 THEN GDISR,OX,0:GDISR,OM,3

17 IFL=1 THEN GDISR,OM-2,0:GDISR,OY,3

18 NEXT R:NEXT L

20 CLG3:IFPEEK(57088)=222 THEN 200

25 CLG2:SCR5,5,"LINE MESH"

30 FOR R=4TO 127 STEP 15

40 FOR S=4 TO 127 STEP 15

50 GDISR,OX,1:GDISS,OY,2

60 NEXT S:NEXT R

70 CLG3:IFPEEK(57088)=222 THEN 200

110 CLG2:SCR5,5,"BOXES"

112 F=1:IFPEEK(65506)<>0THENF=2

115 FOR L=0 TO 3:FOR S=0 TO 1

120 FOR R=1 TO 10*F STEP 2*F

130 GDIS64-7*R/F,OM-R,1-S:GDIS64+7*R/F,OM-R,2+S

140 GDIS64-7*R/F,OM+R,1-S:GDIS64+7*R/F,OM+R,2+S

150 GDIS64-7*R/F,OM-R,1-S:GDIS64-7*R/F,OM+R,2+S

160 GDIS64+7*R/F,OM+R,1-S:GDIS64+7*R/F,OM-R,2+S

190 NEXT R:NEXT S:NEXT L

195 CLG3:IFPEEK(57088)<>222 THEN 7

200 CLG0:PAGE:PRINT"READY"
```

In this program example, placing text by the SCR command and drawing and removing low-res lines by the GDIS command is shown.

**************** Supporting Programs on disk (1 of 3)**

FORMAT.BAS

This BASIC program will format a diskette in the chosen drive (single and double sided only) and transfer the BOOT sector, DOSSUP and FORMAT.BAS to a new disk.

**************** Supporting Programs on disk (2 of 3)**

DCOPY.BAS

This BASIC program will copy all tracks of a diskette in Drive 1 (single sided only) to Drive 2.

Basic Code:

```
5  PAGE:PRINT"* DISK COPY UTILITY *":PRINT"DRIVE 0 TO 1,2 OR 3":PRINT
6  FA=62560:ID=62464:E=0:TM=61952:TA=64768
7  PRINT"TARGET DRIVE NUMBER ?":T=CALLTA,0:IFT<49 OR T>51THEN END
8  DR=T-48:GOSUB800:PRINT
10 TA=64768:PRINT"INSERT SOURCE DISK IN DRIVE 0":PRINT
15 PRINT"PRESS (Y), WHEN READY":T=CALLTA,0:IFT<>89THEN END
20 SEL0:IF ERR>0 THEN PRINT "DRIVE 0 NOT READY":END
25 SELDR:IF ERR>0 THEN PRINT "DRIVE";CHR$(DR+48);" NOT READY":END
30 FOR R=FA TO FA+(70*13) STEP 13: REM CHECK IF DRV IS EMPTY
40 IF PEEK(R)<>0 THEN E=1
50 NEXT R
60 IF E=0 THEN 100
70 PRINT:PRINT ">> DRIVE ";CHR$(DR+48);" IS NOT EMPTY"
80 PRINT"PRESS (Y), TO FORMAT DRIVE":T=CALLTA,0:IFT<>89THEN END
90 PRINT"FORMATTING, PLS WAIT":DFMT DR,0,0
100 REM ****** COPY DISK 0 TO TARGET DISK *****
110 SEL0:IF ERR>0 THEN PRINT "DRIVE 0 NOT AVAILABLE":END
130 FOR R=2 TO 79: REM SEC_T,TRK_T first, E022/E023 second
140 IF PEEK(ID+R)=0 THEN 310: REM TRACK NOT USED
150 PRINT"T="R;"-";
155 BI=1:FOR S=0 TO 7
160 POKE57381,255: REM Finish with E022/23
170 POKE57380,0: REM TAKE NEXT SECTOR
180 POKE57376,0: REM DRIVE 0 SOURCE
190 POKE57374,SS: REM SINGLE SIDED
200 POKE240,0:POKE241,242: REM ADR=$F200
210 POKE57372,1: REM LENGHTH=256 Bytes
220 POKE236,R:POKE237,S: REM Start TRK2..79,SEC
225 IF (PEEK(ID+R) AND BI)=0 THEN 240:REM SKIP EMPTY SECTORS
230 DISK1:IF ERR>0 THEN PRINT "READ SECTOR FAILED":END
240 POKE236,R:POKE237,S: REM Start TRK2..79,SEC
250 REM TRK2,SEC0 info is given by last READ operation
```

```
260 POKE240,0:POKE241,242: REM ADR=$F200
270 POKE57372,1: REM LENGHTH=256 Bytes
280 POKE57376,DR: REM DRIVE DESTINATION
285 IF (PEEK(ID+R) AND BI)=0 THEN 305:REM SKIP EMPTY SECTORS
290 DISK2: IF ERR>0 THEN PRINT "SECTOR SAVE FAILED":END
300 PRINT S;
305 BI=BI*2:NEXT S:PRINT
310 NEXT R
500 DISK6:IF ERR>0 THEN PRINT "UPDATE FAT TO TARGET FAILED":END
510 POKE57381,0: REM Finish with 00 00
520 POKE57380,255: REM Take next free sector
530 SELDR:IF ERR>0 THEN PRINT "DRIVE"DRIVE ";CHR$(DR+48);" FAILED":END
540 SEL0:IF ERR>0 THEN PRINT "DRIVE 0 NOT AVAILABLE":END
600 PAGE:PRINT"COPY BOOT SECTOR AND FAT"
610 POKE57382,255: REM BOOT SECTOR FORMAT DISK
620 POKE57376,DR: REM TARGET DRIVE
630 POKE57374,SS: REM SINGLE SIDED
640 DISK3: REM WRITE BOOT SECTOR
650 IF ERR>0 THEN PRINT "BOOT SECTOR FAILED":END
660 SEL0:IF ERR>0 THEN PRINT "DRIVE 0 NOT AVAILABLE":END
670 PRINT "READY"
700 SELDR:DIR:IF ERR>0 THEN PRINT "DOS FORMAT FAILED"
710 SEL0:IF ERR>0 THEN PRINT "DOS DRIVE 0 FAILURE":END
720 END
800 REM CHECK FOR SS OR DS DRIVES
810 SS=0:IF DR<2 THEN 840
820 IF PEEK(57364+3)<>255 THEN SS=255
830 GOTO 850
840 IF PEEK(57364+1)<>255 THEN SS=255
850 IF SS<>255 THEN RETURN
860 PRINT:PRINT"SINGLE(S) OR DOUBLE(D) SIDED DRIVES ?":T=CALLTA,0
870 IFT=68 OR T=83THEN880
875 GOTO860
880 IFT=83 THEN SS=0
885 RETURN
```

*************** Supporting Programs on disk (3 of 3)**

FCOPY.BAS

This BASIC program will copy a single file form diskette in Drive 1 to Drive 2. Read Only files will be transferred, if confirmed.

Basic Code:

