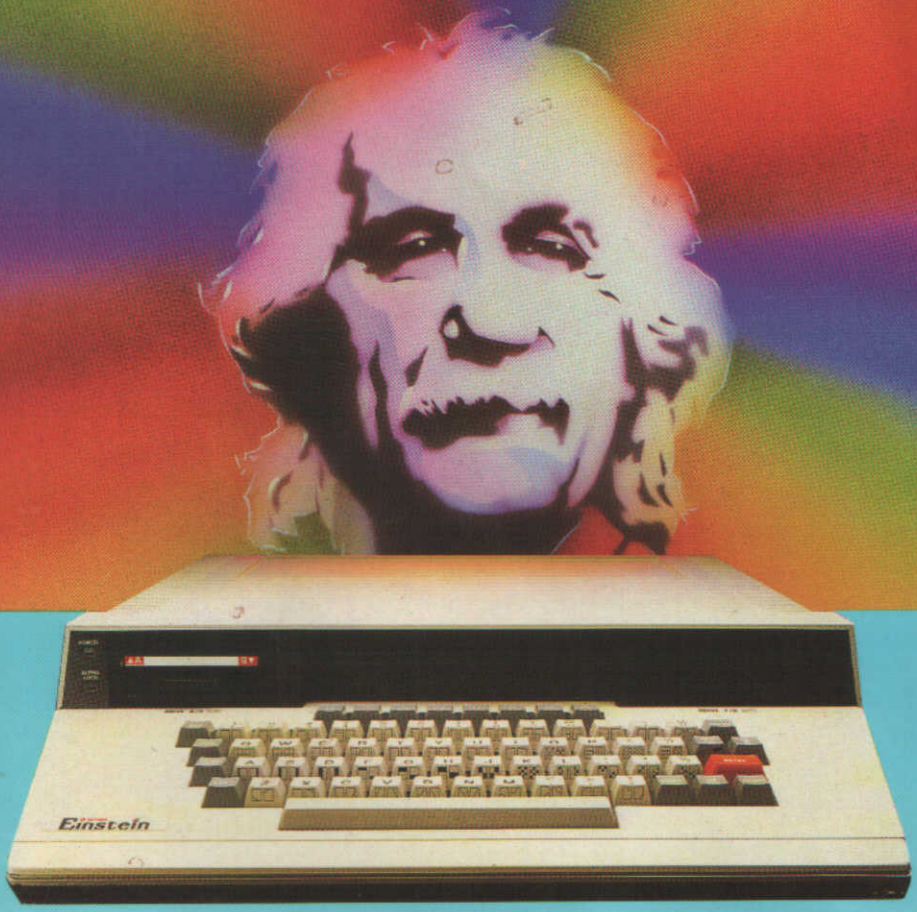
DOS/MOS Introduction

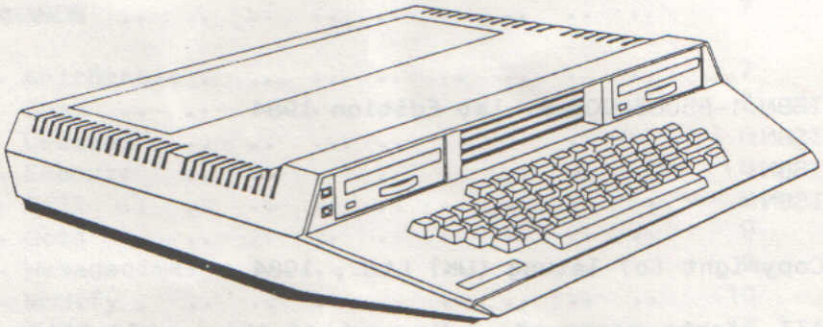


TATUNG

Einstein

COLOUR MICRO COMPUTER

TATUNG Einstein



DOS / MOS
Introduction

Acknowledgements

Written By: Alan Stancliffe
Edited by: R.M. Clarke

Our grateful thanks to Crystal Research Limited of Torquay, for their kind permission to reproduce extracts from their manuals, and for their help in checking this publication.

ISBN:1-85086-006-8. 1st Edition 1984
ISBN:
ISBN:
ISBN:

Tatung (UK) Ltd.,
Computer Division,
BRIDGNORTH,
Shropshire. WV15 6BQ

First Printed in 1984 by:
Spellman Walker Ltd.,
Bradford, West Yorkshire.
ENGLAND.

CONTENTS

SECTION B

INTRODUCTION

This manual consists of two sections. Section A deals with the Machine Code Monitor and outlines the commands and facilities available when working at that level with EINSTEIN. Section B deals with the Disc Operating System (DOS) outlining the conventions used, and the facilities available to the user when working with EINSTEIN.

The Machine Code Monitor is only one part of the overall Machine Operating System (MOS). It is therefore not within the scope of this manual to cover the complete operating system of EINSTEIN.

The Machine Operating System is contained within an 8k by 8 bit ROM integrated circuit situated on the main printed circuit board of the computer.

The Disc Operating System (DOS), as developed by Crystal Reasearch Limited, is designed as a portable system and must be loaded into RAM from the SYSTEM disc supplied with EINSTEIN. Xtal DOS, (as it is known) loades into the top of RAM (The TATUNG/Xtal BASIC 4 interpreter loads into the bottom of RAM from 0100H)

SECTION A

MACHINE CODE MONITOR SYSTEM

This is one section of the overall MACHINE OPERATING SYSTEM which is available to the user, working at the lowest level of operation with the computer. It allows direct access to the memory and various facilities are available for the manipulation and display of data contained therein.

The method of access to the "monitor" depends on the current "state" of the computer. Access can be made as follows:-

a) From BASIC - Type the command MOS and key ENTER.

b) From DOS (Disc Operating System) - Type the command MOS and key ENTER.

c) From power up, or reset, with a disc in the drive unit the computer will come up in DOS - therefore type MOS and key ENTER.

d) From power up, or reset, without a disc in the drive unit - the computer will come up in MOS.

e) The "prompt" character when in MOS is a chevron which appears to the left of the cursor when MOS is initially accessed as illustrated below.

SCREEN EDITING

When working in MOS the screen editing facilities of the INS/DEL key and the cursor movement keys are available.

Various facilities are also offered by the use of control codes.

SCREEN CONTROL CODES

The control key, when operated in conjunction with other character keys, provides the following facilities which assist output to the screen whilst working in MOS.

1. **CTRL-J** Line Feed (LF)

   This will create a "line feed" (move to the next line down). In other words the cursor moves down one line and there is a "repeat" function if the keys are held down.

   When the bottom of the screen is reached, the display will "scroll up" one line at a time. Again the "repeat" function will operate if the keys are held down.

2. **CTRL-L** Cursor Home and Clear Screen.

   This will "clear" the screen and move the cursor to the "home" position (top left hand corner of the screen).

3. **CTRL-M** Carriage Return as ENTER

   This will operate a "carriage return" **with** a line feed (ie. move the cursor to the beginning of the next line).

4. **CTRL-A** Screen Dump to Printer

Will cause a transfer of the screen display contents to a printer (ie. make a hard copy on paper)

5. **CTRL-H** Backspace (BS)

This will simply move the cursor to the LEFT one character space at a time. The function will "repeat" if the keys are held down.

