MiniB supported OS calls

Introduction

This note describes the degree of emulation of the original BBC micro operating system performed by the operating system which is supplied with MiniB.

Not all of the original entry points at &FFxx are implemented either due to ROM space restrictions or differences between the MiniB and BBC micro hardware, though there are additional calls available (for example) to communicate with I²C devices.

Wherever possible the user is encouraged to use and interpret the values returned by system calls in preference to making an assumption about which facilities are available. This way, maximum compatibility is granted and software can be migrated seemlessly forwards and backwards from a "real" BBC micro and MiniB.

It is strongly recommended that the reader has a copy of the "Advanced User Guide for the BBC Micro" (Bray, Dickens, Holmes ISBN 0946827001) as this document does not attempt to detail in full all of the inner workings of a call - more a concise overview of the availability of a call and what to expect back as the answer.

Conventions used in this manual

The following typographical conventions are used throughout this guide:

Hexadecimal numbers are prefixed with ampersand.

Decimal numbers have no prefix.

Binary numbers may be denoted with a leading percent and given in decending bit significant order (ie.for an eight bit number they will be written in the order %76543210).

Multibyte data is stored in memory in little endian form.

Copyright

History

V0.24 Added information on how (and which) events are handled.

V0.25 Update to OSByte 201.

V0.26 Documents changes to VDU driver and updated OSByte support.

V0.31 Updates for OS 0.31

V0.32 Changed title

V0.39 Changes to the filing system section to reflect addition of ROM filing system in OS 0.38, added description to OSByte 200.

OSWrch

Outputs the character in A to all currently active output streams

Entry point	&FFEE
Indirected via	&20E
On entry	A=character to output
	X, Y unimportant
On exit	A, X, Y preserved
	NZCV undefined

Unrecognised VDU sequences are not currently indirected through UKVDUVector (&226).

Non vectored OSWrch is available at &FFCB, but use of this interface is not recommended.

Raw VDU output (where *FX3 settings are ignored) is available at &FFBC, but use of this interface is not recommended.

The following VDU control codes are implemented

Code	***Expected behaviour***	***Actual behaviour***
VDU0	do nothing	do nothing
VDU1	output the next byte to the printer	discards the next byte, there is no printer
VDU2	enable the printer	does nothing, there is no printer
VDU3	disable the printer	does nothing, there is no printer
VDU4	split text and graphics cursors	does nothing, there is no graphics mode
VDU5	join text and graphics cursors	does nothing, there is no graphics mode
VDU6	enable screen output	enable screen output
VDU7	bell	does nothing, there is no sound output
VDU8	backspace	backspace
VDU9	horizontal tab	horizontal tab
VDU10	line feed	line feed
VDU11	vertical tab	vertical tab
VDU12	clear screen	clear screen
VDU13	carriage return	carriage return
VDU14	page mode on	enables paged mode in conjunction with SHIFT
VDU15	page mode off	disables paged mode
VDU16	clear graphics window	does nothing, there is no graphics mode
VDU17	set text colour	does nothing, the LCD is monochrome
VDU18	set graphics colour	does nothing, there is no graphics mode
VDU19	set palette	does nothing, there is no graphics mode
VDU20	restore default colours	does nothing, the LCD is monochrome
VDU21	disable screen output	disable screen output
VDU22	change mode	clears the screen, there is only one mode
VDU23	misc ops	redefine soft characters control the cursor on/off state
VDU24	define graphics window	does nothing, there is no graphics mode
VDU25	plot operation	does nothing, there is no graphics mode
VDU26	restore default windows	does nothing, LCD too small to offer windows
VDU27	do nothing	do nothing
VDU28	define text window	does nothing, LCD too small to offer windows
VDU29	set graphics origin	does nothing, there is no graphics mode
VDU30	home	home
VDU31	position text cursor	position text cursor

The MiniB hardware supports a 20x4 monochrome LCD character display attached to the user port which does not support bitmapped graphics. However with such a compact display this restriction is unlikely to limit the applications for MiniB.