```
5 REM FILE COPY UTILITY TO OTHER DRIVE
10 CLEAR:DR=PEEK(57376):IF DR=0 THEN DN=2
20 IF DR=2 THEN DN=0:REM GET OTHER DRIVE NUMBER
30 PAGE:PRINT"FILE COPY UTILITY FROM";DR;"TO";DN
40 INPUT"ENTER FILENAME ";NA$:IF NA$="" THEN 40
50 FA=62560:ID=62464:E=0:TM=61952:TA=64768:AD=245
```

```
60 DCHK NA$:REM TEST, IF FILENAME EXIST

65 FE=PEEK(AD)+256*PEEK(AD+1)

70 IF ERR=9THEN PRINT"FILENAME NOT FOUND, TRY AGAIN":PRINT:GOTO40

75 SEL DN:IF ERR>0 THEN PRINT "DRIVE";DN;"IS NOT AVAILABLE":END

80 DCHK NA$: IF ERR=9 THEN 110

90 PRINT"FILE EXIST,OVERWRITE IT (Y/N)?": T=CALL TA,0

100 IFT<>89THEN PRINT"QUIT":END

110 SEL DR:IF ERR>0 THEN PRINT "DRIVE";DR;"IS NOT AVAILABLE":END

120 R=PEEK(FE+6):S=PEEK(FE+7)

125 S1=PEEK(FE+8):S2=PEEK(FE+9):S=S1+256*S2

130 E1=PEEK(FE+10):E2=PEEK(FE+11):E=E1+256*E2

135 TP=PEEK(FE+12):Z=FRE(0):IF Z<0 THEN Z=65536+Z

140 IF (E-S+255)<Z THEN 150

145 PRINT"NOT ENOUGH FREE MEMORY TO COPY FILE !":GOTO380

150 DIM M((E-S+256)/4)

152 MS=PTR(M(0)):ME=MS+(E-S):REM GET MEMORY ADDRESS

155 POKE(FE+8),MS AND 255:POKE(FE+9),INT(MS/256)

160 POKE(FE+10),ME AND 255:POKE(FE+11),INT(ME/256)

165 POKE(FE+12),16: REM SIMPLE BINARY FILE

170 DLOD NA$

180 POKE(FE+8),S1:POKE(FE+9),S2

190 POKE(FE+10),E1:POKE(FE+11),E2

200 POKE(FE+12),TP: REM RESTORE ORIGINAL

210 IF ERR>0 THEN PRINT "LOADING FILE ERROR":GOTO380

220 SEL DN:IF ERR>0THEN PRINT"DRIVE";DN;"IS NOT AVAILABLE":END

300 IF (TP AND 15)>=2 THEN DREN NA$,NA$,DN,1,0

305 DSAV NA$,DN,1,0,MS,ME

310 IF ERR>0THEN PRINT"SAVING FILE FAILED, ERROR";ERR:END

320 DCHK NA$:IF ERR>0 THEN PRINT "VERIFY FAILED, ERROR";ERR:END

330 FE=PEEK(AD)+256*PEEK(AD+1)

340 POKE(FE+8),S1:POKE(FE+9),S2

350 POKE(FE+10),E1:POKE(FE+11),E2

360 POKE(FE+12),TP: REM RESTORE ORIGINAL

370 POKE 57375,1:DISK 6:REM FORCE SAVING FAT

380 SEL DR:CLEAR
```

Listing

```
;
;                   (c) Copyright TB 2023
;
;         File:             D_E800_EFFF.bin                DOS Supplement
;
;         Date:             Dec  2024
;
;         CPU:              MOS Technology 6502 (MCS6500 family)
;
;         Modified for C1P or UK101 with 32x32 or 64x16 screen
;         Needs EPROM1_Vxx YE-DOS System ROM
;         Uses BASIC ROM subroutines
;
;
; DOS SUPPLEMENT:
; These are extensions in Basic, that provide additional commands
; Code is located at $E900 to $EFFF, Basic code start is normal at $0300
; Screen Size taken from $FFE1 <32 or >32 and $FFE2 0 or 1 for 2K or 4k
;

D_VERS          = 54            ; Version number
ORG_POS         = $E900
FAT_D           = $F400         ; FAT Memory area
FAT_S           = FAT_D+$60     ; FAT start of name table
FATVER          = FAT_D+$50     ; VERSION TEXT STRING
FATCHANGE       = $E01F         ; FAT Change info flag
PAR_STOR        = $0230         ; Storage if command parameters (7 Bytes)
BASIC_EXT       = $022C         ; Basic extension Vector

FreeM           = $F300         ; Free Memory area (moved to $F300 up)
DOS_PARAM       = $E010         ; DOS Parameter table
DOS_E022        = $E022         ; Low FAT File Name Pointer or Free sector count
DOS_E023        = $E023         ; High FAT Mane Pointer

ACIA_S          = $F000         ; SERIAL ACIA Control Port for turning low res graph on/off

BASIC_16_FLOAT  = $AFC1         ; Convert Fixed Point to Floating Point
BASIC_OUT       = $A8E5         ; BASIC Character output

