Einstein MOS Calls *(Edited by Phil Simmons Document Revision 1.00 16/11/2006)*

This document has been compiled and checked against the Einstein Magazine article by Niqel Deakin 03/12/84 and Albert Revealed. I have re-worded some of the MOS call descriptions to try and give what I hope is the clearest explanation. If you find any errors or have any further information on undocumented MOS calls please email me phil_simmons@bluebottle.com for inclusion in future revisions of this document.

The following is a list of legitimate machine calls to Einstein MOS 1.1
The calls (MCAL 's) are executed on a RSTO8 (Restart 8 instruction) followed by the function number (given in Hex). Return is to the address after the function number.

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 MCAL 9C (ZIKEYIN) 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

Machine Code Monitor (De-Bug) Functions

Function	Label	Description
80	ARITH	Performs as 'A' (ARITHMETIC) from MOS Values Passed xxxx in HL pair yyyy in DE pair (See manual)
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 (see manual) If DE is zero, the mode and command bytes will remain unchanged. For the correct receive and transmit rates, the baud rate factor X 16 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)

Function	Label	Description
83	**DECIML**	**Performs as 'D' (DECIMAL) from MOS** **Values Passed** **xxxx in HL pair (see manual)**
84	**EXEC**	**Performs as 'E'.(EXECUTE) from MOS** **Values Passed** **xxxx - Break point - in HL pair (see manual)**
80	**ARITH**	**Performs as 'A' (ARITHMETIC) from MOS** **Values Passed** **xxxx in HL pair** **yyyy in DE pair (See manual)**
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 (see manual)** **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 (see manual)**
84	**EXEC**	**Performs as 'E'.(EXECUTE) from MOS** **Values Passed** **xxxx - Break point - in HL pair (see manual)**
85	**MFILL**	**Performs as 'F'(FILL) from MOS** **Values Passed** **xxxx - Start - HL pair** **yyyy - Finish - DE pair** **zz - Value - C register (see manual)**
86	**GOTO**	**Performs as 'G' (GOTO) from MOS** **Values Passed** **xxxx - execution address - HL pair** **yyyy - break point - DE pair (see manual)** **If DE is zero, no break point is set**

Function	Label	Description
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
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 Columns - C register If C is passed as zero, zz will default to 8 columns
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) 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.

I/O Functions

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 it 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

Function	Label	Description
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.
9F	ZPOUT	Outputs a character to the parallel printer held in the A register.
AO	ZSLOUT	Serial Output
		Outputs a character to the serial port from the A register.
Al	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 - Drive (0 - 3) FB51H HSTTRK - Track (0 - 27H) FB52H HSTSEC - Sector (0 - 9) FB53H HSTDMA - Sector buffer address (normally FEOOH)
A3	ZWSECT	Sector Write
		Writes a sector from the disc into the sector buffer (see MCAL A2).
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 (220H 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.

Function	Label	Description
A5	ZWBLK	Write Block Writes a block of data to the disc, the values passed are the same as for MCAL A4 and again is written to the next complete sector (200H bytes.
A6	ZCRLF	Outputs a CR (ODH) and LF (OAH)
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	ZPRHX	Outputs two hex digits held in the A register followed by a space.
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 FDC 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
BO	ZIGBLK	Returns a value in the A register from RAM pointed to by the DE pair if DE >7FFFH, or from ROM it DE <8000H. Note: Commas and spaces are ignored (i.e. the first non comma/non zero character is displayed).
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 or 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.

Dos Compatibility Functions

In order to be compatible with the Digital Research CP/M 2.2 operating system we have to make it look as if we are working with 128-byte sectors, not 512 byte sectors as we actually do. MOS contains routines which allow the read/write of logical sectors of 128 bytes in length, and which do the appropriate conversions. In practice speed is faster than if we were using true 128-byte sectors (as well as allowing greater disc capacity!), since all read/writes are done through a 512-byte 'window' called HSTBUF (at FE00H), meaning that many of them are done as memory block moves.

Note that these routines have been provided for the convenience of BDOS and will not usually be required by the user.

Function	Label	Description
B6	ZSLDSC	Selects drive with number in C. Also returns start of disk definition table in HL. Modifies A, BC and HL.
B7	ZSETRK	Set track no in C. Returns with this no in A
B8	ZSETSC	Set sector no in C. Returns with this no in A
B9	ZSETBUF	Set DMA buffer to address in BC

The 4 functions above set up scratch locations Disc, TRAK, SECT and DBUFF. This is all they do and do not physically affect the disc drive. That is left to the next 2 routines.

BA	ZRD128	Read logical (128-byte) sector from the disc.
BB	ZWR128	Write logical (128-byte) sector to disc.

These 2 routines require the user to supply the disc, track, sector and memory buffer address (DMA) in the scratchpad addresses DISC, TRAK, SECT and DBUF. Alternatively, the routines ZSLDSC, ZSETRK, ZSETSC and ZSETBF may be called to set up this information. All registers are affected by these routines, and A returns with the status of any read/write that may have occurred. I say 'may' because in many cases, where the PHYSICAL 512-byte sector that is currently in HSTBUF contains the required LOGICAL 128-byte sector, we will not have to do an actual read/write to disc at all, but just a memory move.