6. **CTRL-D** Horizontal Tabulation (HT)

This will move the cursor to the RIGHT one character space at a time. The function will repeat if the keys are held down.

7. **CTRL-G**

This will invoke the "Beep" sound (ie. cause a 880Hz tone to be sounded)

8. **CTRL-K** Vertical Tabulation (VT)

This will move the cursor UP one line at a time and there is a "repeat" function if the keys are held down.

9. **CTRL-T** Cursor OFF

This will turn the cursor off, should this be required.

10. **CTRL-Q** Cursor ON

This will turn the cursor back on again as a reversal of the CTRL-T function.

11. **CTRL-R** Printer ON

This activates the printer such that any data output to the screen will also be output to the printer. (The data normally coming from the keyboard, or, from the display of text files by use of the DISP command within the Disc Operating System).

12. **CTRL-S** Printer OFF

This simply turns the printer off as a direct reversal of the operation in CTRL-R.

13. **CTRL-↑** Cursor Home

This returns the cursor to the "home" position (top left hand corner of the screen) without affecting the screen contents.

14. **CTRL-X** Erase Whole Line

This returns the cursor to the beginning of a line and then erases to the end of the line.

15. **CTRL-U** Erase to End of Line

This will erase to the end of a line from the current cursor position.

16. **CTRL-V** Erase to End of Screen

This will erase to the end of the screen from the current cursor positon.

17. **CTRL-Y** Delete Character
or
DEL

This deletes a character to the left of the cursor, moving the remainder of the line one character space to the left.

18. **CTRL-F** Delete Character at Cursor
or
CTRL-DEL

Either of these combinations will delete a character at the cursor, moving the remainder of the line one character space to the left.

19. **CTRL-N**

This clears the screen and sets 40 column display.

20. **CTRL-O**

This clears the screen and sets 32 column display

21. **CTRL-Z**
or
INS

This will insert a character space at the cursor position, at the same time moving the existing text one space to the right of that position.

Each command consists of a single letter (either upper or lower case may be used) which in some cases is followed by either,1,2,3, or 4 hexadecimal numbers. These hexadecimal numbers may contain up to four HEX digits. The exception is the H command which requires a single decimal number.

ARITHMETIC COMMAND.

Syntax: A xxxx yyyy

Where 'xxxx' and 'yyyy' are given as two hexadecimal numbers.

Purpose: This command calculates the sum, difference, and the necessary "offset" for a relative jump for the two numbers given.

The results are displayed as follows:-

In cases where a "relative jump" would not be possible for the two given values then '--' is displayed.

COPY COMMAND.

Syntax: C xxxx yyyy zzzz

Purpose: This command is used to copy a block of memory which starts at address 'xxxx' and finishes at address 'yyyy'. The block is copied into a new position beginning at the address given by 'zzzz'.

DECIMAL COMMAND.

Syntax: D xxxx

Purpose: This command converts the hexadecimal number given in 'xxxx' to a decimal number in the range 0 to 65535 and then displays the result on the next line.

Example:

D 003E will give a result of 62 decimal.

EXECUTE COMMAND.

Syntax: E xxxx

Purpose: This command will execute a program starting at the current PROGRAM COUNTER (PC) value as given by the Z command and continuing until the address given by 'xxxx' is encountered, at which point a break will occur and the register contents will be displayed. If 'xxxx' is not given, or is given as zero, then no break will occur.

Register Display:

A BC DE HL PC SZ-H-PNC

The registers would be displayed as above with their current values/contents shown below them. (see Z command for further details of registers).

FILL COMMAND.

Syntax: F xxxx yyyy zz

Purpose: This command will fill a block of memory starting at the location given by 'xxxx' and ending at location 'yyyy', with the value given by 'zz'.

Example:

```
F OFAO OFFF 23
```

This will fill the block of memory from location OFAO to OFFF with 23 HEX (23 being the HEX value which represents the # symbol in the character code table). The T command could then be used to examine the memory locations given above.

GOTO COMMAND.

Syntax: G xxxx yyyy

Purpose: This command causes execution to 'go to' the program starting at address 'xxxx'. If 'yyyy' is given a value then a break point will occur at the location given by 'yyyy' (as in the E command). If xxxx yyyy are both omitted a G0000 will be performed

Example:

```
G B002 BOFF
```

This instructs execution to transfer to location B002 and carry out the routines from there onwards. When location BOFF is reached a break point occurs and the Z80 registers will be displayed with their current contents.

HEXADECIMAL COMMAND.

Syntax: H ddddd

Purpose: This command converts the decimal number given by 'ddddd' (in the range 0 to 65535) to a hexadecimal number and displays the result on the next line.

Example:

H 00246 will give a result of 00F6(HEX)

MODIFY COMMAND.

Syntax: M xxxx

Purpose: This command is used to modify an area of memory starting from the address given by 'xxxx' and the procedure is as follows.

a) Type in the command; followed by the starting address.

b) The address and its contents are displayed on the screen with the cursor positioned at the first digit of the contents.

c) To modify the contents, type in the new values as a two digit HEX number, overtyping the existing contents, and press ENTER.

d) The next address and data then appears, again with the cursor positioned at the first digit of the contents.

e) If no modification is to be made then simply press ENTER so as to examine the next address, and so on.

f) To EXIT the modify command type a full stop (.) at cursor position and press ENTER.

It is possible to modify the contents of consecutive addresses by typing the successive bytes on one line. When ENTER is pressed the next **unmodified** address is displayed.

To examine a different address when using the modify command it is possible to delete to the start of a line and type in the new address followed by ENTER. The contents of that address then appear on the display as usual.

Example:

M 0B2C

This produces a display commencing from the address given (ie. 0B2C) and then continues from there as modifications are made.

Example Display:

```
0B2C 22          (The contents of the addresses
0B2D 41           are given as examples only)
0B2E 3A
0B2F CD
0B30 23
0B31 4F
0B32  F
```

The contents of address 0B2C to 0B31 have been examined in turn. The cursor is shown positioned at the first digit of the contents of the location currently being examined (0B32). After each of the two digits have been examined and the ENTER key pressed, the next location will appear on the display

NOTE: Mistakes in previous lines may be corrected by using the cursor control keys to transfer back to a particular line and then type in the correct values over the incorrect values. Press ENTER whilst still on the corrected line in order to execute the modification and then continue.

READ COMMAND.

Syntax: R xxxx yyyy sstt d

Purpose: This command will read a block of data from the disc currently in the drive specified by d. The 'read' will start from track tt, sector ss and the block will be transferred into memory starting at location 'xxxx' and ending at location 'yyyy'.

The range of values for each of the parameters is as follows:-

Parameters	Range of Values (HEX)
xxxx	0000 to FFFF
yyyy	0000 to FFFF
ss	0 to 9 (default value = 0)
tt	0 to 27 (default value = 0)
d	0 to 3 (default value = 0)

Thus using this facility data may be transferred from any specified area of the disc into memory, as selected by the user.

TABULATE COMMAND.

Syntax: T xxxx yyyy zz

Purpose: This command will tabulate the contents of memory starting from the address given by 'xxxx' and ending at address 'yyyy'. If zz is **not** specified the memory will be displayed in blocks of 8 bytes, otherwise in blocks of zz bytes. The command can be abandoned by pressing the 'ESC' key.