HORZ_SIZE       = $FFE1         ; <32 or >32 will indicate horizontal screen resolution (32 or 64)
VERT_SIZE       = $FFE2         ; 0 will indicate 2k, 1 is 4k Screen memory size

STOP  = 3                   ; DEBUGGING STOP CODE

        .org    ORG_POS

        lda     #DOSSUP&255
        sta     BASIC_EXT
        lda     #DOSSUP>>8
        sta     BASIC_EXT+1
        rts

VER:    .DB "YE-OSI DOS 3."
        .DB 48+D_VERS/10
        .DB 48+D_VERS%10
```

```
FINAME: .DB $00, $00

STAR:      .DB "*", $00

CODETBL:
      .DB $42, $41, $53, $00    ;"BAS"
      .DB $43, $4F, $4D, $00    ;"COM"
      .DB $53, $45, $51, $00    ;"SEQ"
      .DB $56, $41, $52, $00    ;"VAR"
PROTTBL:
      .DB $52, $57, $6E, $00    ;"RWn"
      .DB $52, $57, $61, $00    ;"RWa"
      .DB $52, $20, $6E, $00    ;"R n"
      .DB $52, $20, $61, $00 ;"R a"


GDPIX:     .db $01, $02, $04, $08, $10, $20, $40, $80          ; 8 Bit 4x2 1st is left

TSPACE:
      .db "   ", $00                          ; SPACES
TDEV:   .db $0D, $0A, "DEVICE", $00     ; DEVICE
TSEC:   .db "SECTORS FREE", $00                       ; SECTORS FREE
TNAME:  .db $0D, $0A, $0A, "NAME    LENGTH   TYPE", $0D, $0A, $00                 ; NAME  LENGTH   TYPE



DOSSUP: ldy          #$FF                                 ; ********** DOSSUP ENTRY POINT *******
        ldx          #$00

L0311:
        iny
        dex
L0313:
        lda          TOKEN-256,x                         ; BASIC TOKEN CHECK
        beq          L033F
        sec
        sbc          ($C3),y
        beq          L0311
        cmp          #$80
        beq          L032D
        ldy          #$00
L0323:
        dex
        lda          TOKEN-255,x
        bpl          L0323
        dex
        dex
        bne          L0313
L032D:
        jsr          $A70F                  ; BASIC:
        pla
        pla
        pla
        pla
        lda          TOKEN-257,x          ; Vector to DOSSUP Basic token code
        pha
        lda     TOKEN-258,x
        pha

L0336:
        inc     $C3             ; Moved from ROM to here
        bne     L033C
        inc     $C4
L033C:
        jmp     $00C2           ; Return to Adress $00C2

;
L033F:
        ldy     #$01
        jsr     $A70F           ; BASIC:

        ldx     $BF             ; continue Basic Interpreter
        jmp     L0336

;                               ; ************* PRINT 3x SPACES TO BASIC **********
L0347:
        lda     #TSPACE&255
L0349:                          ; *** PRINT TEXT in same segment (A) ***
        ldy     #TSPACE>>8
        JMP     $A8C3           ; BASIC PRINT 3x Space TEXT


;
L034E:                          ; *************** SUB EVALUATE Byte/Word Parameters *****
        jsr     $AC01           ; BASIC check Next parameter

L0351:                          ; **** Evaluate single Integer parameter ***
        sty     $C0             ; Pointer Counter to Parameter storage
        cpy     #$03            ; Already 3 Paramenetrs
        BCS     L036D           ; Jump, if 3 or more
                                ; Evaluate Integers Parameters 1,2,3
        JSR     $B3AE           ; BASIC ROM: EVALUATE 8BIT EXPRESSION and convert to byte
                                ; Integer value in (X)
        ldy     $C0
L035C:
        txa
        sta     PAR_STOR,y      ; Store Integer Parameter
        iny
        jsr     $00C2           ; Next BASIC value
        cmp     #$2C            ; More Parameters ?
        bne     L037D           ; Test for "," (jump and RETURN, if no more parameters)
        jsr     $00BC           ; BASIC: Advance to next parameter
        bne     L0351           ; Loop back for next parameter
L036D:
        jsr     $AAAD           ; ROM BASIC: EVALUATE 16BIT EXPRESSION, MAKE SURE IT IS NUMERIC
        jsr     $B408           ; CONVERT TO A 16-BIT VALUE
                                ; Result in (A) and (Y)
        tax
        tya
        ldy     $C0
        sta     PAR_STOR,y      ; Store Lower Byte of 16Bit
        iny
        bne     L035C           ; Always jump to store Higher Byte in (X)
```

```
L037D:
        rts
;
L037E:                          ; ***** Get String parameters to Stack ******

        bne     L0383           ; Command without parameters ?
L0380:

        jmp     $AE88           ; STOP WITH ILLEGAL QUANTITY ERROR
;
L0383:
        pla                     ; Get caller Return address to $9C/9D
        sta     $9C
        pla
        sta     $9D             ; Remember Return address

        jsr     $AAC1           ; ROM BASIC, Evaluate expression
        bit     $5F             ; Check for String
        bpl     L0380           ; ERROR of not a Sting

        jsr     $B2B3           ; ROM BASIC, Release string
                                ; Pointer in 71/72, length in A
                                ; Points to String in BASIC code or memory
L0398:
        sta     $BF
        pha                     ; Pushing A(lenth) must be <>0
        tya
        pha                     ; Pushing Y(High vector)
        txa
        pha                     ; Pushing X(Low vector)
        lda     $9D
        pha                     ; Restore Return Adr
        lda     $9C
        pha                     ; Restore Return Adr
        lda     $BF
        rts
;
L03A4:                          ; ***** SUB Process Parameter: Drive *****
        lda     #$00            ; Analyse input parameter for Drive number
        sta     $E027           ; Set ERROR to "0"
        lda     PAR_STOR        ; First Paramneter is DRIVE number
        cmp     $E020           ; Compare with actual Drive number
        beq     L03C3           ; Jump, if the same
        cmp     #$04            ; Check for range 0...3
        bcc     L03B8           ; Jump, if <4
        jmp     $AE88           ; FC-Error
;
L03B8:                          ; New valid Drive number selected
        pha
        jsr     L03CD           ; DOS_WRITE_FAT(6), if FAT needs update
        pla
        sta     $E020           ; Store new drive number
        jsr     L03C7           ; DOS_READ_FAT(7), from new drive
L03C3:
        lda     $E027           ; Hold Error value in (A)
        rts
```