Redefining characters

The LCD display has limited capabilities for redefining characters, but these are still offered to the user through the use of VDU23.

The character array is built up from characters of size 5x7 which is not quite the same as the 8x8 sized cells that the true BBC micro offers. So the 8th part of the definition will be discarded and only 5 bits of each of the other 7 parameters will form part of the cell on screen.

The LCD display only allows a maximum of 8 soft characters at once and furthermore redefining a character a second time while the first instance is still on screen will cause both characters to adopt the new definition - this is because the soft character is held in off screen RAM and the screen is replotted from this every time it is refreshed (unlike the BBC micro which leaves the on screen bitmap untouched when one of the offscreen character definitions is altered). So the character number passed to VDU23 will by logically ANDed with 7 to choose which soft character will be redefined.

The result is that VDU23, character_number, row0, row1, row2, row3, row4, row5, row6, row7
will be interpretted as

soft_character=128+(character_number AND 7)
VDU23, soft_characters, row0, row1, row2, row3, row4, row5, row6, discarded

Cursor control

The cursor may be switched on and off by use of VDU23 as follows

on=1
off=0
VDU23, 1, <on | off>, 0, 0, 0, 0, 0, 0, 0

the last seven parameters are not important but shown here as zeros for clarity.

Missing characters

There are two different varients of the Hitachi LCD controller, those with part number 44780A00 stamped on them and those with part number 44780A02. The former is by far the most common and contains the ROM font for the Japanese market rather than the European market.
As a result the ROM character set does not contain the following characters which are available from the keyboard:

tilde (~)
pound (£)

If an application requires these two to be displayed correctly two of the soft characters can be reprogrammed for this use.

Video memory

There exists above HIMEM and below the base of ROM at &8000 a soft copy of the contents of the LCD display since values cannot be read back from the hardware. This is primarily used to allow scrolling of the non linear address map of the LCD display hardware - the user should not rely on this softcopy nor the format it is stored in as this may change if a different "shape" display from the same family is employed such as a 40x2 display.

OSRdch

Gets a byte from the current input stream, or waits if there are none available.

Entry point	&FFE0
Indirected via	&210
On entry	A, X, Y, unimportant
On exit	C=0 denotes that A contains the character read
	C=1 denotes an error
	the only currently defined error is A=27 denoting Escape was pressed
	X, Y preserved
	NZV undefined

Non vectored OSRdch is available at &FFC8, but use of this interface is not recommended.

OSNewl

Output &0A &0D to the currently selected output streams

Entry point	&FFE7
Indirected via	Not indirected, but will pass through OSWrchV
On entry	A, X, Y, unimportant
On exit	A=&0D
	X, Y preserved
	NZCV undefined

OSAsci

As per OSWrch, except if A=&0D on entry OSNewl is called instead

Entry point	&FFE3
Indirected via	Not indirected, but will pass through OSWrchV
On entry	A=character to output
	X, Y unimportant
On exit	A, X, Y preserved
	NZCV undefined

GSInit

Prepare a string in memory for processing by GSRead

Entry point	&FFC2
Indirected via	Not indirected
On entry	A, X, Y unimportant
	C=0 will consider a space, carriage return, or second quote mark as the terminator
	C=1 will consider a carriage return or second quote mark as the terminator
	(&F2) points to the string
On exit	Y=offset from (&F2) to the first non space character
	A=the first non space character
	X=preserved
	NCV undefined
	Z=1 if the string was empty

The general string processor offers a standardised way of processing strings entered by the user in a consistent manner, and also giving the ability to introduce escaped characters (those less than 32 and greater than 126) which could not otherwise be entered at the keyboard.

GSInit just sets up a status byte which GSRead will then use to process the string, but it is also useful in its own right to quickly strip leading spaces from strings.