Example:

```
T 0100 0120
```

This will produce a display similar to the following (the contents of locations being given as examples only).

```
0100 C3 66 2B C3 79 07 01 3D Cf+Cy..=
0108 BF 39 78 3B C4 38 3D 39 ?9x;D8=9
0110 1A 3D A2 3D E8 3B D2 06 .="=h;R.
0118 EA 2B 0F 00 80 00 7A 01 j+....z.
```

LOCATION

CONTENTS OF EACH LOCATION

CHARACTERS WHICH CORRESPOND TO THE VALUES GIVEN IN THE LOCATIONS

Each line of the listing begins from the left with a location number. This is followed by the current value in that location and then each value of the next 7 consecutive locations. The right hand end of the line gives the corresponding characters represented by the values in the same consecutive sequence.

Looking at the first line of the listing, 0100 is the first location and C3 is the value contained therein. Then 66 is the value in 0101, 2B is the value in 0102, and so on up to location 0107 which contains the value 3E. The characters illustrated show that the value in the first location (0100) represents the letter C, the second location value represents the letter f, the third location value represents the + sign, and so on up to location 0107 which represent the = sign. Full stops indicate non displayable characters (eg. control codes)

The number of consecutive locations given on one line can be varied by use of the zz parameter which specifies the actual number to be displayed.

NOTE:

1. All memory locations (addresses) and values contained therein are given in Hexadecimal.

2. The characters displayed in the right hand section of each line ignore the most significant bit of the corresponding code. An example of this appears in the first line of the display given above. C3 is given as the value representing the character C.

   Thus:-

   C3-80 = 43 (ie. ASCII value for character C).

WRITE COMMAND.

Syntax: W xxxx yyyy sstt d

Purpose: This command will write a block of memory starting at address given by 'xxxx' and ending at address 'yyyy' to a disc currently in the drive specified by d. The data will be entered on the disc starting at the track tt, sector ss.

The range of values for each of the parameters is as follows:-

Parameter	Range of Values (HEX)
xxxx	0000 to FFFF
yyyy	0000 to FFFF
ss	0 to 9 (default value = 0)
tt	0 to 27 (default value = 0)
d	0 to 3 (default value = 0)

Thus using this facility data may be transferred to any specified area of a disc as selected by the user.

COLD START COMMAND.

Syntax: X

Purpose: *This command will cause a "cold start" tansfer* to the current program from MOS. (Always provided the execution vector has been passed into MOS by the program loaded, otherwise performs a G0100)

When using this command to return to BASIC from MOS all programs and variables are lost and the transfer is as if BASIC had just been loaded. Hence the term "cold start".

WARM START COMMAND.

Syntax: Y

Purpose: This command will cause a "warm start" transfer to the current program from MOS.

When using this command to return to BASIC all programs and variables are preserved. If a break has been made in a program then on return execution will continue from that point onwards. Hence the term "warm start".

The "warm start" vector is equivalent to a G0103.

DISPLAY REGISTERS COMMAND.

Syntax: Zx

Purpose: This command will display the Z80 register contents according to the value specified by x as indicated below.

Z0 - Displays normal registers.
A,BC,DE,HL,PC and Flags.
Z1 - Displays alternate registers.
A',BC',DE',HL',PC and Flags'.
Z2 - Displays special registers.
I,IX,IY, SP and PC.

Registers and Flags

MAIN REGISTER SET

A	Accumulator	F	Flag Register
B	General Purpose	C	General Purpose
D	General Purpose	E	General Purpose
H	General Purpose	L	General Purpose

ALTERNATE REGISTER SET

A' Accumulator	F' Flag Register
B' General Purpose	C' General Purpose
D' General Purpose	E' General Purpose
H' General Purpose	L' General Purpose

F Register

S	-	Sign Flag
Z	-	Zero Flag
-	-	Not Used
H	-	Half Carry Flag
-	-	Not Used
P	-	Parity Flag
N	-	Subtract Flag
C	-	Carry Flag

SPECIAL REGISTERS

I	-	Interrupt Register
IX	-	Index Register
IY	-	Index Register
SP	-	Stack Pointer
PC	-	Program Counter

Display:

The registers are normally displayed as shown in the examples below (the contents shown are examples only)

Example

A	BC	DE	HL	PC	SZ-H-PNC
00	0000	0000	0000	B002	00010100

Example

I	IX	IY	SP	PC
FB	0000	0000	FCFF	B004

NOTE:

a) PC is the same for all three displays.
b) The R (Refresh) register is not given since its value is constantly changing.
c) An automatic Z0 occurs after a break from a program.

BREAK

A "break" may be 'patched' into any program by inserting an FF code at the desired location in memory, in place of an op-code.

BAUD RATE COMMAND.

Syntax: B xy ww22

Purpose: This command sets up the baud rate for the RS232-C Port according to the values specified by x and y. x is the "Receive rate" and y is the "Transmit rate" and can be a number in the range 0 to 8 representing baud rate values as given in the table below.

0	-	75 baud	5	-	1200 baud
1	-	110 baud	6	-	2400 baud
2	-	150 baud	7	-	4800 baud
3	-	300 baud	8	-	9600 baud
4	-	600 baud			

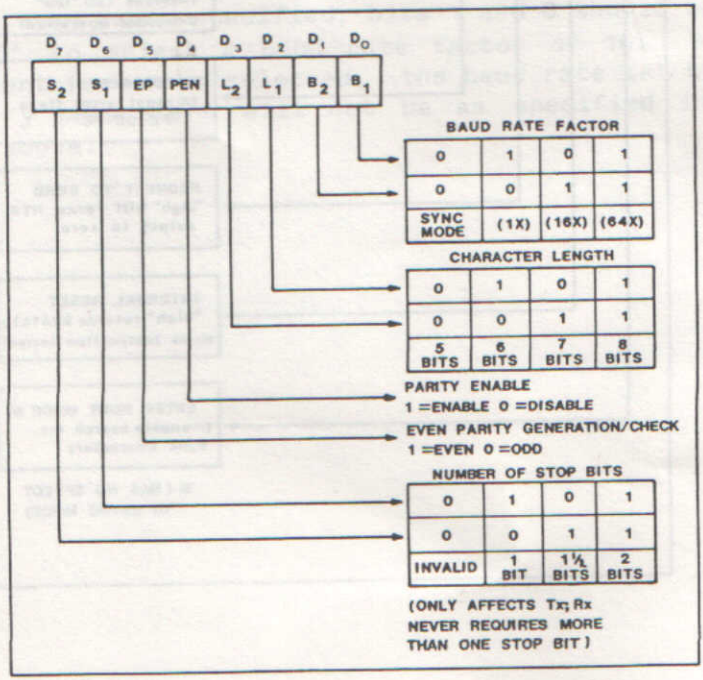
If only one of x or y is specified then both will be set to the same baud rate.

Any mix of baud rates for x and y is permissible with the following exception - 75 bauds can only be set to receive if 75 bauds is **also** set to transmit.