```
;                                       ***************** DIR ******************
DIC:        bne         DIC1            ; Jump on search name
            lda         #$01
            pha                         ; Pushing A(lenth)
            lda         #STAR>>8
            pha                         ; Pushing Y(High vector)
            lda         #STAR&255
            pha                         ; Pushing X(Low vector)
            jmp         DIC2
DIC1:
            jsr         L037E           ; Get String parameters to Stack
DIC2:
            lda         #TDEV&255       ; Print DEVICE
            jsr         L0349           ; PRINT TEXT in same segment (A)
            ldx         $E020
            lda         #$00
            jsr         $B95E           ; BASIC Print value in X,A (device no)
            jsr         $A86C           ; BASIC: Do PRINT CR,LF ?

            lda         #TSEC&255       ; PRINT SECTOR
            jsr         L0349           ; PRINT TEXT in same segment (A)

            lda         #$00            ; SET lenth to 0 (FREE SPACE function)
            sta         $94
            jsr         L03E0           ; ROM: DOS SEARCH FILE returns disk free space
            ldx         $E022           ; Free Space into X,A
            lda         $E01D
            jsr         $B95E           ; BASIC Print value in X,A (sectors free no)

;           jsr         L0347           ; Print 3x SPACES

            lda         #TNAME&255      ; PRINT LF NAME
            jsr         L0349           ; PRINT TEXT in same segment (A)

            inc         $E01D           ; Correct Preset for HL and Motor off

            lda         #FAT_S&255      ; Start of FAT ($F460)
            sta         $97
            lda         #FAT_S>>8
            sta         $98             ; to pointer FAT address pointer $97/98
            lda         #$08
            sta         $C0
            pla                         ; ex: $FD / $FF
            sta         $9E
            pla                         ; ex: $7F / $7F / $77
            sta         $9F             ; Pointer to DIR string to $9E/9F
            pla                         ; String length >0 ?
            sta         $BF             ; Remember length of DIR string
            bne         L041B
L041A:
            rts                         ; BASIC: String "", just return
;
L041B:
            ldy         #$00
L041D:
            lda     ($9E),y
            cmp     #$2A            ; is "*"
            beq     L043F           ; jump to matching name in FAT
            cmp     ($97),y         ; Compare first char of FAT name
            bne     L0427           ; jump to not matching name
            iny
            cpy     $BF
            bcc     L041D
            bcs     L043F           ; jump to matching name in FAT
L0427:
            lda     #$0D            ; Next FAT entry + 13
            clc
            adc     $97             ; Add 13 to FAT address pointer
            sta     $97
            bcc     L0432
            inc     $98
L0432:
            cmp     #$FB            ; END OF FAT REACHED ?
            bne     L041B           ; Loop back search name
L0436:
            ldx     $88             ; Get ??
            inx
            bne     L041A           ; Jump, if was <> FF ,return
            jmp     $A86C           ; BASIC: Do PRINT CR,LF ? and return
;
L043F:
            ldy     #$00            ; Matching first Character in FAT
L0441:
            lda     ($97),y         ; Load last name char from FAT
            beq     L0427           ; Empty FAT entry found with "*", loop back

            jsr     OUTVEC          ; ROM Output
            iny
            cpy     #$06
            bcc     L0441
            jsr     L0347           ; Print 3x spaces
            ldy     #$0B
            lda     ($97),y         ; Calculate file length
            ldy     #$09
            sec
            sbc     ($97),y
            TAX
            INX
            pha                     ; Sector value to stack
            LDA     #$00
            JSR     $B95E           ; PRINT Length of (A) and (X)
            PLA
            CMP     #$09            ; Value >=10
            BCS     L043FX
            lda     #$20            ; PRINT Single Space
            jsr     BASIC_OUT       ; PRINT TEXT in same segment (A)

L043FX:
            jsr     L0347           ; 3x SPACE
            ldy     #$0C
            lda     ($97),y         ; Get File Type
            pha
            lsr
            lsr
            clc
            adc     #CODETBL&255
            jsr     L0349           ; PRINT TEXT in same segment (A)
            jsr     $A8E0           ; PRINT LENGTH
            pla
            and     #$03
            asl
            asl
            clc
            adc     #PROTTBL&255
            jsr     L0349           ; PRINT TEXT in same segment (A)
            jsr     $A86C           ; BASIC Some kind of Print Return
            dec     $C0
            bpl     L0488           ; END of FAT ?
            jsr     INVEC           ; ROM Get Key every 8 lines
            cmp     #$0D
            bne     L0436
            sta     $C0
L0488:
            jmp     L0427
;
L048B:                          ; ********** Copy FILE NAME to FAT ******
            inc     FATCHANGE       ; Mark Change drive
            lda     PAR_STOR+1      ; Get TYPE
            pha
            asl
            asl
            asl
            asl
            ora     PAR_STOR+2      ; OR with PROTECTION
            ldy     #$0C
            sta     ($F5),y         ; STORE TO FAT
            ldy     #$05
L049F:
            cpy     $94             ; Compare to length
            bcc     L04A6           ; Jump if smaller
            lda     #$20            ; Fill Name wit SPACE
            bne     L04A8
L04A6:
            lda     ($A2),y         ; If not, copy string name to FAT
L04A8:
            sta     ($F5),y
            dey
            bpl     L049F           ; Loop for 6 Parameters
            pla                     ; Get back File Type
            rts


;                               ; ************* DSAV STORE ***************
STR:    jsr     L037E           ; Get String parameters to Stack
            jsr     L034E           ; EVALUATE Byte/Word Parameters
            jsr     L03A4           ; Process Parameter: Drive
            beq     L04D5           ; Jump, if NO ERROR
L04D1:
            pla                     ; Clear Stack form String Discriptor
```