GSRead

Read the next character from the string last initialised by GSInit

Entry point	&FFC5
Indirected via	Not indirected
On entry	A, X unimportant
	Y is the offset within the string last returned by GSRead or the call to GSInit
	(&F2) points to the string
On exit	A=character read
	Y=offset from (&F2) to the next character to be read
	X=preserved
	NZV undefined
	C=1 if the end of string has been reached

OSRdrm

Read a byte from paged ROM

Entry point	&FFB9
Indirected via	Not indirected
On entry	A, X unimportant
	Y=ROM number to read
	(&F6) points to the byte to read
On exit	A=byte read
	X, Y undefined
	NZCV undefined

This call allows a byte to be read from the specified paged ROM, and may also be called from a paged ROM which may be useful in applications such as a debugging ROM to allow disassembly of normally paged out software.

OSCLI

Pass a string to the command line interpreter

Entry point	&FFF7
Indirected via	&208
On entry	X, Y=point to the string, terminated by &0D
	A unimportant
On exit	A, X, Y undefined
	NZCV undefined

If the command cannot be found in the internal command table it will be passed to the ROMs as an unknown star command service call, then on to the current filing system if no ROM claimed the call.

OSEven

Simulate an event

Entry point	&FFBF
Indirected via	Not indirected
On entry	Y=the event number which will appear in A during the event
	A=the value which will appear in Y during the event
	X=any other parameter
On exit	A, X, Y preserved
	NZCV preserved

This causes an event to occur, provided it has been enabled (see OSByte 14).

Enabled events pass through EventV with interrupts disabled. The event handler must take care to preserve all registers, not reenable interrupts, and to avoid calling other OS routines which enable interrupts or which may already be threaded.

The following 10 events are defined:

0 - output buffer X is empty
1 - input buffer X is full, and character Y could not be inserted
2 - character Y inserted into buffer X
3 - ADC conversion complete on channel Y
4 - VSync start
5 - Interval timer passed through zero
6 - Escape condition
7 - RS423 error with 6850 status in X shifted right once and the character received in Y
8 - Econet network event
9 - User event

though only 0, 1, 2, 5, 6, and 9 are generated by MiniB.

The default routine pointed to by EventV is simply an RTS instruction.

OSByte

Change a system setting or effect

Entry point	&FFF4
Indirected via	&20A
On entry	A=setting to change
	X, Y=other parameters dependent on the value in A
On exit	A preserved
	X, Y call dependant
	NZV undefined
	C call dependant

If the value in A is not a value handled internally it will be passed to the ROMs as an unknown OSByte

On entry A=0, Return the operating system version
X=0 will cause an error with the error string being the version number
X>0 will return the version number in X
On exit
X=1 denoting that this is broadly equivalent to the BBC Model B

On entry A=1, Read/write the user flag
The user flag is a single OSByte location which is free for user applications
The effect of this call is to perform *newvalue = (oldvalue AND Y) EOR X*
On exit
X=oldvalue

On entry A=2, Select current input stream
Not yet implemented, the keyboard is always the current input stream

On entry A=3, Select current output stream(s)
Not yet implemented, the screen is always the current output stream

On entry A=4, Enable or disable cursor key effects
Not yet implemented

On entry A=5, Select current printer destination
Not yet implemented, there is no printing system currently

On entry A=6, Set character ignored by printer
X=character to ignore
On exit
A=preserved
X=old ignore character

On entry A=7, Set RS423 receive baud rate
As there is no serial hardware on MiniB this call does nothing

On entry A=8, Set RS423 transmit baud rate
As there is no serial hardware on MiniB this call does nothing

On entry A=9, Set mark of flashing colours
As the monochrome LCD cannot flash colours, on MiniB this call does nothing

On entry A=10, Set space of flashing colours
As the monochrome LCD cannot flash colours, on MiniB this call does nothing

On entry A=11, Set auto repeat delay
This determines the delay in centiseconds after which a held key will start to autorepeat.
X=repeat delay (or 0 for default)
On exit
X=old repeat delay