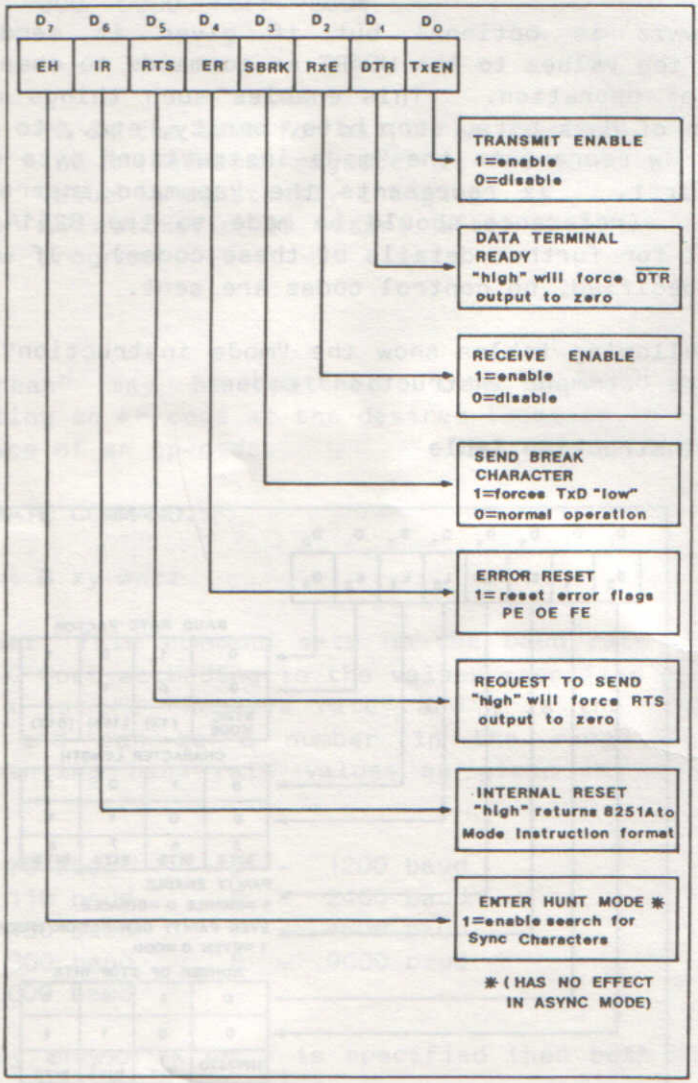
The wwzz is optional but if given it sends the specified values to the USART as commands to change its mode of operation. This enables such things as the number of data bits, stop bits, parity, etc., to be set up. ww represents the "mode instruction" byte and is set first. zz represents the "command instruction" byte. (reference should be made to the 8251A USART manual for further details of these codes). If wwzz is not specified, no control codes are sent.

The following tables show the "mode instruction" codes and the "command instruction" codes.

Mode Instruction Table



Command Instruction Table



Examples:

B7 Sets Rx and Tx rates to 4800 baud

B35 Sets Rx rate at 300 baud, and Tx rate at 1200 baud.

B8 CE27 Sets Rx and Tx rates to 9600 baud. Also sets up the 8251A USART as follows:-

Asynchronous
clocks at 16x baud rate
8 data bits
no parity
2 stop bits
Transmit and Receive enabled
DTR and RTS outputs forced low.

(This is the default setting of the 8251A USART on Power Up)

NOTE: When ww is specified, bits 1 and 0 should be set to '10' to select a baud rate factor of 16. If a different factor is selected, the baud rate set by the x and y parameters will not be as specified in the table above.

SECTION B

DISC OPERATING SYSTEM

DISC OPERATING SYSTEM (DOS)

The Disc Operating System (abbreviated to DOS) for EINSTEIN has been designed as a portable system and as such is situated on the system disc supplied with the computer. For any "formatted" disc, tracks 0 and 1 are the "system tracks" which contain the DOS.

The DOS has control over the operations relating to discs and access of disc drive units, and will run CP/M programs. When loaded, the DOS and MOS combine to provide a complete operating system with many useful facilities which include all file maintenance activities (Reading from, Writing to, updating, appending etc.), utility programs such as BACKUP/FORMAT and COPY, general housekeeping activities and control of the disc directory (allowing the naming of files). The DOS also allows the user to load and run programs, and set passwords.

Under DOS the disc drives are referred to by a number in the range 0 to 3 (0 and 1 being the integral drives, 2 and 3 being external drives). When operating in DOS the prompt at the beginning of a line (to the left of the cursor) consists of the current drive number being accessed (default value being 0) followed by a colon as illustrated below:

```
0:
```

TO ACCESS DOS

The method of access to the Disc Operating System depends on the current "state" of the computer. A disc with system tracks must be placed in the current drive and access to the DOS can then be made as follows:-

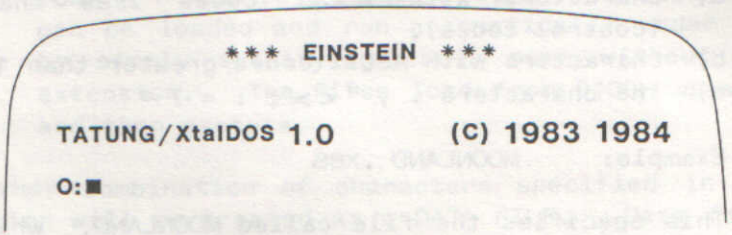
a) From BASIC - type the command DOS and key ENTER.

b) From MOS (with a disc in the drive unit) -
Press CTRL and BREAK keys or

c) From MOS, or power up, without a disc in the drive unit -
First insert the disc and then access DOS as in 'b' above.

d) From power up with a disc in the drive unit -
In this case the computer comes up in DOS.

When DOS is first entered the following should appear on the display:-



The majority of work undertaken in DOS is concerned with processing files of one kind or another. The following section describes the file naming conventions used in relation to this work.

FILE NAMING CONVENTIONS

Filenames

The standard file name convention is as follows:-

filename . extension

where:-

filename - is the title of the file selected by the user and can be up to 8 characters in length.

extension- specifies the type of file and can be up to 3 characters in length.

Both filename and extension may consist of any combination of ASCII characters with the following exceptions:-

a) characters with ASCII codes less than 32 (control codes).
b) characters with ASCII codes greater than 127
c) the characters . , " < > ; : = ? *

Example: *MOONLAND .XBS*

This specifies the file called MOONLAND, which is a BASIC file (as denoted by the extension .XBS).

Standard Extensions

The following are used as "standard" extension to denote specific types of file.

1. **.XBS** - This is a BASIC source file (ie. a normal program file). If the file type is not specified then XBS is assumed. These are text files which contain "tokens" for reserved words.

2. **.ASC** - This indicates on ASCII program file. These files are uncompressed source files. Whereas .XBS files contain tokens for reserved words, .ASC files contain the words in full as they appear in the LIST.

3. **.OBJ** - This is an object file, or machine-code subroutine/data. An area can be set up within memory for the storage of machine-code routines by use of the CLEAR command in BASIC. The .OBJ files can then be loaded to this area and subsequently linked into a BASIC program.

4. **.COM** - This is a command file. These files can be loaded and run automatically under DOS by simply specifying their name without the extension. The files load from 0100H upwards and then execute.

Any other combination of characters specified in the extension will be treated as a DATA FILE. Data files consist of a series of ASCII characters divided into one or more records (serial data) which can be accessed by a BASIC program. Users may select their own combinations for data files and the following are given as examples:-

.DAT - data file
.DOC - document files
.BAK - backup file
.GRA - graphics file

NOTE:

1. The standard extension .XBS, .ASC, and .OBJ are only distinguished under BASIC. When in DOS these files are all regarded as DATA files.