```
          pla
          pla
          rts                          ; and return
;
L04D5:
          lda       $C0                ; Number of parameters found of min 3
          cmp       #$02               ; String plus 2 parameter minimum.
          bcs       L04DE              ; jump if >=3
L04DB:
          jmp       $AE88              ; F-Error
;
L04DE:
          ldx       PAR_STOR+1         ; 2nd parameter (File Type)
          BEQ       L04E9              ; Jump, if Command type is BASIC (0)?

          cmp       #$03               ; Check number of parameters found for Type 1,2,3,..
          BCC       L04DB              ; Jump if number of parameters found <4 to Error
                                       ; String plus 3 parameter minimum (drive,type,prot)
L04E9:
          pla                          ; Continue with 4 or more parameters or BASIC
          sta       $9E                ; String name adress to $9E/9F
          tay
          pla
          sta       $9F
          sta       $A3                ; Store High address to A3
          pla
          sta       $C0                ; String length to $C0
          jsr       L07FA              ; FAT name Search (1 sector)
          beq       L050C              ; Check if found, jump if name exist in FAT

          lda       #FINAME>>8         ; Empty file name vector
          sta       $A3                ; *** SEARCH FOR "00" NAME (EMPTY)? ***
          ldy       #FINAME&255
          lda       #$01               ; Single byte is enough to find empty entry
          jsr       L07FA              ; FAT name Search (1 sector)
          beq       L0519              ; Check if found, jump if name exist in FAT
L0504:
          sta       $E027              ; Remember ERROR 9
          rts                          ; Return if not found (no Free entry ?)
;
L0508:                                 ; Leave, if READ ONLY FILE FOUND
          lda       #$0F
          bne       L0504              ; ERROR 15 - FILE is WRITE PROTECTED
;
L050C:                                 ; *** FILE NAME ALREADY EXIST IN FAT ***
          ldy       #$0C
          lda       ($F5),y            ; Get File Type
          and       #$03
          cmp       #$02               ; Check for READ ONLY
          bcs       L0508              ; Jump if >=2 (means READ ONLY)

          lda       #$00
          sta       $E01D              ; HL and Motor keep on
          jsr       L05D2              ; DELETE CURRENT SELECTED FILE
                                       ; Continue and overwrite same name.
L0519:
          lda       #$00
```

```
        sta     $E01D           ; HL and Motor keep on

        jsr     L05C2           ; COPY NAME and TYPE to FAT
        bne     L056F           ; Check if Type is "BASIC (0)"
        ldx     $7A             ; Type is BASIC
        ldy     $79             ; Get Basic Start -1
        bne     L0525
        dex
L0525:
        dey                     ; Basic -1 is in (X),(Y)
        tya
        ldy     #$08
        sta     ($F5),y         ; Writes to FAT Directory Adress parameter
        txa
        iny
        sta     ($F5),y         ; Start / End Adress
        lda     $7B
        iny
        sta     ($F5),y         ;
        lda     $7C
        iny
        STA     ($F5),y         ;

L0539:                          ; *** and write file to disk
        ldy     #$0C            ; Pointer to Type
L053B:
        lda     ($F5),y         ; Copy all to File discriptor block EE-F4
        sta     $E8,y
        dey
        cpy     #$07
        bne     L053B
        sec
        lda     $F3
        sbc     $F1
        sta     $E01C           ; Length Calc
        lda     $F0
        cmp     $F2
        bcs     L0556
        inc     $E01C           ; +1
L0556:
        lda     #$FF
        sta     $E024           ; Search free (FF default) for storage
        jsr     L03CA           ; DOS Write File
        ldy     #$06
        sty     $E024           ; Search free back to normal

        jsr     L0581           ; Check for ERROR, Read FAT back

        lda     FreeM           ; Get fist Value form Table
        sta     ($F5),y         ; Store Start Track to FAT
        lda     FreeM+1
        iny
        sta     ($F5),y         ; Store Start sector to FAT

        inc     $E01D           ; and back to HL and motor off

        jmp     L03CD           ; Write FAT (6) after file save
```

```
;
L056F:
        cmp     #$04            ; Type is "Others" 1,2,3..
        bcs     L0581           ; Return if Type >=4
        ldy     #$08
L0575:
        lda     PAR_STOR-5,y    ; Get Start Address and End Address
        sta     ($F5),y
        iny
        cpy     #$0C
        bne     L0575
        beq     L0539           ; Always jump to



L0581:                          ; *** Check for ERROR, Read FAT back ***
        lda     $E027           ; Check  Error number
        beq     L058E           ; Return, if no ERROR
        pha
        jsr     L03C7           ; On Error READ FAT back (7)
        pla
        sta     $E027           ; Recover Write Error
        pla
        pla                     ; Remove caller address
L058E:
        rts


;                               ; ************ DREN Rename **************
CHANGE: jsr     L037E           ; Get 1st String parameters to Stack
        jsr     $B117           ; Keep string in memory because of 2nd string
        jsr     $AC01           ; ROM BASIC, Evaluate String
        JSR     L037E           ; Get 2nd String parameters to Stack
        jsr     L034E           ; EVALUATE Byte/Word Parameters
        jsr     L03A4           ; Process Parameter: Drive
        beq     L0599           ; Jump, if NO ERROR

        pla                     ; Clear 2nd String discriptor Stack
        pla
        pla
        jmp     L04D1           ; Goto clear 1st String discriptor Stack and return
;
L0599:                          ; Continue on NO ERROR
        pla
        sta     $9E             ; Get 2nd String
        pla
        sta     $9F
        pla
        sta     $BF             ; Lenghth to $BF
        bne     L05AE

                                ; 2nd string was empty , delete file now
        lda     #FINAME>>8      ; Empty file name vector
        sta     $9F
        lda     #FINAME&255     ; *** POINTER TO "00" NAME (EMPTY)? ***
        sta     $9E
        lda     #$01            ; Lenth of 1 is enough
L05AE:
        sta     $C0
        pla                     ; Get 1st String
        tay
        pla
        sta     $A3
        pla
        jsr     L07FA           ; FAT name Search (1 sector)
        beq     L05BE
        jmp     L0504           ; Return wit ERROR 9 File not found
L05BE:
        lda     $BF             ; if BF length = 0, delete file
        bne     L05C2X          ; Jump, if not delete

        jsr     L05D2           ; DELETE CURRENT SELECTED FILE
L05C2X:
        jsr     L05C2           ; COPY NAME and TYPE to FAT
        jmp     L03CD           ; Write FAT (6) after file save and RETURN


