K E N - D O S

O P E R A T I N G M A N U A L

TABLE OF CONTENTS

INSTALLATION.

In order to have KEN-DOS working it is necessary to connect two points on the main PCB by means of the wire provided with the KEN-DOS system-package. Pin 49 of the X bus has to be connected with pin 8 of the DCE-bus, thus routing the hold-line to the DCE-bus. This hold-line enables handshaking. Without this modification it would be impossible for KEN-DOS to work in double-density mode.

SOME ADVICE WHEN MAKING THE CONNECTION.

Remove all cables, connectors etc. from the back-panel of your PC. Remove the enclosure of your DAI-PC. This is accomplished by removing the four black pins on the right and left side of the enclosure. Put both tumbs on the keyboard and fold your fingertips on both sides under the edge of the white DAI-PC enclosure. Gently pull the enclosure upwards and push to the rear. Be carefull not to damage the hooks on the DCE-bus connector. Take away the bottom of the enclosure by removing the (black) pins (on the latest versions of the DAI-PC these pins have been replaced by nylon nuts and bolts). Carefully lay the main-PCB with component-side down on a soft surface. Put something under the side where the video-board is located to prevent it from damage.

SOLDERING THE WIRE.

Closely examine figure-1. Take the wire provided with your KEN-DOS package and put some solder on both tips. Apply some fresh solder to pin 49 of the X-BUS and to pin 8 of the DCE-BUS (fig 1). Carefully solder one side of the wire to pin 49 of the X-BUS and the the other side of the wire to pin 8 of the DCE-BUS. Use a piece of tape to fix the wire against the PC-board. Be sure not to lead the wire over the base plate of the power transformer, but around it (fig 1). The connection is now made. Check thoroughly for any short-circuits or drops of tin you might have spilled etc.
To be able to RESET the computer and return to the command-mode the User must also solder a wire between pin-12 of the 8080A and pin-18 of the X-bus (fig-1). If the Epromcard is also modified (fig-5) then bank-0 will always be selected on RESET. With no modification the system will often 'hang ' on RESET.
When you are convinced that everything is in order, remount the main PC-board on the bottom of the enclosure. Now install the provided EPROM-board on the X-bus connector, with the EPROM-sockets facing the keyboard. Note that the EPROM-board is installed slanting backwards. DO NOT APPLY FORCE. Pay attention to the pins of the X-bus connector being exactly in line with the connector on the EPROM-board. Look at the back-side as well. Replace the top of the enclosure. After reconnecting the cables to the rear-panel, switch your computer on. If everything functions normally, you have probably done a fine job (you could run a few programs to make sure).

PROBLEMS!!!
When your PC does nothing at all or does not behave the way it should, then re-open the enclosure and check again for short-circuits. Check again if you have connected the wire to the right pins. If you cannot find anything that should not be the way it is, then disconnect the wires you have soldered. Also remove the EPROM-board. Switch your computer on again. If it still malfunctions then the problem lies elswhere. If your computer works fine then the problem might be in the EPROM-board and we advise you to contact us.

TESTING THE SYSTEM.
Switch-off your computer and connect the drive unit to the DCE-bus at the rear-panel of your computer. The connector on the drive-unit cable fits into the DCE-bus connector with the dot on cable turned upwards. First switch your Ken-Dos system on, hereafter your computer. The screen will display the message "KEN-DOS V3.x" or just the cursor is visible or you will get an error message. If one of these things happens the auto-start is working correctly,but if not then retry by switching your computer and Ken-Dos system off and on with FIVE SECONDS in between.Be sure that the 'DCR/DISK ' switch on the rear is put to 'DISK '
Switch-off your computer and drive unit. Put a blank floppy-disc in drive-0 (left unit). First switch-on your drive unit and thereafter your computer. On the screen will appear "KEN-DOS V3.x" followed by an error message 'Read error in FAM etc.'. This error occurs because the diskette inserted was not formatted. If a formatted diskette was inserted, a 'not found ' error will be printed indicating that an autostart file was not found.
Important:
If you have inserted a diskette that was formatted on another computer or with another floppy-system it is necessary that you use the 'INIT 'command before you start formating.
The KEN-DOS system is now ready for testing.

With the sytem still ON and a blank floppy in drive-0, you can now enter 'INIT0 ' and press the return key. The system will respond by displaying the message 'Tracks' on the screen. Now you enter either '40' or '80' depending on the type of drives that are installed and again press the return key. The system will display the current DATE which in this case will be 'xxxx8x '.You can now enter the correct date by pressing the space-bar and typing the current date (DDMMYY) or simply hit the return-key in which case no date will be entered. Then enter "FORMAT" and press the return-key. If everything goes well, you'll see a "0" appearing on the screen every second or so. If a "1" or even a "5" appears after quite a while then there is definitly something wrong and you should refer to the section "TROUBLE-SHOOTING" (use a new diskette to be sure). Depending on the type of drives installed, there will be four or eight rows of ten zero's when the "*" appears on the screen. This indicates that the disk is formatted and ready to be used. Enter "TEST0" and press "RETURN". If there is no error message then the system works well and KEN-DOS is all at your service.....

HARDWARE.

The KEN-DOS system hardware consists of two parts:
1)EPROM-board with system-software in EPROM
2)Drive unit with controller-board, powersupply, enclosure and drives (optional)

EPROM-BOARD.

On the board there is room for a maximum of 96Kbyte of EPROM divided over 6 EPROM's. The smallest EPROM is the 2716 (2Kb), the largest is the 27128(16Kb) EPROM. Should the latter be installed, then one jumper has to be cut, while another has to be connected. (refer to fig.2/3). The EPROM-board is normally configured to accept all types of EPROM's except 27128. The EPROM's on the board are placed in memory area #F000-#F7FF. KEN-DOS also uses a heap at addresses #F900-#FAFF. On power-on the bankswitch routines are written to this area of memory. We recommend not to use this memory-area. Should you do so, you run the risk that the system will "hang" which will certainly cause loss of all your data in memory. The memory-banks are switched at address #F900. To be able to switch to another memory-bank a zero has to be written to location #0296.If you fail to do so and try to switch banks the system will crash. You can avoid this problem by switching on the computer with the DCR/DISK SWITCH to DCR. The Ken-Dos system will not be initialised because the AUTOSTART is then disconnected. The DOS resides in EPROM 1,2 and 3. EPROM 3 is reserved for a CP/M bios which will be available soon. The other sockets are available to the user.It is therefore possible to put often-used software in EPROM's, which can then be addressed via the "BANK"command. In this way it is possible e.g.to load and run DNA within 1 second! Basic programs can also be put into EPROM's, but occupy relatively much memory. The EPROM sockets are numbered as follows: The rightmost socket is no.1 and the leftmost is no.6(fig.2). Bankno's increase by "8" 256 bytes.EPROM no.1 starts with 00(bank 1).The second bank is 00 + 8 = 08 etc.

EPROM	2	starts	with	01
"	3	"	"	02
"	4	"	"	04
"	5	"	"	03
"	6	"	"	05

If you want to read from bank 4 which is located in EPROM 1, you have to write a "0" to memory location #0296. Then write #18 to memory location #F900.(03+8+8+8=27 = #18). To return to KEN-DOS write a "0" to #F900 and "2" to #0296.
If you are using 2716-eproms, beware that they are selected with the base-bank number+8. An eprompprogrammer of the 'ZERO 'type could also be connected to the eprom-card and should be selected as if it were an 2716-eprom.

On EPROM-boards rev.2.x it is possible to use socket 6 for a read/write device. Situated between sockets 5 and 6 are two small points, closely together, to one of which the R/W signal is routed. By connecting these to points and cutting a trace at the back-side of the board (see fig.1,2 W and X), you can place a 6116 static RAM in socket-6.
It is then possible to read and write to address #F000-#F7FF. In order to enable a write operation you will have to issue a "DI" asembler-command first.(disable interrupt).If you fail to do so, a stack-overflow (BREAK with KENDOS) will occur.

Provisions have been made on the EPROM-board to enable users to install a keyboard-beep. This will give acoustical feed-back when a keystroke has been accepted by the computer. An application-note fully describing this feature will follow soon.

THE DRIVE-UNIT.
The drive-unit houses the controller-board. All controll-signals between computer and disk-drives and all data-transfer is handled by this VLSI device. The controller used by the KEN-DOS system is capable of transfering data between floppy-disk and main memory at a rate of 250 Kbits per second, using MFM technique. Data is checked on CRC-errors. The controller can handle a maximum of 4 double-sided drives. The KEN-DOS operating system looks at double-sided drives as two single sided drives. Four double-sided drives are therefore handled like 8 single-sided drives.

Drives are numbered as follows:
Drive (0/1),(2/3),(4/5,(6/7)
If you have installed two single-sided drives you can read from them or write to them as drive 0 and drive 2.
Do you have two double-sided drives you can refer to them
as logical drive 0 and 1 physical the first drive
logical drive 2 and 3 physical the second drive.
The system comes complete with connectors for two drives. Should four drives be connected to the system, a "T" connector has to be pressed on to the cable. The controller can handle both 40- and 80track drives. The software supports both formats. Although it will probably not cause any damage,we do not recommend to connect both kinds of drives to one controller at the same time. With 80 track drives it is possible (using a utility program) to read 40 track diskettes.The other way around is not possible. The directory of either format can be red on both systems. Provisions have been made on the controller-board to support connection of 8" drives.These drives can only be used in single-density mode. For more information on this subject, refer to the 8" application note.

THE FLOPPY DISC.

The floppy-disc used, are of the soft-sectored type and are formated at 5 sectors per track. This makes it possible to store 400 Kbyte of data per disc-side using 80 track drives and 200 Kbyte per side using 40 track drives. In a system with two 80track double-sided drives you can store 1600 Kbyte (1.6 Mbyte)of data. To read 400 Kbyte of data in a sequential cyclus, KEN-DOS only needs 32 seconds. This is fast enough for animated graphics or wordprocessing using overlay's (paging). The first 3 tracks on a disk are reserved. Track zero for the directory and track-1 and track-2 for subdirectories. Sector 5 of track 2 is used by the "TEST" command. This means that the user has access to 400 minus 15 Kbytes of storage.

The directory on double-density disks allows 128 entries. By using sub-directories this can be increased. Maximum file-length is 250Kbyte. A file can be overwritten even if the new file is larger than the old one. For sequential files KEN-DOS uses "dynamic file allocation". Random files have to be created before-hand and are of a specific length. It is, however, possible to make a Random-access file larger than initially created. This influences acces-time, although this will be hardly noticeable. To deal with all mentioned above, KEN-DOS uses a "file allocation map (FAM). This map is located on track zero and occupies 512 bytes. The main directory occupies 4 blocks of 1Kbyte each.