2. .ASC files can be listed to the screen by use of the DISP command. This produces a similar effect to LIST for a .XBS file in BASIC.

Drive Number

When using files from several discs on the system at the same time the drive number, followed by a colon, can be specified in front of the file-name. When omitted, the current default drive number is assumed (initially set to 0)

Examples:

```
0:PINGPONG.ABC - refers to the file PINGPONG.ABC on
                 drive 0
2:WXYZ.ASC     - refers to the file WXYZ.ASC on drive 2
SILLY          - refers to the file SILLY on the
                 current default drive
```

FILE DESCRIPTOR

This is a new concept used internally by the DOS which selects a small area of memory called a File Descriptor, which is used to specify a particular file required. This area contains information on the

area(s) of the disc allocated to the file, any attributes that the file may have, its size, and which drive it belongs to.

The layout of a file descriptor is as follows:-

```
FDESC: DRIVE   1 byte  - Drive number, 00=Default,
                         01=Drive 0, etc.
       FILNAM  8 bytes - File name (up to 8 characters)
       FILTYP  3 bytes - File type (up to 3 characters)
       EXT     1 byte  - Current extent number
       S1      1 byte  - Reserved for internal use
       S2      1 byte  - Reserved for internal use
       RC      1 byte  - Record count, no. of records in
                         this extent
       ALLOC  16 bytes - Allocation block information
       CR      1 byte  - Current record to read/write in
                         sequential file
       RECORD  2 bytes - Random Record number in the
                         range 0-65535
         +     1 overflow byte
```

FDESCs are held in the directory area of a disc, matched up with the FDESC specification in memory when performing functions such as Open, Create, Erase, etc., and then the allocation information is copied into this memory area. The memory copy is updated as file reading and writing proceeds, and is finally rewritten to disc when the file is Closed (and also, in fact, when an extent is filled on writing. In this case, the extent will be closed, a new extent is opened, with the new information applying to that extent, and operations then carry on).

The only section of the FDESC which directly affect the user are DRIVE, FILNAM, and FILTYP since these must be specified when naming a file, as described in the previous section.

DIR (Directory)

Syntax: DIR <filename>

Purpose: This command will display the directory of the disc currently in use, showing the files specified by <filename>. If <filename> is omitted then the whole directory will be displayed.

The DIR command will display up to the first 12 lines (24 entries) of a disc directory on the screen. If the directory is less than 12 lines it will be displayed in its entirety on the first entry of the command and the system prompt (drive number and colon) will then appear at the end of the display If, however, the directory extends beyond 12 lines then the cursor only will appear at the end of the initial display, indicating there is more to follow. To examine the remainder of the directory, the enter key must be pressed again. This will cause the next 12 lines of the directory to scroll up onto the display. The process is repeated until the system prompt (drive number and colon) appears on the display, indicating the end of the directory.

The total size of the files displayed is given at the bottom of the directory listing, together with the free space remaining and the total capacity of the drive.

The following convention is used to match filenames in a directory (multiple file specifications):-

i) A ? character will match with any character in that position in the filename found.

e.g.
X?X.ASC matches XYZ.ASC, XAZ.ASC, X(Z,ASC etc.
?X.D?T matches AX.DAT, BX.DZT, 4X.DZT, etc.

ii) The * character will match all characters at and after its position in the filename or type found.

e.g.
*.XBS matches any .XBS file
. matches any file and is the same as ????????.???
PROG*.ASC matches PROG1.ASC,PROG10.ASC,PROGABC .ASC,etc.

Thus the DIR command can be used to display the whole directory, certain common files only, or individual files. Any locked files are indicated with an * symbol in front of their name in the directory listing.

Example:

0:DIR

This will give a display of all the directory for the disc in drive 0, similary in format to the one given below.

```
0:*XBAS      .COM : MAIL     .XBS
0: BACKUP    .COM : COPY     .COM
0: XYZ,      .ABC :*TESTPROG.ASC
56k Size,  132k Free,  190k Total
```

This shows that the disc capaciy is 190k, the total size of files shown is 56k, and 132k is unused. Note that XBAS.COM and TESTPROG.ASC are locked, and so cannot be written to or RENamed.

Example:

```
0:DIR *.COM

0:*XBAS     .COM : BACKUP    .COM
0: COPY     .COM
39k size, 132k free, 190k Total
```

Example:

```
0:DIR XBAS .COM
0:*XBAS    .COM
16k Size,  162k Free, 190k Total
```

This last example shows how the size of any one file may be obtained by just performing a DIR with a single-file specification. If DIR of a non-existent file is requested, this will be returned as a OK size.

DISP (Display Files)

Syntax: DISP <filename>

Purpose: This command should be used with "ASCII files" only. It causes the contents of a data file to be displayed on the screen (ie. lists it).

Example:

```
DISP PET.ASC
```

This will display the text of the ASCII file named PET, on the screen.

DRIVE (Disc Drive)

Syntax: DRIVE L

L is specified as a number from 0 to 3 depending on the drive units available within individual systems.

If the drive specified in L is not available on the system a "Not Ready, Drive L error" will be given.

Purpose: This command sets up the default disc drive as specified by L for any subsequent access to a disc.

Example:

DRIVE 1

This selects drive 1 as the default drive.

ERA (ERASE)

Syntax: ERA <filename>

Purpose: This Command will erase the file, or a group of files, given by <file-name> from a disc in the current default drive.

The first file name conforming to the given specification will appear on the screen, together with a "?" symbol. The user must then type in one of the following single letters:-

Y - "Yes", erase this file. The word "Erased" will appear on the same line when that operation has completed.

N - "No", do NOT erase this file. The next file to be erased will then be displayed, if one exists.

A - "All", erase all files after and including the displayed file, displaying each name in turn, and the word "Erased" as each file is erased from the directory.

F - "Finish", abandon the ERA command, without erasing any more files (those previously shown as erased will remain erased).

If <file-name> does not exist, a "No File" error will appear on the screen.

If <file-name> is a **locked** file then the following message appears:-

```
File Lock,Drive O:
Unlock (Y/N)?
```

The user must then type in the following single letters:-

Y - "Yes", unlock the file and erase. The word "Erased" will appear on the same line when that operation has completed.

N - "No", do not unlock the file. This then abandons the ERA command, without erasing any more files.

NOTE: The **default drive** may be changed or re-selected by use of the DRIVE command.

GO

Syntax: GO

Purpose: This command will cause a jump to be made to location 0100 (HEX). The currently loaded program will then be executed without reloading it from disc.

LOAD

Syntax: LOAD **<filename>**

Purpose: This command is used to load a file from the disc into memory. The file will be loaded starting at memory location 0100 (HEX). The size of the file, specified as a number of 256 byte BLOCKS, is then displayed on the screen.

Example: - assume WOLF is an existing BASIC file.

LOAD WOLF.XBS

When loading is complete the following would be displayed on the screen:-

N BLOCK(S)

Where N is given as a decimal number

The file can be patched by returning to the "machine operating system" for modification etc., *or run by* means of the GO command.

Command Files:

Command files (.COM) will auto-execute by simply entering the filename without the use of LOAD. ie. they will load and run automatically without requiring further action from the user.