L05C2:                          ; **** COPY NAME and TYPE to FAT *******
        lda     $C0             ; Transfer String vector to A2/A2 and 9F
        sta     $94
        lda     $9E
        sta     $A2
        lda     $9F
        sta     $A3
        jmp     L048B           ; Copy FILE NAME to FAT and RETURN

;
L05D2:                          ; ********* DELETE CURRENT SELECTED FILE ****
        lda     #$FF
        sta     $E021           ; Read or Delete flag set to DELETE (FF)
        jsr     L0618           ; ROM: DOS READ OR DELETE (5)
        ldy     #$00
        sty     $E021           ; Read or Delete flag set to READ (00)
        jmp     L0581           ; Check for ERROR, Read FAT back

;                               ; **************** PTR ******************

DEZ:    jsr     $ABFE           ; BASIC: Check for "("
        beq     L0615           ; FC ERROR

        jsr     $AAC1           ; BASIC: Evaluate
        bit     $5F
        bpl     DEZ_1           ; Jump if Number
        jsr     $B2B3           ; ROM BASIC, Release string
        beq     L0615           ; FC ERROR, String is empty

DEZ_1:
        jsr     $ABFB           ; BASIC: Check for ")"

        ldy     $71
        lda     $72
        jmp     BASIC_16_FLOAT  ; Convert Fixed Point to Floating Point
;
```

```
L0615:
        JMP     $AE88           ; BASIC: FC ERROR
;
SEL:    beq     L0615           ; **************** SEL ***************
        ldy     #$00            ; SEL will always reload FAT (diskchange)
        jsr     L0351           ; Evaluate single Integer parameter
        lda     $E020
        cmp     PAR_STOR        ; First Paramneter is DRIVE number the same ?
        beq     L03C7           ; Reload in case of diskchanges
        jsr     L03A4           ; Process Parameter: Drive
        bne     L03C7           ; on ERROR Second try to reload FAT and exit
        rts


L03E0:
        jmp     ($E000)         ; DOS: SEARCH FILE(0)
L03CA:
        jmp     ($E004)         ; DOS: WRITE FILE(2)
FMT0:
        jmp     ($E006)         ; DOS: FORMAT DISK(6)
L0618:
        jmp     ($E00A)         ; DOS: WRITE OR DELETE FILE (5)
L03CD:
        jmp     ($E00C)         ; DOS: WRITE FAT(6)
L03C7:
        jmp     ($E00E)         ; DOS: LOAD DISK FAT (7)

INVEC:
        jmp     ($0218)         ; INFO: indirect jump INVEC
OUTVEC:
        jmp     ($021A)         ; INFO: indirect jump OUTVEC



;
SCR:    beq     L0659           ; **************** SCR for 32x32 and 64x15/31 ***************
        jsr     $B3AE
        txa
        pha
        jsr     $B3AB
        lda     HORZ_SIZE       ; Create flag for 32 or 64 in $BF
        cmp     #33
        ror     $BF             ; Bit 7 is set for 64 mode
        pla
        bit     $BF             ; Check horizontal chars
        bmi     SCR64
        and     #$1F            ; mask off all above 32
        bpl     SCR32           ; Branch always
SCR64:
        and     #$3F            ; mask off all above 64
SCR32:
        sta     $11
        txa                     ; Y-value
        ldx     #$00
        clc
        eor     #$FF
        bit     $BF
```

```
          bmi     SCR642
          ror
SCR642:
          ror
          ror
          ror
          tay
          rol
          cpx     VERT_SIZE        ; Check for 64x32
          bne     SCR32x32         ; jump on 2k video RAM
          and     #$03
SCR32x32:
          and     #$07
          ora     #$D0
          sta     $12
          tya
          and     #$E0
SCR643:
          ora     $11
          sta     $11
          jsr     $AC01
          bne     L0665
L0659:
          rts
;
L065A:
          jsr     L0675
          jsr     $00C2
          beq     L0659
          jsr     $AC01
L0665:
          jsr     $AAC1
          bit     $5F
          bmi     L065A
          jsr     $B96E
          jsr     $B0AE
          clc
          BCC     L065A

L0675:                                 ; ***** Sub Get Parameter
          jsr     $B2B6                ; as chars to ($11)
          ldy     #$00
          tax
          bne     L0682
          inx
          lda     #$20
          bne     L0688
L0682:
          inx
L0683:
          dex
          beq     L068D
          lda     ($71),y
L0688:
          sta     ($11),y
          iny
          bne     L0683
```

```
L068D:
        tya
        clc
        adc     $11
        sta     $11
        bcc     L0697
        inc     $12
L0697:
        rts

CLG:    beq     L06A1                   ; ************* CLG *************
                                        ; CLG  : same as CLG 0
                                        ; CLG 3: Clear TOP with "00"
                                        ; CLG 2: Clear BOT with "20"
                                        ; CLG 1: ENABLE LOW RES MODE
                                        ; CLG 0: DISABLE LOW RES MODE
L069D:
        cmp     #$30
        bne     CLG0
L06A1:
        lda     #$11
CLGX:
        sta     ACIA_S                  ; Turn off RTS at standard Baudrate /16,8N2
        bne     CLGRET                  ; and return
CLG0:
        cmp     #$31
        bne     CLG1
        lda     #$51                    ; Turn on RTS at standard Baudrate /16,8N2
        bne     CLGX


