INTRODUCTION

This handbook 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 MICRO EINSTEIN. Section B deals with the Disc Operating System (DOS) outlining the conventions used, and the facilities available to the user when working with MICRO 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 handbook to cover the complete operating system of MICRO 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 MICRO EINSTEIN. Xtal DOS, (as it is known) loads into the top of RAM (The TATUNG/Xtal BASIC 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 LOGO - type **bye,** (which will get you into DOS.)

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

d) 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.

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

f) 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.

This will put you in the "Command Level".

Each command consists of a single letter (upper or lower case) which may be followed by 1 to 4 hexadecimal numbers comprising up to 4 hex digits. The exception is the "H". command which requires a single decimal number.

Revision/Amendments

Note 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

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:-

```
AAAA        BBBB        CC

SUM      DIFFERENCE  OFFSET
```

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

COPY

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

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

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

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 0FA0 0FFF 23

This will fill the block of memory from location 0FA0 to 0FFF 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

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 B0FF

This instructs execution to transfer to location B002 and carry out the routines from there onwards. When location B0FF is reached a break point occurs and the Z80 registers will be displayed with their current contents.

HEXADECIMAL

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

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 (i.e. 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
0B3O 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. Use a full stop, (.), to terminate the routine.

READ

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.

WRITE

Syntax: W xxxx yyyy sstt d

Purpose: This command will write a block of memory, from address "xxxx", to address "yyyy", to a disk drive, starting at sector "ss", track "tt", on drive no. "d".

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.

TABULATE

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 (i.e. ASCII value for character C).

COLD START

Syntax: X

Purpose: This command will cause a "cold start" transfer to the current program from MOS. (Always provided the execution vector has been passed into MOS by the program loaded, otherwise performs a G 0100)

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".

There are a number of useful MOS CALL Routines which enable the enthusiast to access some functions which he may otherwise have had to write. They are detailed later.

WARM START

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

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

		S	- Sign Flag
		Z	- Zero Flag
		-	- Not Used
	F	H	- Half Carry Flag
	Register	-	- 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

Syntax: B xy wwzz

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 handbook 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.

MOS MONITOR SYSTEM CODES

SCREEN EDITING

When working in the MOS monitor, 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.

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); (0A)H

   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.(0C)H

   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 (0D)H

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

4. **CTRL-A** Screen Dump to Printer. (01)H
Will cause a transfer of the screen display contents to a printer (i.e. make a hard copy on paper)

5. **CTRL-H** Backspace (BS). (08)H

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). (09)H

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** Bell. (0A)H

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

8. **CTRL-K** Vertical Tabulation. (VT). (0B)H

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. (14)H

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

10. **CTRL-Q** Cursor ON. (11)H

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

11. **CTRL-R** Printer ON. (12)H

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. (13)H

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

13. **CTRL-** Cursor Home. (1E)H

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

14. **CTRL-X** Erase Whole Line. (18)H

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. (15)H

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

16. **CTRL-V** Erase to End of Screen. (16)H

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

17. **CTRL-Y** Delete Character. (19)H
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. (06)H
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** (0E)H

This clears the screen and sets 40 column display.

20. **CTRL-O** (0F)H

This clears the screen and sets 32 column display

21. **CTRL-P** (10)H

This clears the screen and sets the optional 80 col. card display when fitted.

22. **CTRL-Z** (1A)H
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.

23. **CTRL** Cursor Addressing Mode. (1D)H.

This sets up the cursor address. The syntax is &1D,X,Y where X and Y are the location on the screen. Y is in the value 0-23. the X value depends upon the display mode selected.

Useful Machine Routines

A machine call, MCAL, is a routine in ROM which is used by both DOS and any applications program to do the most frequently used operations. They cover things such as getting a character from the keyboard, outputting a character or word to the screen. Not all programs use them, but the home hobbyist may find them very useful when learning to program in machine code. (The ROM is "Sideways mapped" between 00 and &4000.)

The following is a list of legitimate machine calls to MOS 1.1, MOS 1.2 and ,MOS 1.21.

The calls (MCAL's) are executed on a RST08 (Restart 8 instruction) followed by the function number (given in Hex). Return is to the address after the function number.e.g.:-

e.g.

```
ADDRESS     OBJECT     MNEMONIC

100         CF          RST08      (ROM CALL)
101         9C         data 9CH    (FUNCTION NUMBER)
102         FF          RST38      (NEXT INSTRUCTION)
```

On G100, the above will perform MCALL 9C (ZKEYIN) which will wait for a key to be pressed and return the value in the A register.