Example:

MAIL

If MAIL is a command file (.COM) it will load and run automatically.

An extension of this facility in DOS is illustrated by the following example:-

Let XBAS be a command file containing BASIC, and DEMO (a program file written in BASIC).

Then :-

XBAS DEMO

under DOS this will first of all load XBAS (BASIC) and then auto-execute the BASIC program called DEMO.

LOCK

Syntax: LOCK < filename >

Purpose: This command is used to lock a given file so that it cannot be rewritten or altered without first being UNLOCKED.

In addition LOCK may be used to turn the file into a 'SYSTEM' file, by including the letter S after the command. System files do not appear in the directory list but may be read or executed as required.

Example:

```
LOCK MAIL.XBS   - locks the file MAIL.XBS

LOCK XBAS.COM S - locks the file XBAS.COM and makes
                  it into a system file.
```

MOS (Machine Operating System)

Syntax: MOS

Purpose: This command is used to return to MOS. This would be useful if patches (modifications) are required for the currently loaded program/file. The modifications are made in MOS, the G command is used to return to DOS, and finally the program/file is saved using the SAVE command within DOS.

PSW (Password)

Syntax: PSW < password >

Where < password > is an 8 character name selected by the user and may contain any characters other than control characters. Note all 8 characters must be entered.

Purpose: This command sets up the password protection facility which can be used for security purposes to limit the access to any given file to authorised personnel only.

Once a password has been invoked any files saved can then only be loaded back under the same password. Any files which exist on the disc either without a password, or under a different password, cannot be loaded whilst the current password is in operation.

To change the password use PSW again with a different password. To turn off the password (or make sure that no password is in force!) use PSW by itself.

NOTES:

1. An unprotected file must be read back without a password being in force.

2. The password itself is not stored anywhere, therefore the user must know it or record it elsewhere.

3. There is no indication given in the directory that a file has been protected; the file can apparently be read, but appears to be complete rubbish.

4. The directory itself is unaffected by the password so that it is perfectly acceptable to mix unprotected files and files saved under various passwords stored on the same drive (as long as you know which are which!).

Example:

```
PSW IXZ247YT
SAVE MAIL.XBS
```

Having saved the file MAIL.XBS under the password IXZ247YT it can only be read or loaded back if that password is in force. Likewise other files not saved under this password cannot be read or loaded while it is in force.

REN (Rename)

Syntax: REN < old filename > TO < new filename >

Purpose: This command is used to rename an existing file, giving it a different title or extension.

Example - assume that HATS is an existing BASIC file.

```
   OLD NAME        NEW NAME

REN HATS.XBS  TO   BUTS.XBS
```

This will rename the existing BASIC file HATS giving it the new name BUTS.

Example:

```
REN HATS.XBS TO HATS.BAK
```

This will rename the extension of the file HATS from

If < filename > is a **locked** file then the following message appears:-

```
File Lock,DRIVE 0:
Unlock (Y/N)?
```

The user must then type in the following single letters:-

Y - "Yes", unlock the file and rename.

N - "NO", do not unlock the file. This then abandons the REN command, without renaming the file.

SAVE

Syntax: SAVE N <filename>

Where N = BLOCK size of file

Purpose: This command is used to save the file in memory, which has been modified, back to the disc. It is in fact a block of memory from 0100 (HEX) upwards which is transferred to the disc and therefore the block size for the particular file must be specified as a number of 256 byte BLOCKS.

Example

assume a BASIC file, named POT, has been loaded into memory for modification. After modification it occupies 2 BLOCKS. Therefore to transfer the file back to disc the following can be used:-

SAVE 2 POT.XBS

UNLOCK

Syntax: **UNLOCK < filename >**

Purpose: This command is used to unlock a previously locked file so that it may be rewritten or altered. If the file was also a system file, that attribute will automatically be removed by the UNLOCK command and it will then display normally in the directory list.

Example:

UNLOCK MAIL.XBS - unlocks the file MAIL.XBS (assuming it was previously locked).

BACKUP

The **BACKUP** utility provides the facility to **format** and to **backup** a disc. **Format** organises a disc side into tracks and sectors. A formatted disc side is ready to receive data. **Backup** transfers the contents of an entire side of a disc to another side of a disc.

To invoke this utility type **BACKUP** and press ENTER with the System disc in the current disc drive. The **BACKUP** menu is then displayed:-

```
DISC BACKUP & FORMAT V.0.8
(c) 1984 Xtal

Press "B" for BACKUP
   or "F" for FORMAT
   or "X" to exit
Which (B/F/X)?
```

The following points will enhance the instructions displayed by **BACKUP**.

Format

Pressing F at the **BACKUP** menu selects the **format** option.

1. Formatting with a single disc drive.

   Make certain that the write protect notch on the system disc is set to protect. Put the system disc back into the drive (side A upwards).

Follow the instructions provided by **format** until the report

```
OK -- Format Drive (0-3)?
```

is displayed. At this stage remove the system disc and insert the target disc with the side to be formatted upwards. Then continue to follow the instructions displayed.

During the process of formatting the "track numbers" are presented on the screen as below

```
0         1         2         3
01234567890123456789012345678901234567890123456789
```

(ie. 40 tracks in four groups of 10)

As each track is formatted in turn a letter F (for format) will appear below the track number.

```
0         1         2         3
01234567890123456789012345678901234567890123456789
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

When every track has been formatted, each one will then be verified and again a letter V (for verify) will appear below each letter F on the screen in turn as verification takes place.

```
0         1         2         3
01234567890123456789012345678901234567890123456789
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV
```

When formatting and verification are complete the following message will appear on the screen.

Disc Formatted and Verified -- OK
Press ENTER to continue
or 'X' to abandon

At this stage the target disc is formatted. The user may format another side of a disc or return to the **BACKUP** menu.

2. Formatting with multiple disc drives.

As with single drive formatting, make certain that the write protect notch on the system disc is set to protect. Return the system disc to the current disc drive and place the target disc, with the side to be formatted facing upwards, in any other disc drive connected to the system. At the report

OK -- Format drive (0-3)?

type the number of the disc drive containing the disc to be formatted.

The display during the process of formatting is identical to the display in the single drive version above.

The report

Disc Formatted and Verified -- OK
Press ENTER to continue
or 'X' to abandon

is displayed when the target disc is formatted.

Backup

Pressing B at the **BACKUP** menu selects the **backup** option.

Two disc drives must be declared in backing up; the source drive which will contain the disc side to be copied, and the destination drive which will contain the disc side which will receive the copy. If these are both declared as the same disc drive, the user is prompted to change between the source and destination disc side several times during the back up process.

Backup copies the whole side of a disc including the first two tracks containing the disc operating system.

When the operation is complete the report

Disc Backed up -- OK
Press ENTER to continue
or 'X' to abandon

is displayed.

COPY

This is a general-purpose file-transfer program operating from within the disc operating system and provides facilities for the transfer of data between peripheral devices.

The use of COPY will normally involve:-

1. Copying of single or multiple files from disc to disc, using two drives.
2. Copying of files from disc to disc using a single drive.

3. Copying of files between peripheral devices connected to the computer.

Copying Files:

The facility can be used to copy individual files from one disc to another or, a sequence of copies can be made one after the other. During the copying process the file name may be changed if so desired.

1. Type **COPY** file-input **TO** file-output and press ENTER

   (Where file-input and file-output include the respective drive numbers)

   In this form a single file will be copied from one disc to another in the drives specified and the command then terminates. When a drive is not specified the current default drive number is assumed.

   The file need only be given in the input specifications if it is to be copied to another disc under the same name.

   Examples:

   COPY 0:OLD.ABC TO 1:NEW.XYZ

   This copies the file OLD.ABC from drive 0 to drive 1 under the new name NEW.XYZ

   COPY 0:OLD.ABC TO 1:

   This copies the file OLD.ABC from drive 0 to drive 1 under the same name.