CLG1:
        ldy     #$00
        ldx     #$00
        sty     $9E
CLGC2:
        cmp     #$32
        bne     CLGC3                   ; Jump on CLG 3 , clear with 00 starting at D0
        ldx     #$20                    ; clear with 20
        lda     #$D2                    ; clear from D2
        cpy     VERT_SIZE               ; 0=1kB >0=2kB
        beq     CLGC4
        lda     #$D4                    ; clear from D4
        bne     CLGC4
CLGC3:
        lda     #$D0
CLGC4:
        sta     $9F
        txa
        ldx     #$02                    ; 2 blocks
        cpy     VERT_SIZE               ; 0=1kB >0=2kB
        beq     CLGCL
        ldx     #$04                    ; 4 blocks
CLGCL:                                  ; Clear screen section with A, D0 or D2/D4
        sta     ($9E),y
        iny
        bne     CLGCL
```

```
          inc     $9F
          dex
          bne     CLGCL
CLGRET:
          jmp     $00BC                    ; Return to BASIC



GDIS:     jsr     $B3AE                    ; *********** GDIS x 0..127 , y 0...31/63 ***************
          ldy     #$FF
L06BB:
          iny
          sty     $C0
          beq     L06C3
          jsr     $B3AB                    ; Arg from Basic line
L06C3:
          ldy     $C0
          stx     $F0,y                    ; F0=X F1=Y F2=DOT(1) or LINE(2)
          cmp     #$2C
          BEQ     L06BB                    ; Store parameters in F0....

          ldy     $F1                      ; Y
          lda     $F2                      ; Type
          and     #$03                     ; Mask Type     0   1   2   3

          cmp     #$02
          bcc     L06_P
          eor     #$83                     ; 00 01 81 80 (clear dot, dot, line, clear line)
L06_P:
          ldx     $F0
          stx     $30
          sty     $31
          sta     $34

          lda     HORZ_SIZE                ; Create flag for 32 or 64 in $BF
          cmp     #33
          ror     $BF                      ; Bit 7 is set for 64 mode
          lda     $34
          bpl     L06DC                    ; Bit 8 of Mask set? (Line Mode)
          jmp     L072E
L06DC:

          lda     $31
          sta     $37
          lda     $30
          sta     $36

; ***************************************************************************
;
L06E9:                                     ; *** Plot pixel (30,31) x,y preserverd
          stx     $3E
          lda     $30
          cmp     #$80
          bcs     L072B                    ; Jump and quit if X >= 128
          sta     $32
```

```
        bit     $BF
        bmi     PP64                    ; jump to 64 cgar section
        asl     $32                     ; Only for 32 char mode
        and     #$03
        tax

        lda     $31
        clc
        adc     #$C0
        bcs     L072B                   ; Jump and quit if Y >= 32 or Y>=64
        eor     #$FF

        lsr                             ; 32 char section
        php
        lsr
        ror     $32
        lsr
        ror     $32
        lsr
        ror     $32
        ora     #$D0
        sta     $33
        txa
        plp
        bcc     L0716
        adc     #$03
L0716:
        tax

PPexit:

        lda     $34                     ; Check delete or write
        lsr
        lda     GDPIX,x
        ldx     #$00
        bcc     L0725
        ora     ($32,x)
        bne     L0729
L0725:
        eor     #$FF
        and     ($32,x)
L0729:
        sta     ($32,x)
L072B:
        ldx     $3E
        rts

PP64:                                   ; *********** 64 char section
        lda     $31
        clc
        adc     #$C0
        bcs     L072B                   ; Jump and quit if Y >= 64
        eor     #$FF
        sta     $33                     ; Save converted Y
        and     #$03
        asl                             ; Y0,1 for mask offset
        tax
        lda     $33
        asl     $32
        lsr
        lsr                             ; move Y2 down
        lsr
        ror     $32
        lsr
        ror     $32                     ; now x0 is in carry , 32 ready

        bcc     L06E9x
        inx                             ; Mask X0 set, X ready
L06E9x:
        lda     $33                     ; Correction for double size display
        lsr
        lsr
        lsr
        lsr
        clc
        adc     #$D0


L06E9z:
        sta     $33
        jmp     PPexit

;****************************************************************************
;
L072E:                          ;*** LINE MODE
        ldx     #$02
L0730:
        lda     $2F,x
        sec
        sbc     $35,x
        bcs     L073D
        eor     #$FF
        adc     #$01
        clc
L073D:
        sta     $37,x
        ror     $35
        dex
        bne     L0730
        lda     $38
        bne     L074C
        cmp     $39
        beq     L0757
L074C:
        asl
        bcs     L0756
        asl     $39
        bcc     L074C
        ror     $39
        clc
L0756:
        ror
L0757:
        sta     $38
        ldy     #$00
        ldx     #$00
L075D:
        txa
        clc
        adc     $38
        tax
        bcc     L076E
        bit     $35             ; Check Bit 7
        bmi     L076C
        inc     $30             ; increment x coordinate
        bcs     L076E
L076C:
        dec     $30             ; decrement x coordinate
L076E:
        tya                     ; Delta Y in Y
        clc
        adc     $39
        tay
        bcc     L077F
        bit     $35             ; Check Bit 6
        bvs     L077D
        inc     $31             ; increment y coordinate
        bne     L077F
L077D:
        dec     $31             ; decrement y coordinate
L077F:
        JSR     L06E9           ; Plot pixel (30,31) x,y preserverd

        lda     $30             ; compare that end point is reached
        cmp     $36
        bne     L075D
        lda     $31
        cmp     $37
        bne     L075D
        rts
;
SAFI:   jsr     $AAAD           ; ************ SEQW  ***************
        jsr     $B408           ; BASIC GET 16BIT ARG FROM BASIC LINE
        jsr     $00C2           ; BASIC GET CURREMT CHAR FROM BASIC LINE
        beq     L07A2
        jsr     $AC01           ; BASIC CHECK SYBBOLS IN BASIC CODE
        bne     L07C8
L079F:
        jsr     L0675           ; Parameter Loop
L07A2:
        lda     #$2C
        ldy     #$00
        sta     ($11),y
        inc     $11
        bne     L07AE
        inc     $12
L07AE:
        jsr     $00C2           ; BASIC GET CURREMT CHAR FROM BASIC LINE
        bne     L07C5
        ldy     $11
        lda     $12
        ldx     #$00
        stx     $5F
        ldx     #$90
        sta     $AD
        sty     $AE
        sec
        jmp     $B7E8           ; BASIC Return new 16bit address
;
L07C5:
        jsr     $AC01           ; BAIC CHECK SYBBOLS IN BASIC CODE
L07C8:
        jsr     $AAC1           ; BASIC EVALUATE EXPRESSION
        bit     $5F
        bmi     L079F           ; Loop next parameter
        jsr     $B96E           ; BASIC Build ASCII number in 100 form AC-AF
        jsr     $B0AE           ; BASIC PRINT MESSAGE
        jmp     L079F           ; Loop next parameter