On entry A=12, Set auto repeat period
This determines the delay in centiseconds between repeats of a held key once the auto repeat delay has been exceeded.
X=repeat period (or 0 for default)
On exit
X=old repeat delay

On entry A=13, Disable events
Decreases the count for the event number in X. When the count is zero the event is completely disabled.
On exit
X=old event count

On entry A=14, Enable events
Increases the count for the event number in X.
On exit
X=old event count

On entry A=15, Flush chosen buffer class
Buffers cannot currently be flushed

On entry A=16, Select ADC channel(s) to sample
As there is no ADC hardware on MiniB this call does nothing

On entry A=17, Force an ADC conversion
As there is no ADC hardware on MiniB this call does nothing

On entry A=18, Reset the soft keys
Not yet implemented, currently there are no soft keys

On entry A=19, Wait for next VSync
As the LCD display does not have a sync signal, this call simply waits for 20ms before returning

On entry A=20, Explode selected region of the character set
The LCD hardware allows only 8 character redefinitions, so the font is always imploded

On entry A=21, Flush the specified buffer
This call passes X to the count and purge vector with V set
X=buffer to flush
0 = keyboard
1 = RS423 input
2 = RS423 output
3 = printer buffer
4-7 = sound buffers 0 to 3 respectively
8 = speech buffer
Attempting to flush a non existant buffer has undefined effects
On exit
X=preserved

OSBytes 22 to 116 inclusive are reserved for future expansion. They are currently not acted upon by MiniB.

On entry A=117, Read VDU status
Reads the VDU state. Only bit 7 (disabled) and bit 4 (paged mode on) are valid.

On entry A=118, Reflect keyboard status in keyboard LEDs
This call resynchronises the PS/2 keyboard LEDs after an OSByte 202.
On exit
X=top bit set if CTRL was pressed

On entry A=119, Close SPOOL and EXEC files
Spool and exec files are unimplemented at present

On entry A=120, Write current keys pressed information
This call writes two locations which are normally maintained by the keyboard driver to recall the most recently pressed 2 keys in rollover processing.
X=oldest pressed key number
Y=most recently pressed key number
On exit
A, X, Y preserved

On entry A=121, Scan the keyboard
Scans the key matrix from the internal key number passed in X
X=internal key number EOR &80, to scan for a single key
On exit
X<0 if chosen key was pressed
X=internal key number to start at, to scan for a range of keys
On exit
X=first pressed key encountered, or &FF for none

On entry A=122, Scan the keyboard from key 16
Simply calls OSByte 121 with X=16

On entry A=123, Inform the OS of a user printer driver going dormant
Sets the flags denoting that the printing system is now inactive

On entry A=124, Clear the Escape condition
Forcefully clears the escape condition

On entry A=125, Set the Escape condition
Forcefully sets the escape condition

On entry A=126, Acknowledge detection of an Escape
This will attempt to clear the Escape flag maintained by the OS.
On exit
X=&FF means the Escape condition was cleared
X=0 means the Escape condition was not cleared

On entry A=127, Check for EOF
This call tests whether the end of a file has been reached
X=file handle to check
On exit
X=0 if EOF has not been reached (otherwise it has)

On entry A=128, Read ADC channel or buffer status
This reason code interrogates the ADC channels, or the status of the built in buffers
X=0 returns the last ADC channel number to have completed a conversion
X=1-4 returns the ADC channel value for the channel passed in X
On exit
X=Y=0 denoting that no conversion has completed, as there is no ADC hardware on MiniB
X=NOT(buffer number) and Y=&FF
On exit
X = number of characters in the buffer for output buffers, or number of free spaces for input buffers