(On return, the Restart 38 will perform a break)

Abbreviations used

CTC - Counter Timer Circuit
VRAM - Video RAM
VDP - Video Display Processor
PSG - Programmable Sound Generator
FDC - Floppy Disc Controller
MCAL - Machine Call

EINSTEIN MOS MCAL INFORMATION
MOS 1.1 and 1.2 Series

Function Number	Source Label	Description
80	ARITH	Performs as 'A' (ARITHMETIC) from MOS Values Passed xxxx in HL pair yyyy in DE pair
81	BAUD	Performs as 'B' (BAUD) from MOS Values Passed x - Receive rate - upper nibble of L register y - Transmit rate - lower nibble of L Register ww - Mode Byte - D register zz - Command Byte - E register If DE is zero, the mode and command bytes will remain unchanged. For the correct receive and transmit rates, the baud rate factor X16 must be used.
82	COPY	Performs as 'C' (COPY) from MOS Values Passed xxxx - Start - in HL pair yyyy - Finish - in DE pair zzzz - Destination - in BC pair (See manual
83	DECIML	Performs as 'D' (DECIMAL) from MOS Values Passed xxxx in HL pair

Function Number	Source Label	Description
84	EXEC	Performs as 'E' (EXECUTE) from MOS Values Passed xxxx - Break point - in HL pair
85	MFILL	Performs as 'F' (FILL) from MOS Values Passed xxxx - Start - HL pair yyyy - Finish - DE pair zz - Value - C register
86	GOTO	Performs as 'G' (GOTO) from MOS Values Passed xxxx - execution address - HL pair yyyy - break point - DE pair If DE is zero, no break point is set
87	HEX	Performs as 'H' (HEXADECIMAL) from MOS Values Passed Pointer to text (in RAM) - DE pair The decimal number is held at (DE) and terminated with 00. The hexadecimal number is returned in the HL pair in addition to being displayed.
8C	MODIFY	Performs as 'M (MODIFY) from MOS Values Passed xxxx - address to modify from - HL pair

Function Number	Source Label	Description
91	RDBLOK	Performs as 'R' (READ) from MOS Values Passed Pointer to text (in RAM) - DE pair The text takes the format as shown in the manual. (See ZRBLK function no. A4)
93	TBLATE	Performs as 'T' (TABULATE) from MOS Values Passed xxxx - Start address - HL pair yyyy - Finish address - DE pair zz - no. of collums - C register If C is passed as zero, zz will default to 8 colums.
96	WRBLOK	Performs as 'W' (WRITE) from MOS Values Passed Pointer to text (in RAM) - DE pair The text at (DE) takes the format as shown in the manual (See ZWBLK function no. A5)
97	COLD	Performs as 'X' (COLD START) from Mos
98	WARM	Performs as 'Y' (WARM START) from MOS
99	REGSTR	Performs as 'Z' (REGISTER EXAMINE) from MOS Values Passed x - registers to be displayed - L register (0, 1 or 2) Note: This does not display the current register contents but the register contents at the last RST 38H (FFH encountered).

Function Number	Source Label	Description
9A	ZINIT	Re-entry point to MOS
9B	ZRSCAN	- Repeat key scan This will return the value of any key pressed, in the A register. 00 is returned if no key is pressed. 00 can also be returned if the keyboard poll rate (polled with this MCAL) is greater than the key repeat speed. This repeat speed can be altered in scratch pad location FB43H Note: This works as the KBD command in XBAS.
9C	ZKEYIN	- Input key This will return the value of any key pressed, in the A register. Unlike MCAL 9B, this will wait for a key to be pressed. Note: This is similar to the INCH command in XBAS.
9D	ZGETLN	- Get Text from keyboard This will enter a line of text from the keyboard into RAM from the address held in the DE pair. The text is displayed on the screen on pressing the keys and the line is terminated with an ENTER.
9E	ZOUTC	- Character Output. Outputs a character to screen held in the A register.

Function Number	Source Label	Description
9F	ZPOUT	Outputs a character to the parallel printer held in the A register.
A0	ZSLOUT	- Serial Output Outputs a character to the serial port from the A register.
A1	ZSRLIN	- Serial Input Read a byte from the serial port and returns the value in the A register.
A2	ZRSECT	- Sector Read Reads a sector from the disc into the sector buffer (A sector is 200H bytes) The following are set up in the scratch pad LOCATION LABEL FB50H HSTDSC - Disc drive (0-3) FB51H HSTTRK - Track (0-27H) FB52H HSTSEC - Sector (0-9) FB53H HSTDMA - Sector buffer address (normally FE00H)
A3	ZWSECT	- Sector Write Writes a sector from the disc into the sector buffer (See MCAL A2)

Function Number	Source Label	Description
A4	ZRBLK	Read Block Read a block of data from the disc Values Passed Drive no. (0-3) in A register Start address in HL pair Finish address in DE pair Sector (0-9) in B register Track (0-27H) in C register Memory is filled to the next complete sector (200H bytes) i.e. if the start and finish address is specified as 6000H and 6001H respectively, 6000H to 6200H will be read from the disc.
A5	SWBLK	Write Block Writes a block of data to the disc. The values passed are the same as for MCAL A4 and agains is written to the next complete sector (200H bytes)
A6	ZCRLF	Outputs a CR (ODH) and an LF 90AH)
A7	ZCRLFZ	Outputs a CR and LF if the cursor is not at column zero.
A8	ZSPACE	Outputs 1 space.
A9	ZPR4HX	Outputs 4 hex digits held in the HL pair. e.g. if HL = 1234H, 1234 is output
AA	ZP2HXZ	Outputs two hex digits held in the A register followed by a space.