;       ****************************************************************
;                               **** SEQS Set Read poiter to memory ******

SEFI:   jsr     $AAAD           ; BASIC GET 16BIT ARG FROM BASIC LINE
        jsr     $B408           ; BASIC Convert FLOAT to INT, Result in 11-12
        lda     $11
        ldy     $12
        jmp     $A624           ; within BASIC FINALIZE RESTORE

;       ****************************************************************
;
EOF:    ldy     #$01            ; *************** EOF ***********
        lda     ($8F),y
        cmp     #$2C
        bne     L07EE           ; EOF = 1 (more data)
        dey                     ; EOF = 0 (end of file)
L07EE:
        jmp     $AFD0           ; BASIC Returns value in Y ??


;       ****************************************************************

;                               ; *********** DCHK Check/Verify *************

CHECK:  JSR     L037E           ; Get String parameters to Stack
        pla                     ;
        tay                     ; Low address
        pla                     ;
        sta     $A3             ; High address
        pla                     ; String length
        jsr     L07FA           ; FAT name Search
        beq     CHEC1           ; Check if found, jump if name exist in FAT
        sta     $E027           ; ERROR 9 File not found returned
        rts

CHEC1:
        sta     $E026
        jsr     L0618           ; DOS READ/VERIFY FILE(5)
        lda     #$FF
        sta     $E026           ; Verify bit set to 1
        rts
;


;                               ; *************** FAT name Search ********
                                ; (A) length, (Y) low
L07FA:
        bne     L07FF           ; String lenght >0
        jmp     $B268           ; BASIC: Sting length ERROR

L07FF:
        tax

L0800:                          ; MOVED FROM ROM TO DOSSUP
        sty     $A2             ; Store DOS FILNAME VECTOR ADDRESS
        stx     $94             ; String length in X and $94
        jsr     L03E0           ; Call DOS SEARCH FILE (0)
        lda     DOS_E022        ; Load DOS Vector for FILE POINTER
        sta     $F5
        lda     DOS_E023
        sta     $F6

        cmp     #$F8            ; Pointing outside FAT (Means Name not found)
        bne     L0822
        lda     #$09            ; Load ERR=9 (File not found)
        rts
L0822:
        lda     #$00            ; File found
L0824:
        rts
;
; ********************************************************************

                                ; **** VERSION ****
SVER:
        lda     #FATVER&255
        ldy     #FATVER>>8
        jmp     $A8C3           ; BASIC PRNT string

;
; ********************************************************************


FORMAT:                         ; **** FORMAT ****
        ldy     #$00
        sty     PAR_STOR+1      ; Second Paramneter is Full(0) or BootSec(>0) only
        sty     PAR_STOR+1      ; Third Parameter is Single (0) or Double(>0) sided
        JSR     L0351           ; Evaluare integer parameters
        lda     PAR_STOR        ; First Paramneter is DRIVE number
        cmp     #$04            ; Check for range 0…3
        bcc     FMT1            ; Jump, if <4
        jmp     $AE88           ; FC-Error
FMT1:
        sta     $E020           ; Store target drive number
        lda     PAR_STOR+1
        beq     FMT2
        lda     #$FF
FMT2:
        sta     $E026           ; Store 00/FF for format option
        lda     PAR_STOR+2
        beq     FMT3
        lda     #$FF
FMT3:
        sta     $E01E           ; Store 00/FF for single/double sided

        jsr     FMT0            ; DOS: FORMAT DISK (6)
        ldy     #$00
        sty     $E01E           ; Default Single (0)  sided
        dey
        sty     $E026           ; Back to read mode

        lda     $E027           ; Check Error
        bne     L0824           ; Jump on Error
        jsr     L03C7           ; DOS_READ_FAT(7), from new drive
VERSION:                    ; Updates VERSION Info
        ldy     #$0F
VERLP:
        lda     VER,y
        sta     FATVER,y
        dey
        bpl     VERLP
        inc     FATCHANGE       ; Mark changed FAT
        jmp     L03CD           ; DOS: WRITE DISK FAT (7)


        .db $00                 ; End of Table

        .DW SVER-1
        .db $D2, $45, $56                       ; VER Commad

        .DW FORMAT-1
        .db $D4, $4D, $46, $44                          ; DFMT Command

    .DW DIC-1
    .DB $D2, $49, $44                           ; DIR Command

    .DW CLG-1
    .DB $C7, $4C, $43                           ; CLG Command

    .dw SEL-1
    .DB $CC, $45, $53                           ; SEL Command

        .DW CHECK-1
    .DB $CB, $48, $43, $44                      ; DCHK Verify Command

    .DW CHANGE-1
    .DB $CE, $45, $52, $44                      ; DREN Command

    .dw STR-1
    .DB $D6, $41, $53, $44                      ; DSAV Command

    .dw EOF-1
    .DB $C6, $4F, $45                           ; EOF Command
    .DW DEZ-1
    .DB $D2, $54, $50                                  ; PTR Command

    .DW SAFI-1
    .DB $D7, $51, $45, $53                             ; SEQW Command

    .dw SEFI-1
    .DB $D3, $51, $45, $53                             ; SEQS Command

    .DW SCR-1
    .DB $D2, $43, $53                                  ; SCR Command

    .dw GDIS-1
    .DB $D3, $49, $44, $47                             ; GDIS Command


; 7 bytes free

TOKEN:
;
; ****************** END $EFFF ******************************************

HERE_POS        .SET *
                .ORG  $F000
DELTA           .SET HERE_POS - *
                .IF DELTA > 0
                .ERROR "*** A D D R E S S Conflict !! ***"
                .ENDIF
```