POWERSUPPLY.

The powersupply is dimensioned to provide adequate power for at least 2 disk-drives.When the user wants to install 4 slim-line drives it will be necesarry to mount an extra powersupply. Stack-overflow caused by spikes on the mains-supply, can occur. We advise to apply a mains-filter. Without mains-filter you run the risk of loosing data if an stack-overflow occurs during writing to the diskette. The stack-overflow interrupt is redirected to the break-routine.
The DAI is known as a computer that is not properly cooled. We therefore advise to install a cooling-fan if possible. Even with the cooling modifications on the latest versions the fan will not be a waste of money. Some fans are not suitable for the job, because they produce spikes on the power-supply.
Warning: If you have any doubt in modifying your computer, DON'T DO IT. We do not accept responsibility if your computer is damage due to modifications made in connection with KEN-DOS. We have made all the modifications beforehand and there where no problems what so ever. Ask a qualified person to assist you if you have doubt in yourself.

THE DRIVES.

All SHUGART-compatible drives can be used, provided track to track steptime is 6 ms.or less. With longer steptimes the drives can also be used (decreasing system performance), but a modification in the operating system has to be made.

GENERAL INFORMATION

COMMANDS:

It is possible to issue more than one command on the same line. This is done by typing a ': ' behind a command.

Example:

```
DISK:RENAME"TEST,TEST2":DIR
```

If in the commandtable (see COMMAND TABLE) a '* ' is placed behind a command in the table, this means that all commands issued on the same line behind this command will not be executed.

Example:

```
DIR:LOAD"TEST":CAS
```

Only 'DIR ' will be executed.

A KEN-DOS command can not be issued after a BASIC-command on the same line. A syntax error will follow this attempt.

Example:

```
LOAD"TEST":DIR
```

A syntax error will result.

Under program control a command is issued by first executing a 'CALLM #F000 ' followed by 'REM '.

Example:

```
10 CALLM#F000:REM DLOAD"TEST" or
10 CALLM#F000:REM DIR
```

It is also possible to use the print command.

Example:

```
5  COMMAND$="DLOAD":X=2:NAME$="TEST"
10 PRINT CHR$(12):REM clear screen.
20 PRINT COMMAND$:X:CHR$(#22): NAME$:CHR$(#22): :
                                          CALLM #F003
30 END
```

On the screen will be printed: DLOAD2"TEST". The 'CALLM#F003 ' will see that the command is executed.

A special entry-point is provided for the ML-programmer to have acces to all KEN-DOS commands.
A call to #F006 with in the A-reg the desired command number will select the command.
If additional information is required by the command then this information must reside in a buffer starting at #F9A0 .
The first byte of this buffer tells the system howmany parameters are supplied by the calling program. If a name is expected then the adress of the name-string must reside in the next two adresses in the buffer. For the next parameters on the next adresses; etc..
The last byte in the buffer must be a ' 0 '.

The commands with number are:

VERIFY 1	COMPAC 11	RCAS 21	CODE 31
UNLOCK 2	BACKUP 12	OPEN 22	BANK 32
RESTOR 3	VOICE 13	NAME 23	PUT 33
RENAME 4	DSAVE 14	LOCK 24	KEY 34
PROTEC 5	DLOAD 15	KILL 25	GET 35
MANUAL 6	CLOSE 16	HELP 26	DIR 36
LPRINT 7	CLEAR 17	FIND 27	COMP 37
FORMAT 8	WCAS 18	DISK 28	CAS 38
DELETE 9	TIME 19	DATE 29	BUF 39
CREATE 10	SWAP 20	COPY 30	JMP 40

FILE NAMES:

File names must be entered between double quotes. The max. length of a file name may not exceed 14 characters. The only non-alphabetic character that can be used as part of the file name is the '. '. All other characters will be removed by KEN-DOS. If the entered file name is longer than 14 characters no error message will follow, but the valid name will assumed to be the first 14 characters only.

Example:

```
DLOAD"ABCDEFGHIJKLMDOPQRSTUVW"
```

The valid file name will be : "ABCDEFGHIJKLMN"
A file name can be used in a short-hand notation by placing a '/'behind the name. KEN-DOS only looks for the

Example:

```
DLOAD"ABCDEFG/HIJKLMN"
```

KEN-DOS will search for the file with the name "ABCDEFG". This means that the first file encountered with "ABCDEFG" will be loaded. It is advisable not to use this notation when writing to a file. A mistake could be disastrous.

Example:

```
DSAVE"TEST/"
```

KEN-DOS will save a basic program on disk with the name "TEST". If a basic file with the name "TEST" or "TEST1" or "TESTPROGRAM" etc. already exist and this file is not closed, KEN-DOS will write to the first encountered file which starts with "TEST". If a number is placed before a file name, KEN-DOS assumes that the number is ment to be a drive-number and will therefore select the drive according to the number.

DRIVE SELECT:

To select a drive, the drive-number must be placed behind the current command or before the file name. If no number is entered then drive-0 will be selected. An exception is made for utility and basic commands. In that case the last selected drive is the default drive. Thus if first LOAD"2TEST" is issued and after that LOAD"PROGRAM", KEN-DOS will continue to select drive-2 because no drive-number was issued with the file name.

Example:

```
DLOAD2"0TEST"
```

The selected drive will be drive-0 and not drive-2 because there is a '0 ' before "TEST". If drive-2 is the desired drive, just issue DLOAD2"TEST" or DLOAD "2TEST". For basic and utility commands only the second form of drive-select is valid.

FILE TYPE:

KEN-DOS knows file types. The DAI protocol is followed. This means that for BASIC (BAS) a '0 ' is used, for UTILITY (UTY) a '1 ', for ARRAY (ARY) a '2 ', for SOURCE (SRC) a ' 3 ', for RANDOM (RND) a ' 4 ', for TEXT (TXT) a '5 ' and for DATA BASE (DBS) a '6 '. If an attempt is made to load or save a file with a differend file type, a 'type mismatch' error will occur. If an unknown file-type is used, for example '# ' or "% ' or '$ ' KEN-DOS will convert this file type.

AUTO EXECUTION FILE:

There is an auto-start option for files.
On power-on or on reset Kendos will search for an auto file in the directory of drive-0.
A file with AUTOEXEC..... in the name will be loaded automatically and executed if possible.To make an UTILITY file executable it must have a execution address. This address can be added to the file with RENAME.

Example:

REN"SPL+.#8500" or RENAME"SPL+.#8500"

Hereafter SPL can be loaded and eventualy executed if desired. With the auto-start option it is for example possible to set basic pointers or to set the Kendos buffer before loading SPL. Take in consideration that it is better to load a basic file with LOAD then with DLOAD when running a basic program.

COMMAND FILE

Kendos has an option which allows any BAS or UTY file to be loaded and eventually executed without using the command LOAD or DLOAD. The user can merely type the file name or a part of the name followed by the '/ ' and press hereafter the cursor-down key.After RETURN the file will be loaded and if possible executed. Of course the name must not be an command name.It is for example not possible to load a file with the name 'LOAD,DIR,CAS etc ' in this manner.

DCR SWITCH

Within a program it is possible to switch from KEN-DOS to DCR.
To switch from KEN-DOS to DCR first hex 0296 must be put to zero, then hex FA0E to hex 1, then put hex 0A at hex FA00, call hex F00C and finally restore hex 0296 to hex 2.
To switch from DCR to KEN-DOS first zero hex 0296, then zero hex FA00, zero FA0E, call F00C and finally set 296 to 2
Behind the contoler cabinet there is a switch which has to be set to DCR when operating with a DCR. This switch disables the autostart of the disk-system. The system is then auto-booted by the auto-start of the DCR.

MEMORY MAP

addresses in hex.

0000	Start DAI system-heap
02EC	Start free ram
AD50	Start KEN-DOS FAM(file allocation table) Mode-0 Moves with screen-mode. Can be relocated with BUF command to fixed address.
AF50	Start directory buffer (mode-0) Moves with screen mode. Relocatable with BUF.
B350	Bottom of screen ram
C000	Start adrs BASIC ROMS
F000	Start KEN-DOS banks with DOS
F800	Top of DAI-stack
F900	Start KEN-DOS heap
FA50	Start KEN-DOS bankswitch routine
FB00	MATH CHIP
FC00	TIMER
FD00	I/O
FE00	PPI
FFF0	INTERRUPT controler
FFFF	END

IMPORTANT ADDRESSES

0296	INPUT SWITCHING. Is set by KEN-DOS to hex 2. If KEN-DOS bank-switching is desired, then this adrs must be cleared(0 hex). Clearing also disables KEN-DOS COMMANDS.
0297 0298	Pointer to KEN-DOS COMMAND TABLE. Do not change this pointer
02C5 02D9	Cassette & KEN-DOS switching vectors. If the data on this addresses is changed, there will be reading and writing conflicts.
02E0 02E2	Vector to KEN-DOS bank-0. Do not changes this vector
02E3 02E5	DINC vector. jump to DDB4. Do not change.
0D00	Start COPY, COMPAC & BACKUP buffer. When using those commands the contents of the memory will be destroyed from this address till bottom of screen.
8000	Start of FORMAT buffer. Data will be destoyed from this address on till bottom of screen.
AD50	Start of FAM(file allocation table) Relocatable with BUF command.
AF50	Start directory buffer. Relocatable with BUF comm.
F000	Vector to initializing-routine which enables KEN-DOS commands to be used in BASIC programs. (CALLM#F000:REM ....command)
F003	Vector to routine to execute command printed on screen. Used in BASIC programs(see GENERAL INFO)
F006	Vector to routine to execute command called from ML-programs(see GENERAL INFO)
F00C	Vector to routine to select KEN-DOS. Used to switch between DCR & KEN-DOS within a program(see GENERAL INFO)
F01C	Pointer to KEN-DOS command table
F2F2	Cold start
F800	Used by KEN-DOS command FIND to indicate if a string is found or not(see FIND)
F900	Bankswitch address.(see bank numbers) Also #FA00
F901 FAFF	Kendos heap. Do not modify the contents of the heap addresses. The data on the diskette could be ruined is the data on these addresses are modified without knowledge of there purpose.