Function Number	Source Label	Description
AB	ZPR2HX	Outputs two hex digits held in the A register (as MCAL AA with no space output)
AC	ZFC4HX	Get a hex number (up to 4 digits) from text into the HL pair. DE points to the text (in RAM). The number is terminated on a non hex character.
AD	ZFCZHX	Get a hex number (up to 2 digits) from text into the A register. DE points to the text. The number is terminated with a non hex character.
AE	ZDCMD	Outputs a command to the FDC. The command type is passed in the A register. On return A is set to 00 unless the afdc is not executing the command then A is set to FFH.
AF	ZHMDSC	Takes the drive head to track 00 The drive no. is passed in the A register.
B0	ZIGBLK	Returns a value in the A register from RAM pointed to by the DE pair if DE 7FFFH, or from ROM if DE 8000H. Note: Commas and spaces are ignored (i.e. the first non comma/non zero character is displayed).

Function Number	Source Label	Description
B1	ZRDMEM	Returns a value in the A register from RAM pointed to by the HL pair.
B2	ZRCPYU	Performs an LDIR instruction (switches ROM out first).
B3	ZRCPYD	Performs an LDDR instruction (switches ROM out first)
B4	ZMOUT	Outputs a value held in the B register to the PSG port no. held in the C register
B5	ZKSCAN	Returns the value in the A register of any key pressed. 00 is returned if no key is pressed. This is similar to MCAL 9B (ZRSCAN) except that it is unaffected by the key repeat speed. That is for each MCAL execution, a value is returned.
BC	ZZTIME	Set up CTC channels 0 and 3 to generate 1 second interrupts for clock.
BD	ZFDRST	Resets the FDC AFTER AN ERROR. Also resets the PSG (using MCAL C0 (ZPINIT))

Function Number	Source Label	Description
BE	ZSYSRS	This performs the following:- 1. Clears the screen to 40 colums 2. Resets all characters 3. Removes sprites 4. Resets the FDC and PSG 5. Masks the keyboard, fire and ADC interrupts
BF	ZLOGO	Outputs '***EINSTEIN***' logo
C0	ZPINIT	Sets PSG register 7 to 7FH and all other registers to 00
C1	ZSREG	Sends an address held in the BC pair to the VDP. Data can then be output to VRAM from port 8. Subsequent data bytes sent will be loaded into subsequent VRAM locations. Note: A delay of 8 is necessary between any VRAM reads or writes (e.g. PUSH/POP) Note: When ROM is switched in, RST 20H will execute this MCAL.
C2	ZVRIN	Returns a value in the A register from the VRAM address pointed to by the BC pair.
C3	ZVROUT	Writes data held in the A register to the VRAM address held in the BC pair.
CE	ZIMULT	Multiplys the contents of the DE pair and the contents of the BC pair and returns the value in DEHL. (The DE pair is the most significant)

Function Number	Source Label	Description
CF	ZPRM	Outputs a message to the screen. The data follows the C data byte and must be a character in the range 0 to 7FH. The message is terminated by adding 80H to the last character in the message.
D0	ZVOUT	Outputs a character from the A register to the current cursor position without incrementing the cursor position. This can be useful to prevent scrolling, linefeeds etc.
D1	ZSCURS	Returns VRAM addresses relating to the current cursor positions. The ASCII text map address is returned in the BC pair (will be in the range 3C00H to 3FBFH). The table pattern generator address (first byte) is returned in the DE pair (will be in the range 0000 to 17FFH). The start of the sprite patterntext pattern table is returned in the HL pair (normally 1800H)

Function Number	Source Label	Description
D2	ZROM	Starts execution from address 4004H in the 2nd ROM. All registers can be used to pass and return values. Return is achieved by a RET instruction. 2nd ROM protocol If memory location 4000H (Rom Detection Byte) is 00, (=Auto-Execution) on 'Power Up' or Reset, after printing 'Einstein' logo and before booting any disk present, then execution will start from 4001H. An MCAL D2 will execute from 4004H.
		ROM 1.2 series only.
D3	ZIN80	Initialises the 80 column card as follows:- 1). Checks if the 80 column card is present (return if not). 2). Programs the 6845 registers (programmed for 525 or 625 line operation according to 80 column card internal switch. 3). Selects 40 column or 80 columns according to 80 column card internal switch. This MCAL is performed on power up/reset.
D4	ZIN80	Programs the 80 column card status line (25th row). DE points to the text in RAM (80 bytes) and must be greater than 7FFFH.

NOTE: A clear screen can be performed on the 80 column screen leaving the status line intact with output codes IEH (HOME) and 16H (CLEAR TO END OF SCREEN)

GRAPHICS ROUTINES STORED IN ROM

An understanding of these routines can be gained from using the graphics commands in XBAS. e.g. DRAW, FILL, ELLIPSE, POLYGON, PLOT.

Function Number	Source Label	Description
C4	ZPLOT	PLOTs or UNPLOTs a pixel. A = 1 for PLOT A = 0 for UNPLOT IX holds the x coordinate IY holds the y coordinate

C5 ZPLTXY Plots a point according to the line type (see below)
IX holds the x coordinate
IY holds the y coordinate

Line Type
Four scratch pad values contain information as to the line type to be drawn e.g.

these are:-

SCRATCH LOCATIONS
FBABH DOTON - Length to end of first line
FBA9H DOTOFF - Length to end of first space
FBAAH DOTON2 - Length to end of second line
FBABH DOTOF2 - Length to end of second space

line type

DOTON

DOTOFF

DOTON2

DOTOF2

Normally these 4 values are in ascending order.
For a continuous line, DOT ON is set to FFH and the other to zero.
For a continuous unplot, DOTOFF is set to FFH and the other to zero.
Note:If all lines type bytes are zero, the system will hang up.