On entry A=129, Read key with time limit
This call performs several discrete functions dependant on the value in Y
On entry
Y=0-127
Scan for any key with a time limit defined by Y (MSB) and X (LSB)
On exit
Y=0 and C=0 then X=character detected
Y=&FF and C=1 then a timeout occurred
Y=&1B and C=1 then Escape was pressed
On entry
Y=&FF
X=-ve INKEY value=Scan for a specific key immediately
X=0=Return value representing operating system id in X
On exit
X=Y=&FF signifies the key being scanned for was being pressed, else 0

On entry A=130, Read machine high order address
These are the high 16 bits of the 32 bit address at which this processor is running
On exit
X=bits 16-23 of the machine address
Y=bits 24-31 of the machine address

On entry A=131, Read OSHWM
After all of the ROMs have claimed any RAM they need the top of the OS workspace is set to be the OSHWM
On exit
X=low byte of OSHWM
Y=high byte of OSHWM

On entry A=132, Read bottom of display memory
This is the equivalent of BASIC's HIMEM variable
On exit
X=low byte of HIMEM
Y=high byte of HIMEM

On entry A=133, Read bottom of display memory for a given mode
This allows the value of HIMEM to be determined without actually changing mode.
X=mode number
On exit
X=low byte of HIMEM in mode X
Y=high byte of HIMEM in mode X
As all of the modes are the same on MiniB, the value will be constant

On entry A=134, Read the current text cursor X and Y position
On exit
X=X position
Y=Y position

On entry A=135, Read the character at the current text input cursor position
On exit
X=character at text cursor position (or zero if unrecognised)
Y=current mode (always returns 5, which is a 20 column mode)

On entry A=136, Call USERV
This is directly equivalent to *CODE
X=value to pass to code
Y=value to pass to code
On exit
Depends on user code

On entry A=137, Switch on cassette relay
As there is no relay on MiniB this call does nothing

On entry A=138, Insert value into buffer
Inserts a single character into the given buffer
X=buffer to insert into
Y=character to insert

On entry A=139, Do *OPT
This is directly equivalent to *OPT
X=first parameter to *OPT
Y=second parameter to *OPT

On entry A=140, Do *TAPE
As there is no cassette hardware on MiniB this call does nothing

On entry A=141, Do *ROM
Selects and initialises the ROM filing system

On entry A=142, Enter language ROM
This call makes the given ROM number the current language
X=ROM to enter
Does not return

On entry A=143, Issue paged ROM service call
The given service call message will be passed to the ROMs for claiming
X=service call code
Y=any argument for that service call
On exit
Y=result from service call (if applicable)

On entry A=144, Do *TV
Does nothing, as the LCD vertical offset does not need compensating

On entry A=145, Get character from buffer
Removes a single character from the given buffer
X=buffer to insert into
On exit
C=1 denotes that the buffer was empty
C=0 means that Y=character removed

On entry A=146-151, Read/write memory mapped region

These are the Tube compatible methods of reading and writing from I/O

&FC00-&FCFF (*'Fred'*)	A=146 to read	A=147 to write
&FD00-&FDFF (*'Jim'*)	A=148 to read	A=149 to write
&FE00-&FEFF (*'Sheila'*)	A=150 to read	A=151 to write

For writes

X=offset within the region to access

Y=value to store

For reads

X=offset within the region to access

On exit

Y=value read

On entry A=152, Examine buffer status

Returns the status of the buffer specified in X

On exit

C=0 and Y=next value to be removed

C=1 means the buffer is empty, with Y preserved

On entry A=153, Insert character into buffer testing for Escape

Does nothing currently

On entry A=154, Write to video ULA and soft copy

As there is no video ULA in MiniB this call does nothing

On entry A=155, Write to video palette and soft copy

As there is no video palette in MiniB this call does nothing

On entry A=156, Read/Write 6850 control register

As there is no serial hardware on MiniB this call does nothing

On entry A=157, Fast access via Tube to BPUT

This calls the normal BPUT code in the host

X=byte to write

Y=file handle to write to

On entry A=158, Read byte from speech processor

As there is no speech processor on MiniB this call does nothing

On entry A=159, Write byte to the speech processor

As there is no speech processor on MiniB this call does nothing

On entry A=160, Read VDU variable

The VDU variables are currently for internal use only, so this call does nothing

On entry A=161
Reads a value from the CMOS RAM
X=location to read (0 to 55 inclusive)
On exit
Y=value read
A, X preserved