F98C	
F98D	Drive motor-on time select. Normally set to hex 10
F98E	
F98F	Count-down buffer for motor-on time. Decremented by the interrupt-7 routine. Pointer of this routine is changed during motor-on status.(hex 70/71).
FA76	Vector to basic monitor. Can be modified for error trapping within a ML-program. Must be restored when trapping is finished.
FA7C	Vector to break routine.Restore after modification
FA82	Vector to Syntax error routine. Restore after modification.
FA96	
FAD3	Bankswitch routine. Do not modify this routine.
FAE5	Drive select buffer.
FAE6	Always zero
FAE7	
FAE8	Command table extension. Normally cleared.
FAEC	Funtion key buffer. If this buffer is zero keys act normal. Set to hex 1 on power-up.
FAEE	Drive select buffer.
FAEF	Track buffer. Set to hex 50 on power-up. Can be modified with INIT command.
FAF9	Buffer for paralel printer. Modifies PPI(8255)
FAFA	Bit select port-c PPI
FAFB	Bit select port-c PPI
FAFE	
FAFF	Pointer to FIXED address of KEN-DOS FAM & DIR buffer. For moving with screen ram these addresses must be zero.
0296	
FAD4	If you modify address #296 then you must also modify address #FAD4. If you are using a serial-input then normally #296 must be put to #1. If you do not put a #1 also at #FAD4 then after accessing the DOSsystem the #1 at #296 will be put back to #2 and you will not have your serial-input working.

COMMAND TABLE

BAS**	BACKUP
BANK	BUF
CAS	CLOSE
CLR	COMP*
CODE*	COMPAC*
COPY	CPM**
CREATE	DATE*
DCR**	DELETE
DIR*	DISK
DLOAD	DNA**
DSAVE	FIND(*)
FORMAT*	FWP**
GET	HELP**
INIT*	JMP
KEY(*)	
KILL	LIB**
LINK	LOAD
LOADA	LOCK
LPRINT**	NAME
MANUAL	OPEN
PRT	PUT
R	RCAS
RENAME	RESTOR
SAVE	SAVEA
SPL**	SWAP
TEST	TIME**
UNLOCK	VERIFY*
VOICE**	W
WCAS	

A ' * ' means that commands on the same line after the command with the '* ' will not be executed. The '(*) ' means sometimes i.e. if there is a message connected to the command
A '** ' means that no return is possible to the caller.

COMMAND SUMMARY.

Brackets '()' have no significance.
(Dr) means DRIVENUMBER unless stated otherwise.
(Nm) means NAME
(St) means STARTADDRESS
(En) means ENDADDRESS
(adrs) means address

BAS Syntax: BAS or BAS"*"
Enter address in hex. or dec.
*= restore basic pointers.
Useful after 'RESET'.
To be able to restore the pointers the 'BAS ' command must be issue beforehand. Beware that restoring the pointers will not always enables you to save your basic file. This is most likely to happen when basic starts at address hex 02EC. On reset the free ram is initialised by the DAI-system and a few addresses are modified. You can then only list your resident basic file, but not run it.
Purpose: Is used to relocate a Basic-program in memory. Can also be used to change or check Basic-pointers.

BUF Syntax: BUF or BUF"St"
Just BUF clears the previous setting and locates diskbuffer under screen-memory (normal situation). BUF"St" fixes the buffer
Purpose: Used to clear or fix the setting of the 1.5kbyte buffer used by KEN-DOS.
Before loading SPL, fix buffer to #0300.
On reset the setting is cleared.
Only #-addresses.

CAS Syntax: CAS
Purpose: Assigns system to cassette read/write.

CLR Syntax: CLR(dr)
Purpose: Changes disc protect-status.

CMP Syntax: CPM
Purpose: Assigns system to CP/M

DCR Syntax: DCR
Purpose: Assigns system to DCR

DIR

Syntax: DIR(dr) or DIR(dr)"I" or DIR(dr)"type"
(type=BAS,UTY,ARY,etc).
A (!) entered with the type will print all files of the desired type of the resident directory-block. Use space-bar for next block

Space-bar	:scroll files
Cursor-L	:load file
Cursor-R	:load and run file
Cursor-Up	:go back
Cursor-Dwn	:next blok
Shift-(d)	:delete file
RETURN	:end command.

In the command mode the DIR0 command can be executed with the cursor-left key.

Purpose: Display directory on screen.

DNA

Syntax: DNA
Purpose: Jump to #1100 where the assembler starts.
(DNA-file must be in memory)

FWP

Syntax: FWP
Purpose: Jump to #0400 where text-editor starts.
(FWP-file must be in memory)

GET

Syntax: GET(dr)"(Nm+),n,St" or GET(dr)"(Nm*),n"
n=recordnumber from 1-256.
The (+) is optional and is used to fix the address entered (St). Hereafter the second form of get can be used or not.

Purpose: Reads record (n) from file (Nm).
Puts data in buffer starting at St.
Buffer is always 1Kbyte long.

JMP

Syntax: JMP(n) or JMP(n)"(adrs)"
(n)=0...4
(n)=0 may be omitted.

Purpose: Jump to a specified address.
If JMP(n) is performed the system assumes that the jump-adrs is already in a buffer.
The jump-adrs can be set with 'JMP(n)"adrs"'in which case the jump is also performed. Only #-addresses. JMP is like CALLM but the adrs is preserved.

KEY

Syntax: KEY or KEY"*" or KEY"+" or KEY"(dr)(Nm)" or KEY"Nm,St,En"

KEY	function is executed
+=	Cursor-up function='HELP'
*=	Clear all functions.

(dr)(Nm):Cursor-up function='SAVE'
(dr=Drivenr,Nm=filename.)

Purpose: Assign Cursor-up key for 'HELP' or 'SAVE'.
Also for utility files; not in utility mode

```
LIB       Syntax:   LIB
          Purpose:  Display commandtable
                    LIB = display KEN-DOS commands

PRT       Syntax:   PRT(dr)
          Purpose:  Protect disk against 'FORMAT'

PUT       Syntax:   PUT(dr)"(Nm),n,St"
          Purpose:  Write  buffer starting at St to record
                    number n in file (Nm).(see GET)


SPL       Syntax:   SPL
          Purpose:  Jump to #8500 (SPL-program).
                    File must be in memory.
                    Use BUF-command before loading SPL.

BANK      Syntax:   BANK(n)
                    (n)=1.....32
          Purpose:  Get data from bank-n and put in memory.
                    (optional run.)
                    Data must start at #F010.
                    #F000=(St); in every bank
                    #F002=(Length); in every bank
                    #F004=execution-address or 0000
                         Only in the first bank.
                    #F006=:0=bas;1=uty;#FF=no data
                    Only in start-bank a '0' or '1'.
                    next bank always '#FF'.
                    #F007=next banknumber;0=end
                    #F008=:0=no execution;1=execution
                         Basic files:
                    #F00A=tolal length Basic-file
                    #F00C=total length textbuffer

                    #F010=start of data in bank

                    BANK(n)
                    n=1 startbanknr.=#04
                    n=2 startbanknr.=#03
                    n=3 startbanknr.=#05
                    n=4 startbanknr.=#02
                    n=5 startbanknr.=#0C
                    n=6 startbanknr.=#0B
                    n=7 startbanknr.=#0D
                    n=8 startbanknr.=#0A
                    For more details see 'Hardware'
```

CODE	Syntax:	CODE
	Purpose:	Entering of lockcode. On locking a disk or file the resident code is used.(max 5 decimal digits) If error message ' Locked ' is printed, the system expect that you enter the right code if you don't, you can not have acces to the file or disk.
COPY	Syntax:	COPY(dr1)"(Nm),(dr2)(Nm)" or COPY(dr)"(Nm),(dr)(*)" The (*) means that the first name will be used as the second name. If drive numbers are equal, KEN-DOS will ask for disk exchange. Just press RETURN if no swap is desired. Making a copy on the same disk can be useful for reorganisation of the disk. Syntax error will folow if '(dr2) ' is not issued.
	Purpose:	Copy files from drive (dr) to drive (dr)
DATE	Syntax:	DATE
	Purpose:	Entering date. Displays contents of date-buffer. Date can be entered after pressing space-bar (DDMMYY; DD=1...31, MM=1...12, YY=84 not 1984). On reset the date is preserved.
DISK	Syntax:	DISK
	Purpose:	Assigns system to diskette read/write.
FIND	Syntax:	FIND"($/#),(St),(En)" or FIND1"..... $=string to be searched for(max 14 char.). #=bytes to be searched for(max 2 bytes). Put $ before string data. Put # before bytes.

	Purpose:	Search for a string or bytes in memory.
		On address #F800 0 or 1 when using FIND1.
		0=not found;1=found
		$/# address on #F801/#F802.

HELP	Syntax:	HELP
	Purpose:	Display menu with syntax of some commands.
INIT	Syntax:	INIT(dr)
	Purpose:	Initializes drives. Drive heads are put to track-1. Init is followed by 'Tracks:' Entering number of tracks is advisable(80/40). Hereafter the date can be entered (see DATE).

KILL	Syntax:	KILL(dr)"(Nm)"
	Purpose:	Remove a file in directory. No recover is possible. To be able to 'KILL' a file, it has to have the 'deleted-status (§)'. Usefull to free disc-space.
LOCK	Syntax:	LOCK(dr) or LOCK(dr)"(Nm)"
	Purpose:	Adding a code to diskette or file. This code prevents read/write of diskette or file if not the right code was entered. Use CODE-command to enter code. The (*) is used as lock-symb in the directory
NAME	Syntax:	NAME(dr)
	Purpose:	Name a diskette (directory). Max. 31 characters.
OPEN	Syntax:	OPEN(dr)"(Nm)"
	Purpose:	Open a file for writing. This command removes the write protect status of the file. Symbol (O) in directory
RCAS	Syntax:	RCAS
	Purpose:	Enable cassette-read /disk-write.
SWAP	Syntax:	SWAP(adrs1,adrs2,Lngt) Only addresses with #-mark will be accepted.
	Purpose:	Swap data in buffers.
TEST	Syntax:	TEST(dr)
	Purpose:	Checks read/write of drive-(dr). Read/write and compair to sector-5 of track-2.
TIME	Syntax:	TIME
	Purpose:	Display time and put current date in date-buffer.(only if realtimeclock is installed)
WCAS	Syntax:	WCAS
	Purpose:	Enable cassette-write /disk-read.
CLOSE	Syntax:	CLOSE(dr)"(Nm)"
	Purpose:	Close file for writing. (C) is symbol
VOICE	Syntax:	VOICE"($)"
	Purpose:	Send data to speech-processor (if install.).
BACKUP	Syntax:	BACKUP(dr1)"(dr2)" (dr1) and (dr2) may be equal.
	Purpose:	Copy data from disc in dr1 to disc in (dr2). Diskette in (dr2) must not have bad-sectors. If dr2 is not empty then message 'not empty' Use RETURN to continue

COMPAC Syntax: COMPAC(dr1)"(dr2)"
(dr1) and (dr2) may be equal.
Purpose: Copy all files, except deleted files, from (dr1) to (dr2). For quick delete use DIR-command and restore with just REST afterwards.