Function Number	Source Label	Description
C6	ZPOINT	Returns the status of any pixel in the A register 1 = foreground 0 = background (the zero flag is also set) FFH = off screen BC contains the x coordinate DE contains the y coordinate The VRAM address of the pixel is returned in the BC pair.
C7	ZPNTXY	Returns the status of any pixel in the A register. This is identical to MCAL C6 except IX contains the x coordinate and IY contains the 7 coordinate. The VRAM address of the pixel is returned in the BC pair.
C8	ZDRWTO	This will draw a line from the coordinates held in the IX and IY registers to values in scratch pad locations FB96H (X1) and FB98H (Y1). Each is a two byte number. The type of line drawn is determined by the line type values (see MCAL C5)

Function Number	Source Label	Description
C9	ZPOLYG	This draws a polygon (or ellipse) The polygon centre coordinates are held in scratch pad locations FB9EG (CX) and FBA0H (CY), each is a two byte number. The horizontal and vertical radii are held in locations FBA2H (RADX-2BYTE) and FBA4H (RADY-2BYTE). The number of sides on the polygon is determined by a two byte number in scratch pad location FBA6H (CINC). A value of 4 will give a circle A value of 80H will give an octagon A value of 100H will give a rectangle etc. The start angle is passed in the DE pair, the finish angle is passed in the BC pair, each is in the range 0 to 1024. The lines drawn to the polygon centre are selected by setting the carry flag. The type of line drawn is determined by the line type values of (see MCAL C5).

CA ZORGCO Adds the x coordinate Origin value held in scratch location FB9AH (ORGX - 2Byte) to the contents of BC and returns the result in the BC pair and adds the y coordinate Origin value held in scratch location FB9CH (ORGY - 2Byte) to the contents of DE and returns the result in the DE pair.
This MCAL is of little use and is called from within other graphics MCALS.
The values ORGX and ORGY are normally zero but when altered will cause all graphics output to be offset by that value. This is similar to the ORIGIN command in XBAS.

CB ZCALAD Returns the VRAM address in the BC pair for coordinates x and y passed in the IX and IY registers respectively. The pixel position within the 8 pixel row is returned in the E register, counting from the left hand pixel (0-7).

CC ZSETCL Writes data held in the A register to the pattern generator table address passed in the BC pair (0 to 17FFH) and sets the corresponding byte in the pattern colour table (2000H to 37FFH) to the contents of scratch locations FB39H (GCOLR)

Note All MCAL in MOS 1.1 which output information to the screen will output information to the 80 col. card, (where present and selected). The exceptions are MCAL (ZVROUT) and graphics MCAL'S, (C4 to CD), which output to VRAM.

MCAL C2,(ZVRIN), works differently on MOS 1.2 series. When the 80 col. card is present and selected, MCAL C2 reads from the 80 col. card RAM and **not** VRAM. **BC** is still passed containing the address and the data returned in the **A** register. The 80 col. card RAM is addressed as follows;-

M.S. Byte C - 40 to 47 H
L.S. Byte B - 0 to FF H

(The type number of the 80 col. mono card is TK02).

CD ZFILL Fills an area on screen surrounding coordinates passed in the IX and IY registers (x and y coordinates respectively).
If the fill is in foreground (i.e. the point at x, y is not set) then scratch location FBADH (FILLMOD) must be set to FFH.
If the fill is in background (i.e. the point at x, y is et) then scratch location FBADH (FILLMOD) must be set to zero.
MCAL XPNTXY (C7) can be used to find the fill type needed.

SECTION B

DISC OPERATING SYSTEM

DISC OPERATING SYSTEM (DOS)

The Disc Operating System (abbreviated to DOS) for MICRO 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 many 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 0 or 1. 0 is the internal drive, 1 is the optional external drive. 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 Logo - type **bye** ENTER

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

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

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

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.

Editing :

At the "command level" in the DOS monitor, the "Line Editor" only is available. Correction of a mis-typed file name, or command is therefore limited to using the DEL or cursor left keys and re-typing the line. There are no facilities "Insert" function.

Similarities to CP/M:

Tatung/Xtal DOS is very similar to the CP/M operating system. Brief details of equivalence are given below.

CP/M Function Name	Xtal Dos equivalent.
ERA	ERA
DIR	DIR
REN	REN
SAVE	SAVE

(note: while the functions are very similar, the syntax is different. See the "SAVE" command in the DOS section).

TYPE	DISP
LOAD	LOAD

(note: not quite the same, but performs a similar function).

SYSGEN	(This is performed in FORMAT).
PIP	(" " " " COPY).
STAT.	(limited information displayed).
	(in DIR).

CP/M is a trademark of Digital Research.

FILE NAMING CONVENTIONS

Filenames

The standard file name convention is as follows:-

drive: filename . extension

where:-

drive - is the disc drive in use

file name- 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" extensions to denote specific types of file.

1. **.XBS** - This is an Xtal BASIC source file (i.e. 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 an 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.

3. Other extentions are often assigned to files created by a program. For example, the program "Hangman" on the System Disc calls from within itself one of two files, (DICTIONH, and DICTIONE) both of which have the extention .DAT, since these Data files are read by the program. In the same way, a Data Base program may store as .DAT any number of files, so the index might be sorted .IND, or a Word Processor might store a text as .TXT, .UFT., or DOC.