```
COPY OLD.ABC TO 1:NEW.XYZ
```

This copies the file OLD.ABC from the current default drive to drive 1 under the new name NEW.XYZ.

```
COPY 1:*.XBS TO 0:
```

This copies all .XBS files from drive 1 to drive 0.

2. Type **COPY** and press ENTER

In this form the command is used by itself and an "*" prompt appears on the screen after ENTER has been keyed. The file-input TO file-output statement (in any of the forms given above) is then typed in following the "*" prompt and ENTER is keyed. The copying process takes place and the "*" appears on the next line for another entry.

In this way a series of files may be copied one at a time under the single command. To terminate COPY simply key ENTER immediately following the "*" prompt.

NOTE: If file-input and file-output are identical (ie same name and same drive number) a prompt is displayed on the screen so the user may swap the discs when using a single drive for copying.

However, if the names of the files are different, then it is possible to make a copy of a file on the same disc under the new name.

Copying Using Multiple File Specifications:
Multiple file specifications (*,?,see Page 28) can be used in both the aforementioned versions of COPY and several files may then be copied at one go (even an entire disc). In this instance each file name is displayed on the screen followed by a question-mark prompt, before being copied. To continue the user must type a single letter response as follows:-

Y - Yes, copy this file.

N - No, do not copy this file, go to the next file.

A - All, copy this and all other appropriate files without further prompting.

F - Finish, terminate the COPY command.

Each file successfully copied displays the word "copied" to the right of the name on completion. The following shows an example of the type of display to be expected when using multiple file specifications.

```
0:COPY *.XBS TO 1:
CHESS    .XBS?Y copied
MUSIC    .XBS?N
CIRCLES  .XBS?Y copied
ELLIPSES.XBS?A copied
HANGMAN  .XBS   copied
TWITCH   .XBS   copied
0:
```

On a one drive system:-

```
0:COPY 0:*.XBS TO 0:
```

would copy all .XBS files and give prompts to swap discs.

Copy Using Peripherals/Devices:
Both versions of the COPY facility may be used in conjunction with peripherals devices in place of file names as shown in the format below.

Type **COPY** Device-input **TO** Device-output and press ENTER.

COPY can also be used in combination with files and devices as shown below.

Type **COPY** Device-input **TO** file-output and press ENTER.

Type **COPY** file-input **TO** Device-output and press ENTER.

The peripheral device names which are allowed in the COPY command are as follows:-

Logical Devices -
CON: console, input or output
AUX: auxiliary, input or output
LST: list, output only. An error message will result if LST: is used as an input device.

Physical Devices -
VDU: VDU, output only
KBD: keyboard, input only
SRL: serial RS232, input or output
LPT: parallel line printer, output only.

Examples:

COPY CON: TO LST:

This copies data input at the keyboard to the printer.

COPY SRL: TO 0:DAT.XBS

This copies the input from the RS232 to the disc in drive 0 under the name DAT.XBS

Additional Facilities/Options:

In addition to specifying the input and output within the COPY command, the user may also give a list of optional parameters. If used, this parameter list must follow the particular file or device it relates to, and be contained within chevrons (< >). Some of the parameters may be followed by an optional decimal number "n", while others may require a string "s" terminated by a ctrl-Z (shown as ↑Z on the screen).

Parameters may be selected from the following list:

Dn - Delete any characters extending beyond "n" in a given line. When copying text files, COPY counts the number of characters copied after a carriage-return (ie. the length of a line of text). Upon receipt of the nth character in the line, COPY will ignore all characters until another carriage-return is received. This feature is useful when dumping text files to narrow printers, and when only a few lines are longer than the printer width.

E - Echo all copied characters to the console device (ie the screen)

F - Ignore any form-feed/clear-screen characters (OCH) received

L - Translate any upper-case (A to Z) characters to lower-case.

N - Add a line number followed by a ":" to each line of a text file transferred, starting at 1 and incrementing by 1. Leading zeroes are suppressed unless N2 is given, in which case leading zeros are included, together with a "TAB" character (09H) following the number.

O - Object code transfer; Treat any end-of-file (EOF) characters as if they were normal received characters. This only applies to devices- the EOF code would normally be the only way to indicate termination of a transfer from a peripheral device. With this option, however, object code may be received from a device, and termination effected by pressing CTRL-S on the keyboard.

Pn - Include form-feed characters (0CH) at every n lines, with one at the start. If n is excluded or set to 1, a default value of 60 will be assumed. If the F parameter is also used, the old form-feeds are removed and new ones added at the appropriate places.

Qs↑Z - "Quit" - Terminate copying after the string s has been received. The string s will be included in the copied data.

R - Read system files System files would normally be ignored on file copying, but will be included if this parameter is given (see Page 36 - System Files).

Ss↑Z - Ignore all input until string s is received, then start copying. The string s will be included in the copied data. The Q and S parameters may be used to "abstract" portions from a file. Note the use of the ↑Z (CTRL-Z) code as the delimiter in both cases.

Tn - Expand "TAB" characters (09H) with spaces to every nth column during the copy. (Normally "TAB" characters would be transferred as received, so for example, a printer could be set up with its own tab points rather than expanding with spaces).

U - Translate any lower-case (a to z) characters into upper-case.

V - Verify that files have been copied correctly, by re-reading after the write operation. (This only applies if the output is to a file).

W - Write over "locked" files without prompting the user. Normally, COPY will prompt the user before attempting to over-write a file that has been "locked".

Z - Zeros the parity bit (bit 7) of each character received.

Examples:

```
COPY XMPL.ASM <SSUB1:↑ZQJP  LABL2↑Z >TO SILLY.LIB
```

This copies from the file XMPL.ASM to the file SILLY.LIB on the same (default) drive, the section starting with string "SUB1:" up to the string "JP LABL2"

```
COPY TEST.ASC <NT8U> TO LST:
```

This copies the file TEST.ASC on drive 0 to the current "list" device, with each line numbered, tabs expanded with spaces to every 8th column, and lower-case letters converted to upper-case.