CREATE Syntax: CREATE(dr)"(Nm),n"
n=number of records from 1.... 256.
Record length=1kbyte.
Purpose: Create (n) random-acces records in file (Nm).

DELETE Syntax: DELETE(dr)"(Nm)" or DEL(dr)"(Nm)" or use DIR-command for quick delete.
Purpose: Delete a file. Hereafter file can be 'killed'.

FORMAT Syntax: FORMAT(dr)
Purpose: Format a diskette in drive(dr)
All diskettes must be formatted before use.
If a format failes, don't read the directory of the 'BAD DISK ' or the track information will be disturbed. Better is to use INIT before FORMAT. In this way it is certain that the right track-information is in the track-buffer. The information about the tracks is derived from the disk.

LPRINT Syntax: LPRINT(n)"St,En"
n= to screen/RS232
n=1 to screen only
n=2 to DCE-bus
n=4 input from RS232
n=3 to RS232 only
Purpose: Send data from buffer to screen, RS232 DCE-bus or get data from RS232, print data on screen and store in buffer.
Baud-rate is normally 9600 with two stop-bits.
With 'POKE #FFF5,x' baud-rate can be changed.
x=0 110 baud,2 stop-bits, no parity,8 data-bits.
x=2 150,2
x=4 300,2
x=8 1200,2
x=16 2400,2
x=32 4800,2
x=64 9600,2
For 1 stop-bit add x+128

MANUAL	Syntax:	MANUAL(dr)"(R/W),(track),(sector),(adrs)" example: MANUAL2"R,#17,#2,#3000" Read sector-2 of track-23 and put it at address hex 3000. Use only hex numbers with the (#) sign in front.
	Purpose:	Manual read/write of sectors and tracks.
RENAME	Syntax:	RENAME(dr)"(Nm1),(Nm2)" or REN(dr)"(Nm1),(Nm2)" or REN(AME)(dr)"(Nm1)(+),(adrs)"
	Purpose:	Change file name. Put execution address in buffer. Only for UTY-files (Symbol (+) in directory)
RESTOR	Syntax:	REST(OR)(dr)"(Nm)" or REST(OR)(dr) If no name is entered then all files on the disk will be restored.
	Purpose:	Restore deleted files.
UNLOCK	Syntax:	UNLOCK(dr) or UNLOCK(dr)"(Nm)"
	Purpose:	Remove locked-status from disk or file.
VERIFY	Syntax:	VERIFY(dr) or VERIFY(dr)"(Nm)"
	Purpose:	Check data of diskette or file. No data compairing is performed.
LOAD	Syntax:	LOAD"(dr)(Nm)" (Nm)+(+)= load and run
	Purpose:	Load (Run) BASIC-file.
SAVE	Syntax:	SAVE"(dr)(Nm)"
	Purpose:	Write BASIC-file. (Nm)+(!)=close file after writing
DLOAD	Syntax:	DLOAD(dr)"(Nm),(St)" or DLOAD(dr)"(Nm)"
	Purpose:	Read BAS-file or UTY-file. (Nm)+(%)=Load a picture or don't check for 'out of memory'.
DSAVE	Syntax:	DSAVE(dr)"(Nm)" or DSAVE(dr)"(Nm),(St),(En)"
	Purpose:	Write a BAS/UTY-file to disk. BAS without (St) and (En) address (Nm)+(!)=close file after writing.
LOADA	Syntax:	LOADA"(dr)(Nm)"
	Purpose:	Read array.
SAVEA	Syntax:	SAVEA"(dr)(Nm)"
	Purpose:	Write array.

R

Syntax: R(offset) (dr)(Nm)
offset=optional
Purpose: Read UTY-file.
(Nm)+(%)=load picture or do not perform out of memory check.

W

Syntax: W(St) (En) (dr)(Nm)
Purpose: Write utility-file.
(Nm)+(!)=close file after writing.

ERROR MESSAGES

GENERAL:
If you fail to switch-on your KENDOS system before you turn your computer on, you will get beautiful colours on the screen and your computer will 'hang '. You can hereafter first turn on your KENDOS system and then reset your computer and every thing should then be normal. If there is no autoexecution file on the diskette an 'Not found ' error will be printed after reset or power-on.

File recovery.
If you try to write to track-0 you will get a message saying that this is not possible. If you want to save the FAM of the diskette extra on the disk, you can do so by using the MANUAL command to first read the FAM on track-0 sector-1 and hereafter write it to track-2 sector-1. If you might get an error saying 'Read error in FAM, files probably lost ' you can then read-back the FAM saved on track-2 and derived from it the locations of the diferent files on the diskette. The numbers in the FAM represent the file numbers in the directory. The first file always starts with number 21 and so on. The files are written from track-3 sector-1 till sector-5 of the last track.
If you for example have red the extra saved FAM and written it to a buffer starting at say #3000 and you want to find the track number and sector number of the file you intent to recover you must start counting from #3040 and stop counting at the first file number encountered. Divide your count by 5 to get the track number and the remainder is your sector number. In this manner you have found the first sector of your file. To find the second you have to continue counting etc.
If you have found all the correct track- and sector numbers in this manner, you can then use the MANUAL command to read them to the right memory location. If you don't know the start address of the file it will be quite difficult to restore your file. For BASIC files you have to know the length of the symbol table or the length of the text buffer to restore the file. This manner is more suitable to examine the contents of a file then to recover a file. As a rule it would be far more better to keep a save backup of all your diskettes then to run the risk of losing all your data or to perform the difficult operation of recovery.

Drive error: This report means that you have not inserted a diskette in the selected drive or that the door of the drive is not closed. If you select a drive that is not connected you will also get an error message. An error can also be printed if the power of the KENDOS system is turned off.
All other conditions can indicate a hardware failure of either the drives or of the KENDOS system and we recommend you to contact us.

Read error in FAM; files probably lost :
This is a very serious report. It could mean that all your files are lost on this diskette. The best thing to do is try to read the diskette several times before accepting the fact that your files are lost. Of course you will have a save backup of your diskette; if you don't you are in trouble.

Out of memory: This can indicate that there is no more room on the resident diskette or that you have failed to use the ' % ' mark on loading a picture or writing to the sreen-ram and above. This report is different from the normal basic report.

Read error: This indicates that a badsector was encountered and that reading is abandoned hereafter. Use verify to find and mark the badsector or try again. If you want to be sure that your writing was oke, perform a verify after writing. An error may not be detected on writing if this error is an disc-hardware error. This kind of errors can only be detected on reading the disc. If you have used VERIFY to mark the badsector it is necessary to KILL the file with the badsector.

Write error: This error will seldom occur. If it does then it might indicate that you have write-protected the diskette with a tap. In the worse case it means that there is something wrong with the system.

Type mismatch: This indicates that you have tried to read or write to file of a different type then the type that is required by the program.
If for example you want to read a source file with a basic file you will get this report.

Locked: This means that the diskette or the file is locked and that the wrong code is in the code buffer. If you want to get acces to the disk or file, you have to enter the right code using the CODE command. If you failed to do so you will not have acces to the diskette or file under no circumstances. This feature enables the user to prevent unauthorized persons to have acces to the diskette or to a specific file(s).

Protected: Indicates that you can not format the diskette.

Closed: Indicates that you have to open the file before you can write to it.

Not found: The file name that you have entered is not in the directory of the resident diskette. The file name must be entered in the same way that it is resident in the directory or you can use the short notation way with the /(see general information).

Seek error: This report will occur if the drive-head cannot be placed on the desired track. This can indicate a hardware problem or you have inserted a diskette that was formatted on a different type of drive i.e. different tracks.

UTILITIES

AUTOEXEC files
On power-on or on reset KENDOS will search for a special file on the diskette that is resident in drive-0. This special file must have the prefix 'AUTOEXEC ' and the first file that is encountered with this prefix will be loaded and if possible executed. This feature makes it possible to prepair the system for special purposes.
The next programs will give an example of how to use this feature:

Loading a basic file with a machine language part:

PROGRAM NAME: AUTOEXEC.BASML

```
10 REM basic must start at #D00
20 POKE #29B,0: POKE #29C,#0D : REM set start address basic
30 CALLM #DEB8: REM execute basic command NEW
40  REM  now load second basic program which  will  load  machine
C  language part.
50 CALLM#F000: REM DLOAD "MLBOOT+"
60 REM the + is for load and RUN.
70 END
```

PROGRAM NAME: MLBOOT

```
10 REM the mach.lang. program to be loaded is for example ' FGT '
20 CALLM#F000: REM DLOAD "FGT"
30 REM your FGT program is now loaded and basic starts at #D00
40 REM you can now load another basic program if you wish
50 REM loading another basic program ' BASICTEST '
60 LOAD "BASICTEST"
70 REM basic test will now be loaded and executed
70 END
```

Setting the KENDOS diskbuffer before loading ' SPL '

PROGRAM NAME: AUTOEXEC.SPLBF

```
10 CALLM#F000: REM BUF"#300"
15 REM  the diskbuffer is now set to #300 to #900 so SPL  source
C  must start at least from #900.
20 CALLM#F000: REM DLOAD "SPL+"
30 REM  SPL is now loaded and executed
40 end
```

Setting the date

PROGRAM NAME: AUTOEXEC.DATE

```
10 CALLM#F000: REM DATE
20 END
```

With the use of some simple basic programs you can prepair your system on powerup or reset.

PROGRAM NAME: KENDOS.DCR

```
10 REM switch to DCR
20 POKE #296,0
30 POKE #FA0E,1
40 POKE #FA00,#0A
50 CALLM #F00C
60 POKE #296,2
70 REM now DCR is selected
80 REM switch to KENDOS
90 POKE #296,0
100 POKE #FA0E,0
110 POKE #FA00,0
120 CALLM #F00C
130 POKE #296,2
140 REM disk is now selected
150 END
```

PROGRAM NAME: BASICVARKENDOS

```
10 INPUT A$:?
20 ?A$;:CALLM#F003
30 END
```

```
10 A$="DIR"
20 ?
30 ?A$;:CALLM#F003
40 END
```

In the DOS-versions 3.x- 3.7 there is a bug which do not give problems unless you stop a BASIC-program and want to use the command CONT. If you have accessed a DOS-command within a BASIC-program you must add a line number with POKE #118,#FF. This is to set the BASIC-runflag right again.