4. Matching File Names - In some disc applications it is necessary to specify more than one file, for example when specifying the files for a DIRECTORY list. If one of the characters in a file name is replaced by a ?, then it will match any character in that position in the file name found.

   EXAMPLE:

   X?Z.ASC - Matches XYZ.ASC,XAZ.ASC,X9Z.ASC,etc.
   ?X.D?T - Matches AX.DAT,BX.D7T,4X.DZT,etc

5. If an * is inserted in place of a character then it will match all the characters at and after that particular position in the file name or file type in which it appears.

EXAMPLE:

*.XBS - matches any XBS file.
* - also matches any XBS file (XBS being the default)
. - matches any file, and is the same as ????????.???
PROG*.ASC matches PROG1.ASC,PROG10,ASC,PROGABC.
ASC,etc.

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
1:WXYZ.ASC     - refers to the file WXYZ.ASC on drive 1
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.

CHARACTER SET

TATUNG/Xtal BASIC 4 character set which consists of:-

i) alphabetic characters
ii) numeric characters
iii) graphics characters
iv) selection of punctuation and other symbols normally found on a typewriter keyboard.

The normal keyboard characters are listed below with their corresponding ASCII code. Graphics characters will be found in the full table given in Appendix D of the Introduction Manual.

ASCII CODE	CHAR.	HEX	ASCII CODE	CHAR.	HEX	ASCII CODE	CHAR.	HEX
32	SP	20	52	4	34	72	H	48
33	!	21	53	5	35	73	I	49
34	"	22	54	6	36	74	J	4A
35	#	23	55	7	37	75	K	4B
36	$	24	56	8	38	76	L	4C
37	%	25	57	9	39	77	M	4D
38	&	26	58	:	3A	78	N	4E
39	'	27	59	;	3B	79	O	4F
40	(	28	60		3C	80	P	50
41	)	29	61	=	3D	81	Q	51
42	*	2A	62		3E	82	R	52
43	+	2B	63	?	3F	83	S	53
44	,	2C	64	@	40	84	T	54
45	-	2D	65	A	41	85	U	55
46	.	2E	66	B	42	86	V	56
47	/	2F	67	C	43	87	W	57
48	0	30	68	D	44	88	X	58
49	1	31	69	E	45	89	Y	59
50	2	32	70	F	46	90	Z	5A
51	3	33	71	G	47	91		5B

ASCII CODE	CHAR.	HEX	ASCII CODE	CHAR.	HEX	ASCII CODE	CHAR.	HEX
92	½	5C	104	h	68	116	t	74
93		5D	105	i	69	117	u	75
94		5E	106	j	6A	118	v	76
95		5F	107	k	6B	119	w	77
96	£	60	108	l	6C	120	x	78
97	a	61	109	m	6D	121	y	79
98	b	62	110	n	6E	122	z	7A
99	c	63	111	o	6F	123	¼	7B
100	d	64	112	p	70	124		7C
101	e	65	113	q	71	125	¾	7D
102	f	66	114	r	72	126	÷	7E
103	g	67	115	s	73			

DOS COMMANDS

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. Note the "locked" signal on XBAS.COM and test prog.ASC

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, it will be returned as a 0k size preceded by 'No File'.

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 (i.e. lists it).

Example:

```
DISP PET.ASC
```

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

Note: Using DISP will show some rather surprising results onto the screen. Don't forget that most languages store their data as "tokens", (a sort of short-hand), so you won't make a lot of sense out of an .XBS file, for example. Many .COM files have incorporated into them some form of "Anti-DISP" device, so you'll see even less on the screen. Some machine code (.OBJ) files contain no printable characters at all, but the DISP function is very useful for inspecting .TXT, .UFT, or .ASC files, rather than calling them from their own program.

Some discs have a file called "READ.ME" which is often a late addition to a supplied documentation. It is designed to be displayed. If you want a hard copy and have a printer, use CTRL-A to dump the displayed file, page by page, to the printer. DO NOT try it if you don't have a printer - you may have to do a "cold start" if the system "hangs up".

DRIVE (Disc Drive)

Syntax: DRIVE d

d is specified as a number 0 or 1 depending on the drive units available within the system.

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

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

Example:

DRIVE 1

This selects drive 1 as the default drive.

Note: This will allow, for example, of the inspection of a file on one drive while the main program, or utilities, are on another.

ERA (ERASE)

Syntax: ERA filename.ext.

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 0:
Unlock (Y/N)?
```

The user must then type in one of 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.

Note: If, for some reason, you have left a program (say Backup or Copy) and you want to return to it for another operation, (say formatting, or back-up of a file), then GO is the right command. If, however, you jumped out of a language, and you want to keep the variables, then type MOS, and then Y, or G103 which will take you to the start of the program, but "warm started".(This is mainly applicable to a language, like BASIC.)

LOAD

Syntax: LOAD filename.ext

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. i.e. 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.ext

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.

Note: The lock marker is a small star *. When looking at the list of files under the DIR command, the list would have the star at the beginning of the filename:-

e.g.

*MAIL.XBS - would indicate that the file MAIL.XBS is locked.

A system file is not displayed on the DIR listing.

UNLOCK

Syntax: UNLOCK filename.ext

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).

NOTE: Unlock will also expose 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 .XBS (BASIC) to .BAK (say backup file)

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.ext

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
```

Note: See also LOAD.