On entry A=162
Writes a value to the CMOS RAM
X=location to write (0 to 55 inclusive)
Y=value to write

OSBytes 163 to 165 inclusive are reserved for future expansion and are not currently acted upon by MiniB. Calls in the range 166 to 255 inclusive are infact just reading/writing values directly inside the OS workspace using the values in X and Y to determine the action:

newvalue = (oldvalue AND Y) EOR X
On exit
X=oldvalue

Hence, to write the value set Y=0 and X=value
to read the value set Y=255 and X=0
or some combination of bits where only certain bits are to be altered

On entry A=166-167, Read base of OSByte variables
These two values give the address of variables returned by OSBytes 166-255

On entry A=168-169, Read base of ROM extended vector table
These two values give the address of the start of the extended vector table in RAM

On entry A=170-171, Read base of ROM info byte table
These two values give the address of the 16 byte table of ROM type bytes for the installed ROMs in the machine

On entry A=172-173, Read base of keyboard translation table
This is currently zero for the PS/2 keyboard

On entry A=174-175, Read base of VDU variables
This is currently zero for the LCD display

On entry A=176, Read/write CFS timeout value
As there is no cassette filing system, this location will remain static

On entry A=177, Read/write currently selected input source
This location should only contain 0 (keyboard)

On entry A=178, Read/write keyboard semaphore
Not used by the PS/2 driver at present, set to zero

On entry A=179, Read/write initial OSHWM before font explosion
Default value of OSHWM before any font changes

On entry A=180, Read/write current OSHWM
See OSByte 131

On entry A=181, Read/write RS423 interception of Escape and soft keys
Not used, set to zero

On entry A=182, Read/write character definition explosion status
This location should only contain 0 (not exploded)

On entry A=183, Read/write CFS switch
Contains 2 during RFS use, and 0 during CFS use (default 2)

On entry A=184-185, Read/write video ULA and palette soft copies
See OSByte 154 and 155 respectively. Not used, set to zero

On entry A=186, Read/write ROM active at last BRK instruction
Contains the ROM number of the ROM which was paged in when the processor last executed a BRK

On entry A=187, Read/write ROM socket containing BASIC
If BASIC is fitted this location contains its ROM number, or &FF otherwise

On entry A=188, Read/write current ADC channel converting
Not used, set to zero

On entry A=189, Read/write highest ADC channel number
Not used, set to zero

On entry A=190, Read/write ADC conversion accuracy
Not used, set to zero

On entry A=191, Read/write RS423 in use flag
Not used, set to zero

On entry A=192, Read/write 6850 control soft copy
See OSByte 156

On entry A=193, Read/write flash counter
Not used, set to zero

On entry A=194, Read/write flash mark period
See OSByte 9

On entry A=195, Read/write flash space period
See OSByte 10

On entry A=196, Read/write keyboard auto repeat delay
See OSByte 11

On entry A=197, Read/write keyboard auto repeat period
See OSByte 12

On entry A=198, Read/write EXEC file handle
Not currently used, set to zero

On entry A=199, Read/write SPOOL file handle
Not currently used, set to zero

On entry A=200, Read/write effect of Escape and Break
Governs the actions of Escape and Break, for use as copy protection of programs
Bit 1=makes the next reset look like a power on reset (forces a complete RAM clear)
Bit 0=not currently used

On entry A=201, Read/write keyboard disable
When zero (default) the keyboard handler inserts characters into the input buffer, when non zero normal all normal keyboard processing occurs but no characters ever get inserted. This facility is for use by the Econet *REMOTE command.