COMMAND EXPLANATIONS

Command: DIR
Syntax : DIR(dr) or DIR(dr)"x" or DIR(dr)"x!"
x=I print all informations of files (see Ex.D3)
x=BAS,ARY,UTY,SRC,RND,TXT,DBS
print selected type
x=type + ' ! '(see Ex.D4)
print all file names of the selected file-type wide
No waiting for spacebar in the selected directory
block. Use spacebar for the next directory block.

The directory of drive-0 can be printed by using the cursor-left key. Hereafter you can scroll the files over the screen by pressing the spacebar. For fast scrolling press the spacebar and the REPT-key simultaneously. With the cursor-up key you can go back to the first file name in the resident directory block. Pressing the cursor-down key will let you select the next block. If you press the cursor-left key the last file that was printed on the screen will be loaded in to memory(only BAS,UTY,TXT files) And if you press the cursor-right key the last printed file will be loaded and if possible executed(UTY-files must have a + sign). Deleting a file is possible by using the SHIFT- and the D-key. The last printed file will be deleted. This is very useful when you are planning to issue the COMPAC-command and don't want to copy certain files(see COMPAC).

Now closely examen Ex.D1.
On top of the screen the KENDOS version is printed. The next line shows the diskname if resident. On the third line the number of free sectors is printed which equals the free kilobytes on the diskette. To the right are the number of badsectors. These are sectors with errors which cannot be used. They are marked in the FAM with 'FF '. On the fourth line the number of tracks. This number is derived from the disk and placed in the track buffer. If you try to read a disk with a different format or a not properly formatted disk, you will supply the system with faulty track information. This will cause problems in case you should try to format a disk hereafter.If you have accidently done so you must issue the INIT-command prior to the FORMAT-command to avoid format problems.
Next to the track information the number of the selected drive is displayed. Line fifth displays the date on which the disk was formatted and on the right-side the status of the disk. This can be 'Protected ' or 'Locked ' or both (see Ex.D3). Protected means that you can neither format nor use the disk for a backup. First you have to clear the protected status with the CLR-command.
If the disk is locked the only way you can have acces to the disk is when the correct code is in the code-buffer. The correct code can be entered with the CODE-command. On the sixth and final line the information about the free entries is displayed. The total number of entries is 128. If no file was created on the resident disk, the 'E m p t y ' message will be displayed (Ex.D1).

```
K E N - D O S  V3.4
Diskname :
Free sec : 385/385KByte        Badsec: 000
Dens.mode: DD/80Tracks         Drvsel: Nr.1
Form.date: 070784              Status:
Fre entry: 128

E m p t y ?
```

1 **Ex.D1**

Now closely examen Ex.D2
Under the last line of the diskette header the first file name is printed, preceded first by the filecount number. Hereafter is the file-type followed by a * and an O. These are status symbol-signs for resp. ' locked ' and ' open '. Other symbols are § for a deleted file, C for a closed file, + for a UTY-file that can be executed, $ for a string array, % for a floatingpoint array and ! for an integer array.

```
K E N - D O S  V3.4
Diskname : EXAMPLES.of.DIRECTORY
Free sec : 348/348KByte        Badsec: 000
Dens.mode: DD/80Tracks         Drvsel: Nr.1
Form.date: 070784              Status: Protected!Locked!
Fre entry: 121

001 [TXT *O] TEXTtest
002 [UTY O@] UTYtest
003 [BAS C] BAStest
004 [RND O] RNDtest
005 [ARY $C] ARRAYtest
006 [SRC O] SOURCEtest
007 [UTY O+] UTYtest2
```

Ex.D2

Now closely examen Ex.D3
After the file name the first number represent the start address of the file, the second number is the file length, the third number is the total number of sectors that the file occupies on the disk. All these numbers are in hex notation. Following these numbers are two numbers. The first is the date on which the file was created and the second the date on which the file was last modified i.e. written to.

```
K E N - D O S  V3.4
Diskname : EXAMPLES.of.DIRECTORY
Free sec : 348/348KByte          Badsec: 000
Dens.mode: DD/80Tracks           Drvsel: Nr.1
Form.date: 070784                Status: Protected!Locked!
Fre entry: 121

001 [TXT *0] TEXTtest 3000 003C 0001 070784 070784
002 [UTY 0@] UTYtest 1000 1001 0005 070784 070784
003 [BAS C] BAStest A100 0002 0001 070784 070784
004 [RND 0] RNDtest 0000 000A 000A 070784 070784
005 [ARY $C] ARRAYtest 0C20 2628 000A 000080 000080
006 [SRC 0] SOURCEtest 71D0 1230 0005 000080 000080
007 [UTY 0+] UTYtest2 1000 1001 0005 070784 070784
```

Ex.D3

```
IMPLEMENT        RESTORE          DISPLAY          TRANSLATOR
BNKaa
```

Ex.D4

Physical construction of directory
The directory consist of 4-blocks which can contain 32 file name each. When the DIR-command is executed the first directory block that will be displayed is the physical sector-3 of track-0. The second is sector-5, the third is sector-2 and the fourth is sector-4 etc.
One block is 1024 bytes long. Each file claimes 32 bytes. The first byte of the 32-bytes is the file number which can also be found in the FAM (sector-1) if a file is created (see FAM). The second byte is the length byte for the file name and the next 14-bytes are preserved for the file name. Byte-17 is the status byte, byte-18 the type-byte, byte-19/20 start-adrs, byte-21/22 file-length, byte-23/24 execution address or length textbuffer in case of a basic file.

```
Command : BANK
Syntax  : BANK(n)
          n=1.....32
                    Data must start at #F010.
                    #F000=(St); in every bank
                    #F002=(Length); in every bank
                    #F004=execution-address or 0000
                         Only in the first bank.
                    #F006=:0=bas;1=uty;#FF=no data
                    Only in start-bank a '0' or '1'.
                    next bank always '#FF'.
                    #F007=next banknumber;0=end
                    #F008=:0=no execution;1=execution
                         Basic files:
                    #F00A=totall length Basic-file
                    #F00C=totall length textbuffer

                    #F010=start of data in bank

BANK(n)
        n=1  startbanknr.=#04 eprom socket-4 1e bank  (see fig.2)
        n=2  startbanknr.=#03 eprom socket-5 1e bank
        n=3  startbanknr.=#05 eprom socket-6 1e bank
        n=4  startbanknr.=#02 eprom socket-3 1e bank
        n=5  startbanknr.=#0C  -4 2e
        n=6  startbanknr.=#0B  -5 2e
        n=7  startbanknr.=#0D  -6 2e
        n=8  startbanknr.=#0A  -3 2e
        n=9  startbanknr.=#14  -4 3e
        n=10 startbanknr.=#13  -5 3e
        n=11 startbanknr.=#15  -6 3e
        n=12 startbanknr.=#12  -3 3e
        n=13 startbanknr.=#1C  -4 4e
        n=14 startbanknr.=#1B  -5 4e
        n=15 startbanknr.=#1D  -6 4e
        n=16 startbanknr.=#1A  -3 4e
        n=17 startbanknr.=#24  -4 5e
        n=18 startbanknr.=#23  -5 5e
        n=19 startbanknr.=#25  -6 5e
        n=20 startbanknr.=#22  -3 5e
        n=21 startbanknr.=#2C  -4 6e
        n=22 startbanknr.=#2B  -5 6e
        n=23 startbanknr.=#2D  -6 6e
        n=24 startbanknr.=#2A  -3 6e
        n=25 startbanknr.=#34  -4 7e
        n=26 startbanknr.=#33  -5 7e
        n=27 startbanknr.=#35  -6 7e
        n=28 startbanknr.=#32  -3 7e
        n=29 startbanknr.=#3C  -4 8e
        n=30 startbanknr.=#3B  -5 8e
        n=31 startbanknr.=#3D  -6 8e
        n=32 startbanknr.=#3A  -3 8e
```

The purpose of this command is to enable the user to derive data from the eprom-card. The data can be just bytes or bytes that form a program. In case of a program it is also possible to give control to the program after it is derived from the eprom-card. The data on the eprom-card is limitted to a memory address space of only 2 kbyte. However, due to the banked construction it is possible to store more than 2kilobytes of data. In fact it is possible to store 96kilobyte of data on the eprom-card and since the KENDOS system occupies 16kilobytes, the remaining 80 kilobytes can be used by the user to store data that can be quickly accessed. Of course the user must have an eprom-programmer to put the data in the eproms. If he hasn't he could ask a friendly owner or he could contact us. We will be happy to put his software in eprom agains minor costs.
If however he possesses a programmer the next instructions will enable him to program his own eproms according to the requirement of the BANK-command:
The next example will put a ML-program in eprom which starts at address hex 400 and ends at address hex 2130.

First you have to determine in which socket your eprom will be stored.

In this example we will use socket-4.

Next we have to calculate the total number of banks that we will need for our program.

* End address is #2130
* Start address is #400
* End adrs minus start adrs = #1D30 = 7472 bytes
  (use the print command to calculate)
  (example: PRINT HEX$(#2130-#400))
* In one bank there is room for #800-#10=#7F0= 2032 bytes
  (The first 16bytes in every bank are used by the BANK-command)
* Divide 7472 by 2032 = 3 remaining 1376=#560
  So we need 4 banks;3 fully occupied and one with 1376b

Now we must determine the data blocks and for the clearness we are going to prepaire a block of memory for this purpose. For example we will use #3000 till #5000
To do so we use the utility command fill.

* F3000 5000 0
  So we have zeroed all memory locations from #3000 till #5000.
* Determine the 4-block
  #400+#7F0=#BF0 first block from #400 -#BEF (#BF0-1)
  #BF0+#7F0=#13E0 second block from #BF0 -#13DF (#13E0-1)
  #13E0+#7F0=#1BD0 third block from #13E0-#1BCF (#1BD0-1)
  #1BD0+#560=#2130 fourth block #13D0-#2130
* Move the blocks to the cleared buffer with the move com.
  Skip the first 16bytes in the buffer for every move.
  M0400 0BEF 3010 1e block
  M0BF0 13DF 3810 2e block
  M13E0 1BCF 4010 3e block
  M13D0 2130 4810 4e block

Next we have to program the memory area from #3000 till #5000 into an 8kbyte eprom which will be located in socket-4.
But first we have to add the information for the BANK-command to the data in the buffer.
Lets see what banknumbers we will need for an 8kilobyte eprom in socket-4 and what start adrs and length.(see bank(n))

1e =#04 st.adrs=#0400 length=#7F0 next bankno=#0C
2e =#0C st.adrs=#0BF0 length=#7F0 next bankno=#14
3e =#14 st.adrs=#13E0 length=#7F0 next bankno=#1C
4e =#1C st.adrs=#1BD0 length=#560 next bankno=0
The last banknumber is 0 because there is no next banknumber.
Our execution address=#0400

Now we are going to put the information in the buffers with the substitute command in utility mode.