APPENDIX A

UTILITIES

There are three utilities which will allow the user to prepare discs and transfer files.

a) **FORMAT**
No disc ever left the makers ready for use by a computer. There are System Tracks (the Directory, DOS, etc.,) to be "put on" before it can be used and **FORMAT** is this process. It is a good idea to Format all discs when you buy them.

b) **BACKUP**
This utility transfers the entire contents of a disc side to another. It does not re-organise the files, and includes all the redundant files you have probably forgotten you had, because it is an **exact** copy.

c) **COPY**
This is the one to move files/programs from one disc to another. (Note: with some games it is not only illegal, it is difficult as well). So you can take a backup of one of your programs to another disc.

To call these utilities you start from DOS and follow the instructions on each utility.(The files are on the master disk).

1. **Format**
From DOS type FORMAT.
You will be asked to insert a disc with the required system tracks. You are strongly advised to ensure that the 'Write Protect' tab is in the 'Protect' postion, (see Introductory section, ch.2)., before going further.

You will now see another message:-

OK -- Format Drive (0 or 1)?

Now remove the reference disc and insert the new disc to be formatted. (The destination disc).

Select 0 or 1 (0 is the internal drive), and press ENTER.

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

```
0         1         2         3
01234567890123456789012345678901234567890
```

(i.e. 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
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

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
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV
```

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 exit
```

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

1. Formatting with multiple disc drives.

As with single drive formatting, make certain that the write protect notch on the source 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 or 1)?
```

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 exit.
```

is displayed when the destination disc is formatted.

Backup

Typing **BACKUP** selects the **backup** utility.
The screen will display:-

```
DISC BACKUP UTILITY V x.x
 (c) 1984 Xtal

Key ENTER to BACKUP
 or X to exit
```

Two disc drives must be declared when using **BACKUP**, 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 exit
```

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 files from disc to disc using a single drive.

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

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.
   NOTE: If file-input and file-output are identical (i.e. 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. You only need to quote the input specification if you copy to another disc.

Examples:

COPY 0:OLD.ABC TO 0:

will lift OLD.ABC from the source disc, and tell you to:

Insert destination disc.

When you press ENTER, the file will be deposited onto the new (destination) disc with no change of name.

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

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

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.UYZ

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 a "*" 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. COPY may be terminated by CTR-C; ENTER.

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. The parameter list is used in the output specification only, immediately following the particular file or device it relates to, and is 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 (i.e. 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 (i.e. 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 (OCH) 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 - LOCK). The **R** parameter is only valid when using wild cards and should not be used when specifying individual 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  TO SILLY.LIB   SSUB1: ZQJP  LABL2 Z
```