Initialisation Functions

Function	Label	Description
BC	ZZTEME	Set up CTC channel- 1 and 3 to generate 1 second Interrupts for clock
BD	ZFDRST	Resets the FDC after an error. Also resets the PSG (using MCAL CO (ZPINIT))
BE	ZSYSRS	Performs the following: 1. Clears the screen to 40 columns 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.

Function	Label	Description
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 8u 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 A register from 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**

Graphics Functions

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

C4	**ZPLOT**	**PLOT or UNPLOT a pixel.** **A = 1 for PLOT** **A = 0 for UNPLOT** **IX holds the x coordinate** **1Y 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 -LOCATION		
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**

Normally these 4 values are in ascending order For a continuous line, DOTON is set to FFH and the other 3 zero. For a continuous un-plot, DOTOFF is set to FFH; and the other 3 zero
Note: If all line type bytes are zero, the system will hang up.

C6	**ZPOINT**	**Returns the status of any pixel in the A register.** **I = foreground** **0 = background** **BC contains the x coordinate** **DE contains the y coordinate** **The VRAM address of the pixel is returned 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 y coordinate The VRAM address of the pixel is returned in the BC pair

Function	Label	Description
CB	**ZDRWRO**	**This will draw a line from the coordinates held in the IX and IY registers to values in scratch pad locations FB96H (XI) and FB98H (YI). Each is a two byte number The type of line drawn is determined by the line type values (see MCAL C5)**
C9	**ZPOLYG**	**This draws a polygon (or ellipse)**

The polygon centre coordinates are held in scratch pad locations FB9EH (CX) and FBAOH CY), each is 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 8OH will give an octagon
A value of 1OOH 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 (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 t y coordinate Origin value held in, scratch location FB9CH (ORGY - 2 Byte) 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)**
CD	**ZFILL**	**Fills an area on screen surrounding coordinates passed in the IX arid IY registers (x and y coordinate 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 set)then scratch location FBADH (FILLMOD) must be**

		set to zero. **MCAL XPNTXY (C7) can be used to find the fill type needed.**
CE	**ZINULT**	**Multiplies 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)**
CF	**ZPRM**	**Outputs a message to the screen. The data follows the CF 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 he useful to prevent scrolling, linefeeds etc.**
D1	**ZSCUW**	**Returns VRAM addresses relating to the cursor position.** **The ASCII text map address is returned in the BC pair (will be in the range 3COOH to 3FBFH)** **The table pattern generator address (first byte) is returned in the DE pair (will in the range 0000 to 17FFH).** **The start of the sprite pattern/text pattern table is returned in the HL pair (normally 1800H).**

2nd ROM functions

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** **Execution from here if 4000H is zero on power up or reset , after printing Einstein logo and before booting any disc present** **A RET will return execution to the disc auto boot routine** **4000H 4001H 4002H 4003H 4004H** **ROM detection byte** — **Execution from here on** **00 - for auto execution** — **MCAL D2**

EINSTEIN MOS 1.2 MCAL (ROM CALL) INFORMATION (Detailing changes from MOS 1.1)

Source: Nigel Deakin 4.12.84

New MCALS on MOS 1.2.

Number	Label	Description
D3	**ZIN80**	**Initializes 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 switch).** **3). Selects 40 columns or 80 columns according to 80 column card 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)**

Altered MCALS from MOS 1.1

All MCALS in MOS 1.1 which output information to the screen will output information to the 80 column card if present and selected. The exception to this is MCAL C3 (ZVROUT) and the graphics MCALS (C4 to CD) which still output to the VRAM.

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

M.S. Byte C - 40H to 47H 2K total
L.S. Byte B - 0 to FFH

Einstein Boot Sequence – following Power On or Reset.

1. Entire memory to 64k cleared by writing Value FFH to every byte

2. Interrupt vector, scratch-pad and interrupt service routines written to memory at FB00H.
MCAL and Break vectors written to low memory

3. Real time clock is zeroed and Z80 CTC (counter Timer Circuit) initialised.
The timer interrupt is enabled.

4. Serial chip (8251) set to transmit/receive at 9600 baud

5. Z80 PIO port A initialised for output to parallel port (printer) if present. Interrupt vector for Port is installed but not initialised.

6. Video Display Processor initialised. All sprites, function keys and screen cleared. Screen set up for 40 col.
Default character set is copied from MOS to VRAM.
Test made for 80 col card. If present screen memory cleared. If Auto-80 select jumper on card is enabled then display switched to 80 col.

7. Display * Einstein *** logo on screen.**

8. Location 4000H of spare ROM socket checked for presence of second ROM if present jump to location stored at 4001H and pass control to 2nd ROM.

9. Check for disk in drive 0 if present then read track 0 into memory and execute
(e.g. Xtal Dos loader or application).

10. Else drop into MOS monitor. Issue 'Insert disk and press Ctrl-Break' message and present the '>' prompt. Await user input.

Note that when MOS is 'paged in' it occupies shadow memory up to 8000H