The first bank
S3000 00 04 F0 07 00 04 01 0C 01

Respectively: start adrs, length, execution adrs, utility flag, next banknumber and flag for execution.
The data for the first bank will be from #3000-#37FF= 2kilobyte

The second bank
S3800 F0 0B F0 07 00 00 FF 14 01

Respectively: start adrs, length, no execution adrs needed so double zero, no start bank so FF, next banknumber, execution flag
The data for the second bank is from #3800 till #3FFF.

The third bank
S4000 E0 13 F0 07 00 00 FF 1C 01

The data for the third bank is from #4000 till #47FF.

The fourth bank
S4800 D0 1B 60 05 00 00 FF 00 01

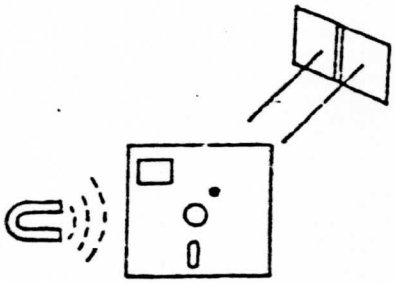
The data for the fourth bank is from #4800 till #4FFF.

If you program this data in an 8kbyte eprom and put this eprom in socket-4, you can then issue the BANK1-command to get this program loaded at #400 till #2130 and executed afterwards.

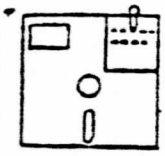
4. Handling Mini floppy Disks

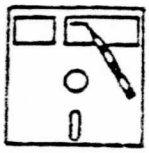
The following are the precautions to be observed when handling mini floppy disks.

[Unsatisfactory]

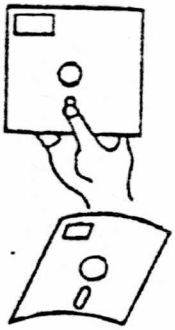


- o Do not expose disks to direct sunlight or place them near a source of heat.
- o Do not place disks in a place which is subject to the influence of a magnetic field.
- o Do not expose disks to cigarette smoke.
- o Do not put clips or rubber bands on disks.



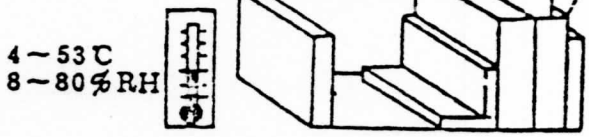


- o Do not write directly on disks using a pen or pencil.

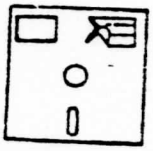


- o Do not touch the recording face of disks (oblong hole portion).
- o Do not bend or fold disks.

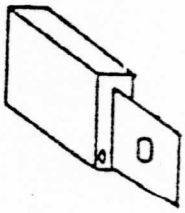
[Satisfactory]



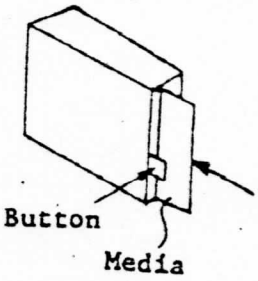
- o Store disks in a clean environment at suitable temperature and humidity.
- o When not using a disk, insert it in an envelope, then insert the envelope in a special-purpose case, and store it vertically.



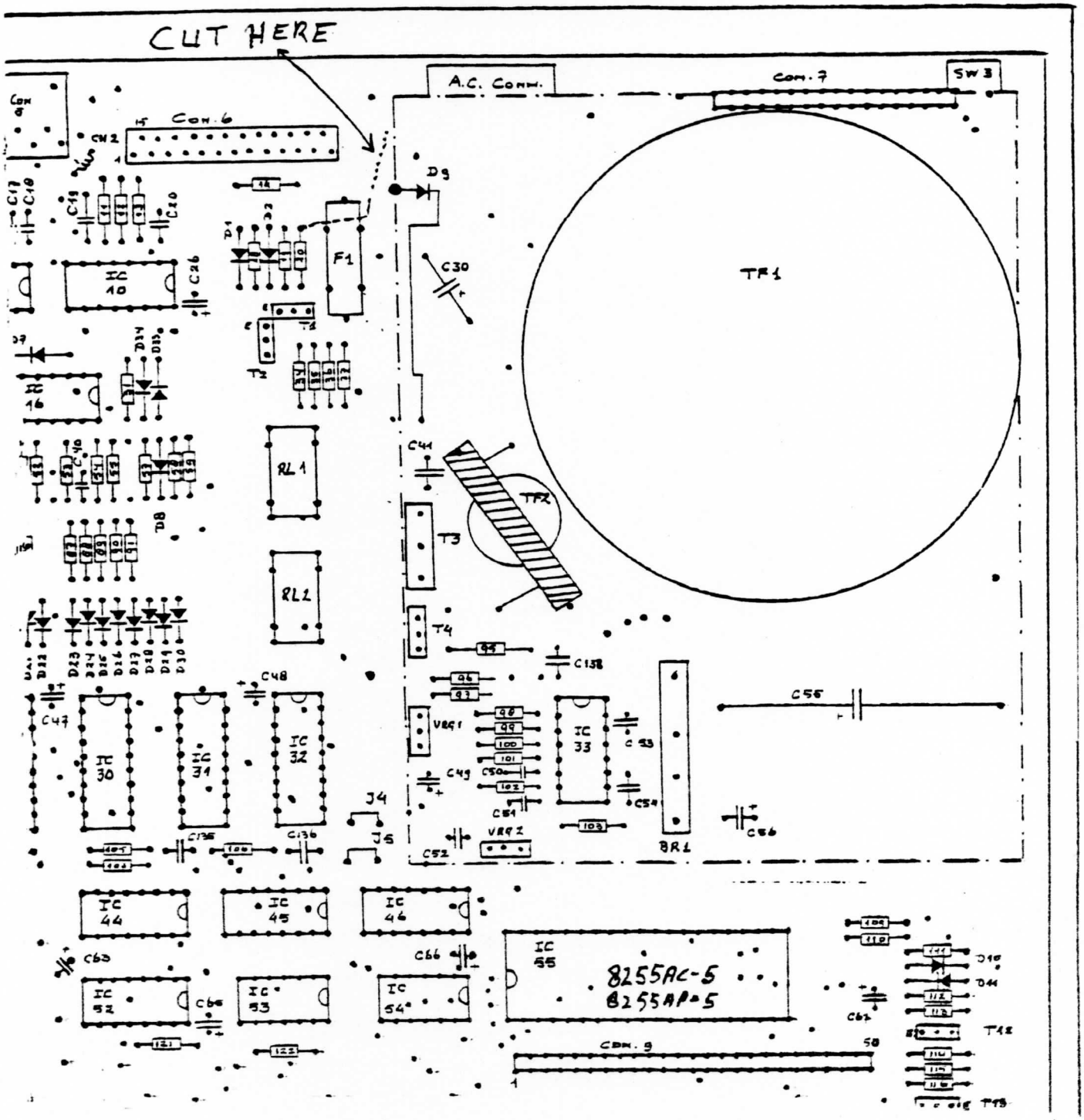
- o Paste labels on disks after writing on them first.



- o Before using a disk, it is recommended that it be left for a suitable time in the same environment as the drive in order to acclimatize it.



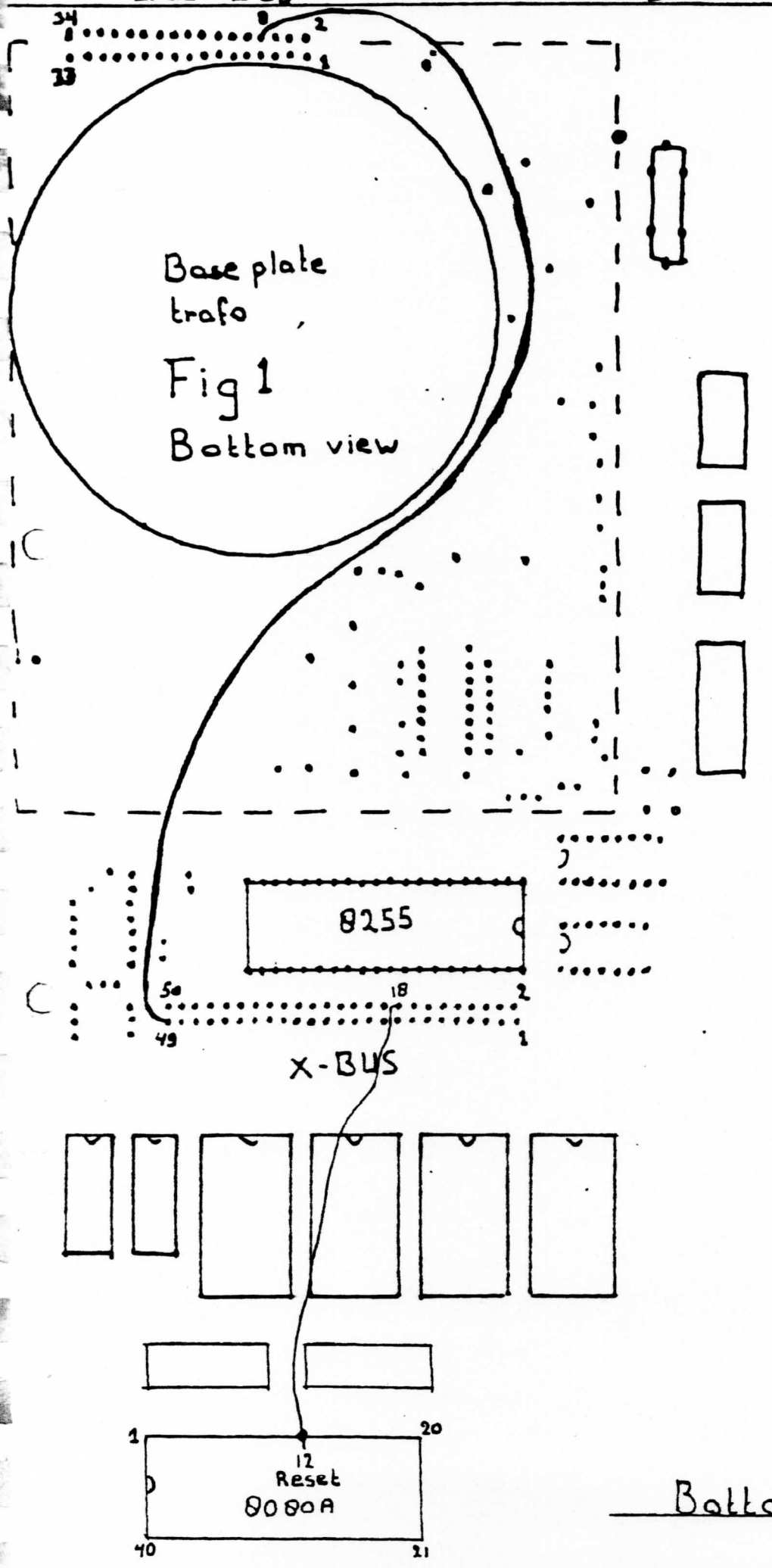
- o Completely insert the media to the back of the drive before pushing the button.