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  TO LST:  NT8U
```

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 verification of all files after copying. The V can also appear on the "input" side, with the same effect.

DISC FORMAT

	Bytes	Value	
Track Preamble	32	4E	
	12	00	
	3	F5	
	1	FE	
	1	"	Track No.
	1	00	Side No.
	1	"	Sector No.
	1	02	
	1	F7	CRC (Converted to 2 bytes by controller)
	22	4E	
	12	00	
	3	F5	
	1	FB	Data Address Mark
	512	4E	Data
Padding	44	4E	To end to track

NOTES:

On a disk, tracks 0 & 1 are "System tracks", (usually used for DOS, etc.), but may also contain a "map" of the disc.

APPENDIX B

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 CALLS 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 1st 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: Buffer 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.

Memory Map (M1)

Hex. Address

0000	Jump Vectors/Buffer/Workspace
0100	Transient Program Area,(TPA.)
E100	DOS from Disc.
E300	Start of DOS. (DOS monitor module), (CCP)
EC00	BDOS. Operating system logic. (OSM)
FA00	BIOS. Hardware Dependant Moduel (HDM) Interface to machine
FB00	MOS Scratch pad.
FC00	Interrupt routines.
FC80	MOS stack. (down from FCFF)
FD00	Directory Entries. Disc allocation maps.
FE00	Sector buffer.
FFFF	Top of RAM.

NOTES

As may be seen from the memory map, (M1), the DOS is divided into several modules. The similarity to CP/M will be noted, and in order to maintain some concordance, the start addresses are the same. The DOS Monitor Module, (DMM), which is approximately the same as the CP/M "C.C.P" is a separate program which often resides below the OSM, (Operating System Module), but which may be over-written by a long program in the TPA, or a file, (or data), stored in that "out of the way" area.

The other modules, (O.S.M., Operating System Module), which in CP/M terms may be likened to the BDOS, and the HDM (Hardware Dependant Module), (or BIOS), are permanently resident while under DOS. In order to link the modules together, the last part of the DOS in a jump vector table which accesses the required functions.

The normal exit from a transient program is via a jump to the "Warm Boot" location 0103H. The T.P.A. starts at 0100H.

THE DOS MODULES

The DOS has been designed as a portable disc operating system and consists of three distinct modules.

Memory Map (M1)

Hex. Address

0000	Jump Vectors/Buffer/Workspace
0100	Transient Program Area,(TPA.)
E100	DOS from Disc.
E300	Start of DOS. (DOS monitor module), (CCP)
EC00	BDOS. Operating system logic. (OSM)
FA00	BIOS. Hardware Dependant Module, (HDM) Interface to machine
FB00	MOS Scratch pad.
FC00	Interrupt routines.
FC80	MOS stack. (down from FCFF)
FD00	Directory Entries. Disc allocation maps.
FE00	Sector buffer.
FFFF	Top of RAM.

NOTES

As may be seen from the memory map, (M1), the DOS is divided into several modules. The similarity to CP/M will be noted, and in order to maintain some concordance, the start addresses are the same. The DOS Monitor Module, (DMM), which is approximately the same as the CP/M "C.C.P" is a separate program which often resides below the OSM, (Operating System Module), but which may be over-written by a long program in the TPA, or a file, (or data), stored in that "out of the way" area.

The other modules, (O.S.M., Operating System Module), which in CP/M terms may be likened to the BDOS, and the HDM (Hardware Dependant Module), (or BIOS), are permanently resident while under DOS. In order to link the modules together, the last part of the DOS in a jump vector table which accesses the required functions.

The normal exit from a transient program is via a jump to the "Warm Boot" location 0103H. The T.P.A. starts at 0100H.

APPENDIX D

INDEX TO ERROR MESSAGES

ERROR MESSAGE	CODE HEX	DECIMAL	
Break	00	0	Interruption from keyboard or execution of STOP
Next	01	1	NEXT statement found without corresponding FOR
Syntax	02	2	Typing error in current line
Return	03	3	RETURN found without corresponding GOSUB
Data	04	4	No more DATA statements for READ
Qty	05	5	Number specified outside allowable range
Ovfl	06	6	Number too large
Mem Full	07	7	No more memory left
Branch	08	8	Attempt to refer to non-existent line
Range	09	9	Outside dimensions specified for array
Dimension	0A	10	DIM encountered for already dimensioned array
Division	0B	11	Divide by Zero
Stack Full	0C	12	No more stack for FILL FOR, GOSUB, or expressions
Type	0D	13	String given when number expected or vice versa
Cmd	0E	14	Reserved Word not defined in system
Str Ovfl	0F	15	String expression too long
Str Complex	10	16	String expression too complex -- split it up!
Cont	11	17	Cannot continue after error or program modification

Fn Defn	12	18	FN user function not defined by a previous DEF
Operand	13	19	Operand expected in expression
Bad Data	14	20	Disc checksum error