```
COPY *.* TO 1: <V>
```

This copies all files from the current default drive onto drive 1, with verifcation of all files after copying. The V can also appear on the "input" side, with the same effect.

INTERFACING TRANSIENT PROGRAMS

This information is included for the experienced user who has some knowledge of Z80 Machine Code programming. Inexperienced users will need to refer to other publications relating to Machine Code Programming before a full understanding of this section can be grasped.

To run transient programs will normally require the use of DOS facilites such as input/output, file creation, opening, closing, reading and writing. All of these and other facilities are handled by a set of DOS "functions", accessed by placing the "function number" into the Z80 C register, and performing an RST 28H instruction. For compatibility with Digital Research CP/M operating system, DOS functions may also be called by means of a CALL to location 0005H in memory.

In addition to supplying a function number, it is usually necessary to pass parameters into the DOS function, and to obtain results. The following conditions then usually apply.

E register - 8 bit value passed to the function.
DE register - 16 bit value passed to the function.
A register - 8 bit value returned by the function.
HL register - 16 bit value returned by the function.

DOS Functions

Function 0 Warm Boot
Parameters: C: 00H
Returned Value: NIL

Function 1 Console Input
Parameters: C: 01H
Returned Value: A: ASCII character input

```
Function 2      Console Output
   Parameters:      C: 02H  E: ASCII character to output
   Returned Value: NIL

Function 3      Auxiliary Output
   Parameters:      C: 03H  A: ASCII character input

Function 4      Auxiliary Output
   Parameters:      C: 04H  E: ASCII character to output
   Returned Value: NIL

Function 5      Printer Output
   Parameters:      C: 05H  E: ASCII character to output
   Returned Value: NIL

Function 6      Direct I/O
   Parameters:      C: 06H  E: 0FEH if status required,
                               0FFH if input required,
                               else ASCII character to
                               output
   Returned Value: A: ASCII character or Status value

Function 7      Get I/O vector
   Parameters:      C: 07H
   Returned Value: A: I/O vector setting

Function 8      Set I/O vector
   Parameters:      C: 08H  E: I/O vector setting
   Returned Value: NIL

Function 9      Display message
   Parameters:      C: 09H  DE: Address of Start of
                                Message

   Returned Value: NIL

Function 10     Console Line Input
   Parameters:      C: 0AH  DE: Input Buffer Address
   Returned Value: Input characters in buffer
```

```
Function 11     Console Status
   Parameters:        C: 0BH
   Returned Value: A: Status value

Function 12     Return Version No.
   Parameters:        C: 0CH
   Returned Value: HL: Version No.

Function 13     Reset Disc System
   Parameters:        C: 0DH
   Returned Value: NIL

Function 14     Select Drive
   Parameters:        C: 0EH  E: Drive No.
   Returned Value: NIL

Function 15     Open File
   Parameters:        C: 0FH  DE: FDESC Address
   Returned Value: A: Directory Code

Function 16     Close File
   Parameters:        C: 10H  DE: FDESC Address
   Returned Value: A: Directory Code

Function 17     Get lst Directory Entry
   Parameters:        C: 11H  DE: FDESC Address
   Returned Value: A: Directory Code

Function 18     Get next Directory Entry
   Parameters:        C: 12H
   Returned Value: A: Directory Code

Function 19     Erase File
   Parameters:        C: 13H  DE: FDESC Address
   Returned Value: A: Directory Code
```

```
Function 20     Read Sequential
   Parameters:      C: 14H  DE: FDESC Address
   Returned Value: A: Directory Code

Function 21     Write Sequential
   Parameters:      C: 15H  DE: FDESC Address
   Returned Value: A: Directory Code

Function 22     Create File
   Parameters:      C: 16H  DE: FDESC Address
   Returned Value: A: Directory Code

Function 23     Rename File
   Parameters:      C: 17H  DE: FDESC Address
   Returned Value: A: Directory Code

Function 24     Get Drive Log Vector
   Parameters:      C: 18H
   Returned Value: HL: Log Vector

Function 25     Get Current Drive
   Parameters:      C: 19H
   Returned Value: A: Current Drive No.

Function 26     Set Buffer Address
   Parameters:      C: 1AH  DE: FDESC Address
   Returned Value: NIL

Function 27     Get Allocation Vector
   Parameters:      C: 1BH
   Returned Value: HL: Allocation Vector Address

Function 28     Lock Drive
   Parameters:      C: 1CH
   Returned Value: NIL
```

```
Function 29     Get Drive Lock Vector
   Parameters:       C: 1DH
   Returned Value: HL: Drive Lock Vector

Function 30     Set File Attributes
   Parameters:       C: 1EH  DE: FDESC Address
   Returned Value: A: Directory Code

Function 31     Get Drive Parameter Address
   Parameters:       C: 1FH
   Returned Value: HL: Drive Parameter Block Address

Function 32     Set/Get User Code
   Parameters:       C: 20H  E: OFFH if getting USER code
                                CODE if setting USER code
   Returned Value: A: Current User code

Function 33     Read Random
   Parameters:       C: 21H  DE: FDESC Address
   Returned Value: A: Error Code

Function 34     Write Random
   Parameters:       C: 22H  DE: FDESC Address
   Returned Value: A: Error Code

Function 35     Compute File Size
   Parameters:       C: 23H  DE: FDESC Address
   Returned Value: A: Random RECORD set

Function 36     Set Random Record No.
   Parameters:       C: 24H  DE: FDESC Address
   Returned Value: A: Random RECORD set

Function 37     Reset Drive
   Parameters:       C: 25H  DE: Drive Vector
   Returned Value: A: 0
```

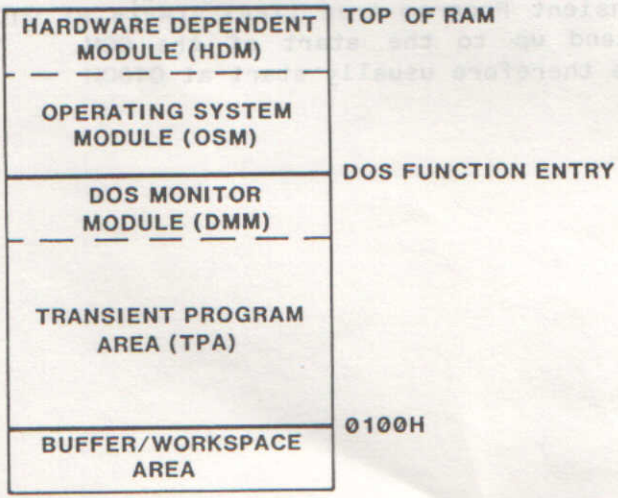
Function 38 Return to MOS
Parameters: C: 26H
Returned Value: NIL

Function 39 Set Password
Parameters C: 27H DE: Address of start of password
Returned Value: A: Error Code

Function 40 Write Random (zero fill)
Parameters: C: 28H DE: FDESC Address
Returned Value: A: Error Code

THE DOS MODULES

The DOS has been designed as a portable disc operating system and consists of three distinct modules incorporated into the memory map as shown below.



The Operating System Module (OSM) and DOS Monitor Module (DMM) are both hardware-independant, and require no changes in order to run on other Z80 systems (other than relocation, when required).

In order to operate with the facilities of the system, the OSM interfaces to the Hardware-Dependant Module (HDM) via a fixed table of jump vectors, which access the required routines to allow console I/O, printer output, serial port I/O, Disc sector I/O and so on.

The OSM and HDM are permanently resident while DOS is running. The DMM is a separate program in its own right, which loads directly below the OSM in the memory map, but which may be overwritten by transient programs or data areas used by transient programs. The normal exit from a transient program is via a jump to location 0000H, which causes a "warm boot", ie. reloading and execution of the DMM.

The Transient Program Area (TPA) starts at 0100H, and may extend up to the start of the OSM. Transient programs therefore usually start at 0100H

ISBN: 1-85086-006-8

CODE NO: 79-0883-0