An additional modification is necessary to have a better writing performance. From resistor-20 the interrupt line for the stack-overflow is routed to the DCE-bus. Cut this connection with a sharp knife. If your 8255 is not of the type 8255AC-5 or 8255AP-5 you could have problems in getting the system working. Remove this I.C.(IC 55) and replace it with a proper one.

D.C.E.-Bus

Fig 1



Connect pin-8 of the DCE-bus with pin-49 of the X-bus.

Connect pin-18 of the X-bus with pin 12 of the 8080A

Bottom view

To be able to write to eprom-socket no.6 the User must solder the solder-island 'W' which is located between socket-5 and socket no.6. On the rear-side of the epromcard (fig.3) at point 'W' the lead must be cut to finish this modification.
Hereafter the user can insert a 6116 static-ram into socket-6. T
On writing to the specified socket the socket must be selected by writing '0B ' ØD `to hex F9ØØ.
First zero hex 296.
Writing to hex FØØØ-F7FF will cause a stack-interrupt.

Fig 2

For selecting dif. eproms see fig.3.

The User must disable the interrupts before writing to hex FØØØ-F7FF.

```
example: BUFFER EQU :AØØØ      ;data for bank-6
         ORG :Ø4ØØ
START    PUSH ALL
         LXI H,BUFFER
         LXI D,:FØØØ
         MVI C,100      ; write 100 bytes
         DI             ; disable interrupts
LOOP     MOV A,M
         INX H
         STAX D
         INX D
         DCR C          ; decrement count
         JNZ LOOP
         EI
         POP ALL
         RET
```

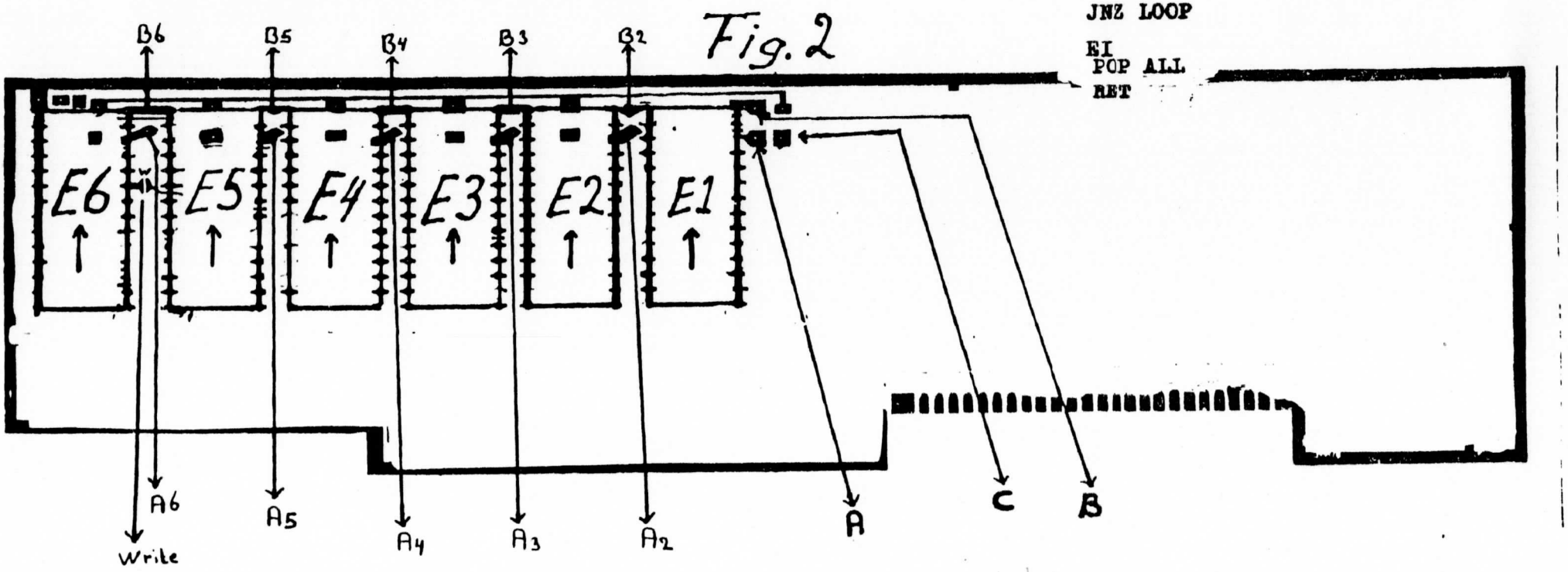
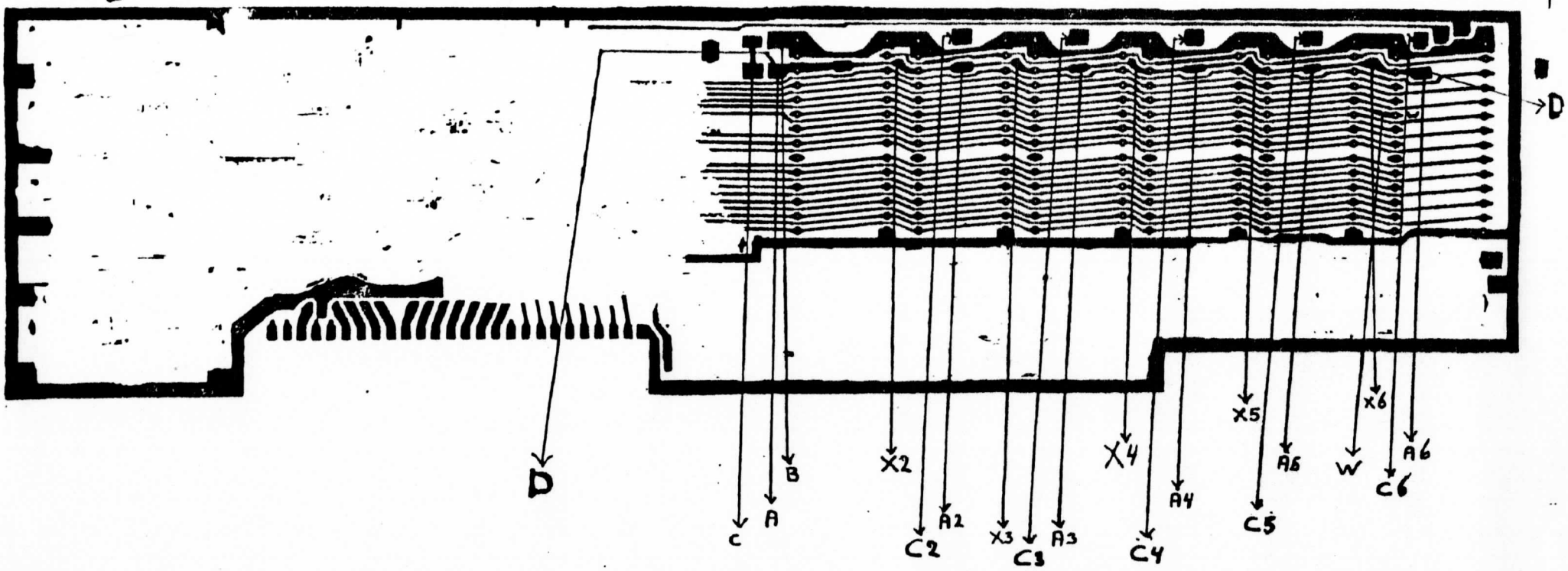


Fig. 2

Fig 3.



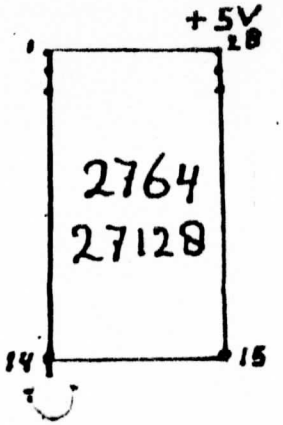
Normaly A is connected to B. In this mode all the sockets are set for 2716/2732/2764 eproms.

To be able to use 27128 eproms it is necessary to disconnect A and B and to connect C to A.

In this mode all the sockets will be set for 27128 only.

If it is desirable to set the sockets individual, the line D must be interrupted before X. A2 and C2 can be connected for 27128 or (see fig.2) A2 can be connected to B2 for 2716/2732 and 2764 . Etc. for the remaining sockets.

To interrupt line D a sharp knife must be used, but beoareful not to cut other leeds.