End of Text	15	21	End of File encountered or last block of file read.
File	16	22	FDESC not defined (or used by another file)
Drive Select	17	23	Drive selected not available in system
File Type	18	24	File of incorrect type

DISC ERRORS

No File	19	25	File not found
File Exists	1A	26	REnaming existing file
File Locked	1B	27	File has been locked ("Read Only")
Disc Locked	1C	28	Disc is in "Read Only" mode
Disc Seek	1D	29	Attempt to seek beyond end of disc
Disc Full	1E	30	No space for file contents
Dir Full	1F	31	Too many files in Directory
Shape Defn	20	32	Wrong number or character type in definition string
Key Defn	21	33	Function key string is too long
Write Protect	22	34	Save to write protected disc

APPENDIX E

COLOUR NUMBERS in Tatung Xtal Basic

D	Colour	HEX
0	Transparent	0
1	Black	1
2	Medium Green	2
3	Light Green	3
4	Dark Blue	4
5	Light Blue	5
6	Dark Red	6
7	Cyan	7
8	Medium Red	8
9	Light Red	9
10	Dark Yellow (Gold)	A
11	Light Yellow	B
12	Dark Green	C
13	Magenta	D
14	Grey	E
15	White	F

APPENDIX F

1) Analogue 1/Analogue 2 7-Pin Din Connector M014 & M015

PIN NO.	ANALOGUE - 1 CONNECTOR M014	ANALOGUE - 2 CONNECTOR M015
1	Channel 0	Channel 2
2	Signal Ground	Signal Ground
3	Channel 1	Channel 3
4	Fire - 1	Fire - 2
5	Vref	Vref
6	Ov	Ov
7	+5v	+5v

2) RS232-C 5-Pin Din Connector M013

PIN NO.	SIGNAL	DIRECTION
1	Ov	- -
2	CTS	To Computer
3	TxD	To DCE
4	RTS	To DCE
5	RxD	To Computer

Key-: DCE - Data Communication equipment (e.g. MODEM)
TxD - Transmit Data
RxD - Receive Data
CTS - Clear to Send
RTS - Request to Send

ANALOGUE RS232

3) User Input/output IDC Connector M002

PIN NO.	SIGNAL
1	5v
2	DO
3	Ov
4	Dl
5	RDY
6	D2
7	Ov
8	D3
9	Ov
10	D4
11	STB
12	D5
13	Ov
14	D6
15	5v
16	D7

4) Printer Output IDC Connector M001

PIN NO.	SIGNAL	PIN NO.	SIGNAL
1	STROBE	18	Ov
2	Ov	19	ACK
3	Dl	20	Ov
4	Ov	21	BUSY
5	D2	22	Ov
6	Ov	23	PE
7	D3	24	Ov
8	Ov	25	N/C
9	D4	26	N/C
10	Ov	27	N/C
11	D5	28	ERROR
12	Ov	29	N/C
13	D6	30	N/C
14	Ov	31	Ov
15	D7	32	N/C
16	Ov	33	Ov
17	D8	34	N/C

KEY:- D - Data Bit
ACK - Acknowledge
PE - Paper End
N/C - No Connection

6) Tatung Pipe 60-Way IDC Connector M003

PIN NO.	SIGNAL	PIN NO.	SIGNAL	PIN NO.	SIGNAL
1	+5V	21	A12	41	OV
2	D7	22	A11	42	WR
3	+5V	23	A10	43	OV
4	D6	24	A9	44	RD
5	OV	25	A8	45	OV
6	D5	26	A7	46	IORQ
7	OV	27	A6	47	OV
8	D4	28	A5	48	MREQ
9	Ov	29	A4	49	OV
10	D3	30	A3	50	HALT
11	OV	31	A2	51	OV
12	D2	32	A1	52	NMI
13	Ov	33	A0	53	OV
14	D1	34	RST	54	INT
15	OV	35	OV	55	OV
16	D0	36	RFSH	56	WAIT
17	Ov	37	OV	57	OV
18	A15	38	M1	58	BUSREQ
19	A14	39	OV	59	OV
20	A13	40	BUSACK	60	SYS CLK (4MHz)

7) External Disc-Drive IDC Connector M004

PIN NO.	SIGNAL	PIN NO.	SIGNAL	DIRECTION
1	Ov	2	N/C	---
3	Ov	4	N/C	---
5	Ov	6	D/S-3	TO DRIVE
7	Ov	8	INDEX	TO COMPUTER
9	Ov	10	D/S-0	TO DRIVE
11	Ov	12	D/S-1	TO DRIVE
13	Ov	14	D/S-2	TO DRIVE
15	Ov	16	MOTOR ON	TO DRIVE
17	Ov	18	DIR/S	TO DRIVE
19	Ov	20	STEP	TO DRIVE
21	Ov	22	WRITE DATA	TO DRIVE
23	Ov	24	WRITE GATE	TO DRIVE
25	Ov	26	TRACK 0	TO COMPUTER
27	Ov	28	WRITE PROTECT	TO COMPUTER
29	Ov	30	READ DATA	TO COMPUTER
31	Ov	32	SIDE SELECT	TO DRIVE
33	Ov	34	N/C	

KEY:- N/C - No Connection
D/S - Drive Select
DIR/S - Direction Select

8) YUV-RGB Linear (Int.Select), 6-Pin-Din Socket for Video

PIN	RGB SIGNAL*	YUV OUTPUT*
1	R	V
2	G	Y+SYNCS
3	B	U
4	SYNCS	N/C
5	OV	OV
6	N/C	N/C

N/C - No Connection

*The outputs are internally user selectable via link switches M100 and M101. (See Appendix I).

Ext.Drive Video

APPENDIX G

ASCII Character Codes

INTERNATIONAL SET

ASCII Code	Character	ASCII Code	Character	ASCII Code	Character
000	NUL	043	+	086	V
001	SOH	044	,	087	W
002	STX	045	-	088	X
003	ETX	046	.	089	Y
004	EOT	047	/	090	Z
005	ENQ	048	0	091	
006	ACK	049	1	092	
007	BEL	050	2	093	
008	BS	051	3	094	
009	HT	052	4	095	
010	LF	053	5	096	'
011	VT	054	6	097	a
012	FF	055	7	098	b
013	CR	056	8	099	c
014	SO	057	9	100	d
015	SI	058	:	101	e
016	DLE	059	;	102	f
017	DC1	060		103	g
018	DC2	061	=	104	h
019	DC3	062		105	i
020	DC4	063	?	106	j
021	NAK	064	@	107	k
022	SYN	065	A	108	l
023	ETB	066	B	109	m
024	CAN	067	C	110	n
025	EM	068	D	111	o
026	SUB	069	E	112	p
027	ESCAPE	070	F	113	q
028	FS	071	G	114	r
029	GS	072	H	115	s
030	RS	073	I	116	t
031	US	074	J	117	u
032	SPACE	075	K	118	v
033	!	076	L	119	w
034	"	077	M	120	x
035	#	078	N	121	y
036	$	079	O	122	z
037	%	080	P	123	
038	&	081	Q	124	
039	'	082	R	125	
040	(	083	S	126	
041	)	084	T	127	DEL
042	*	085	U		

ASCII codes are in decimal
LF=Line Feed, FF=Form Feed, CR=Carriage Return, DEL=Delete

Key To Control Character Codes:-

NUL	-	Null
DLE	-	Data link Excercise
STX	-	Start Link Escape
ETX	-	End of text
ENQ	-	Enquiry
BS	-	Back Space
HT	-	Horizontal Tabulation
LF	-	Line Feed
VT	-	Vertical Tabulation
FF	-	Form Feed
CR	-	Carriage Return
SO	-	Shift - out
SI	-	Shift - in
ETB	-	End of Transmission Block
DEL	-	Delete
INS	-	Insert
ESC	-	Escape
FS	-	File Seperator
GS	-	Group Seperator
HOME	-	Cursor Home
US	-	Unit Seperator

APPENDIX H

CODE TABLE/GRAPHICS SET