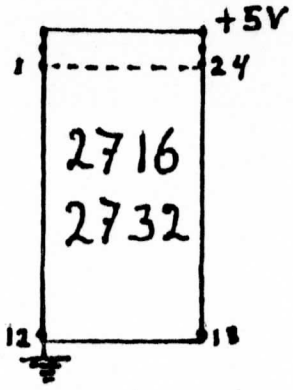
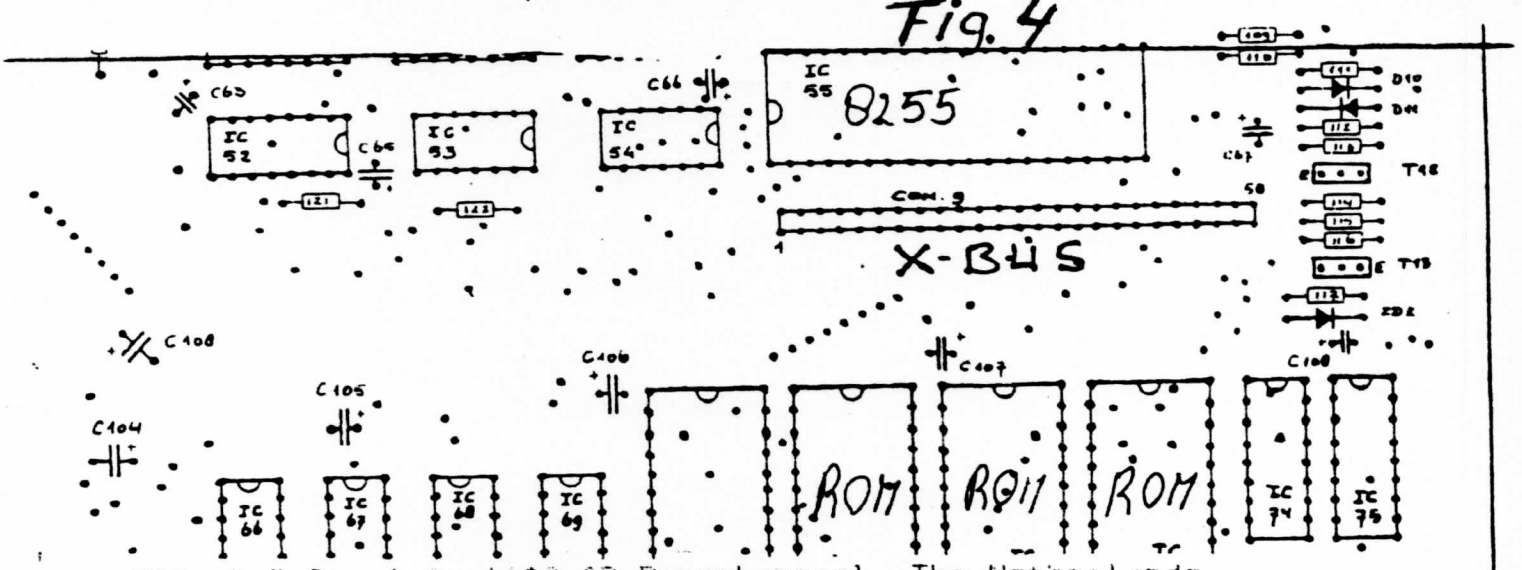
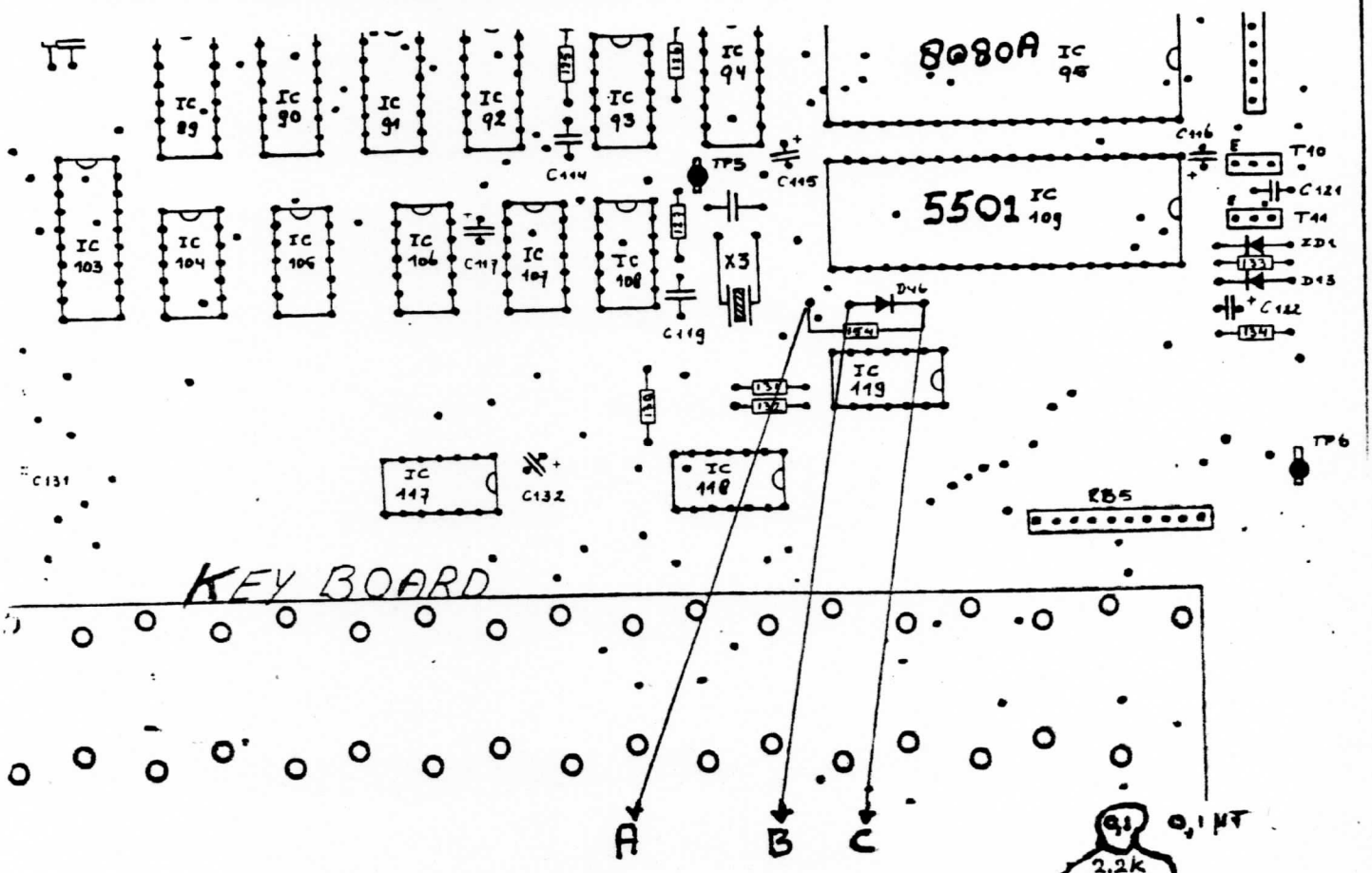


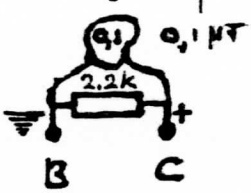
Fig. 4



MTPI, P.O.Box 180, 1810 AD Bovenkarspel, The Netherlands

ATTENTION!!!!!
Make this modification if you have stack-overflow (break with KENDOS) PROBLEMS. If your computer doesn't function well restore the old situation.





Remove <u>all</u> components between A, B and C
A diode and a resistor or only a resistor.
On some computers even more components.
Solder a 0.1 µf tantal condensat and a 2.2 kΩ resistor between B and C.
Be sure that the '+' of the condensat is connected to C.

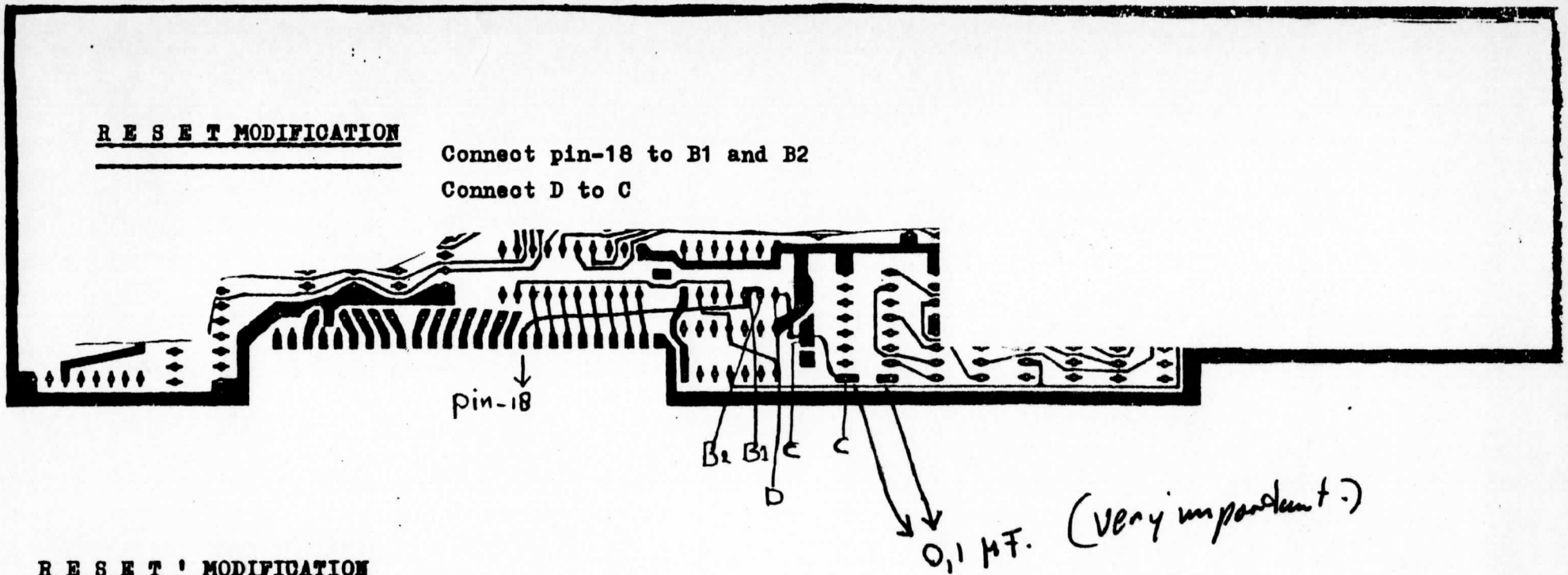
A = -5 V
B = ground
C = stack interrupt.

TOP VIEW

Fig. 5

R E S E T MODIFICATION

Connect pin-18 to B1 and B2
Connect D to C



R E S E T ' MODIFICATION

On some eprom-cards this modification is already done.
(The latest version is already modified)
This MODIFICATION will make it possible to return always to BANK-ø on RESET. This will prevent the system to 'hang' after a reset. See for additional modification fig.1.
(connect pin-18 of the X-bus to pin-12 of the 8080A.)

F990 1e byte n sec to read
2e, 3e, 4e, 5e, 6e } byte sector number to read. first the highest etc.
after every read or write update this buffer.
F995 = track number buffer.

first put 2ø6 to ø
put Føøø to 8
put track number in
CALL init (FØ2D)
select track
CALL seek. (FØ12)

push all
put buffer address in HL
CALL read or write
SET Buffer at F990
CALL READ or write
R = F000 W = F003
pop all. ret.