On entry A=202, Read/write keyboard status
Contains a bit mask describing the status of the keyboard driver
As the PS/2 keyboard contains more keys than the original BBC Micro two extra bits are returned
Bit 2=clear to signify that NumLock is on
Bit 1=set to signify that ScrollLock is on
Bit 0=internal use only

On entry A=203, Read/write RS423 handshake threshold
Not used, set to zero

On entry A=204, Read/write RS423 input supression state
Not used, set to zero

On entry A=205, Read/write cassette/RS423 selection switch
Not used, set to zero

On entry A=206-208, Read/write Econet interception switches
If bit 7 of 206 is set OSByte and OSWords will be indirected through EconetV too
If bit 7 of 207 is set OSRdCh will be indirected through EconetV too (unimplemented)
If bit 7 of 208 is set OSWrCh will be indirected through EconetV too (unimplemented)

On entry A=209, Read/write speech suppression status
As there is no speech hardware on MiniB, this value contains a speech "NOP" opcode

On entry A=210, Read/write sound suppression status
Not used, set to zero

On entry A=211-214, Read/write VDU7 parameters
These 4 consecutive locations define the 4 parameters for a SOUND command which will be played when a BEL is required, and may include the use of envelopes.
Not used, set to zero

On entry A=215, Read/write !Boot option and suppression
Only two bits have a defined meaning in this variable
Bit 7=set will cause the normal startup banner to be printed (else suppressed)
Bit 0=set then any errors during the search for !Boot in ROM will be ignored but errors from a disc based !Boot will hang the machine as no language is present. When clear the opposite occurs.
Default value of &81 returned.

On entry A=216, Read/write number of characters remaining in a softkey expansion
Not currently used, set to zero

On entry A=217, Read/write lines printed to screen since last page
Not currently used, set to zero

On entry A=218, Read/write items in the VDU queue
Not currently used, set to zero

On entry A=219, Read/write character representing TAB
When the TAB key is pressed this character will be substituted (default 9)

On entry A=220, Read/write character representing Escape
When the Escape key is pressed this character will be substituted (default 27)

On entry A=221-228, Read/write character interpretation for a group of *'F'* key codes
These locations affect the interpretation of groups of the function key characters entered at the keyboard in conjunction with SHIFT or CTRL or both together.
Not currently used.

On entry A=229, Read/write interpretation of Escape
Not currently used, set to zero

On entry A=230, Read/write flags determining the Escape effects
Not currently used, set to zero

On entry A=231, Read/write IRQ mask for interception of the user 6522
Not currently used, set to 255

On entry A=232, Read/write IRQ mask for interception of the 6850
Not currently used, set to zero

On entry A=233, Read/write IRQ mask for interception of the system 6522
Not currently used, set to 255

On entry A=234, Read/write Tube presence
As there is no Tube hardware this value is 0

On entry A=235, Read/write speech processor presence
As there is no speech hardware this value is 0

On entry A=236, Read/write output stream destination(s)
See OSByte 3

On entry A=237, Read/write cursor editing state
See OSByte 4

On entry A=238-241, Unused locations
Not used, set to zero

On entry A=242, Read/write serial ULA soft copy
Not currently used, set to zero

On entry A=243, Read/write timer toggle switch
To ensure consistent values are always returned for TIME, two clocks are maintained which are toggled between - this location contains the toggle value.

On entry A=244, Read/write soft key update consistency
Not currently used, set to zero

On entry A=245, Read/write printer output destination
See OSByte 5

On entry A=246, Read/write printer ignore character
See OSByte 6

On entry A=247-249, Read/write reset interception code
These 3 consecutive locations may contain a single 6502 "JMP" instruction to a piece of user installed code.
When the computer is reset this code will be jumped into twice
C=0 straight after reset
C=1 denotes that the reset banner has been printed and any Tube hardware ready

On entry A=250-251, Unused
Not used, set to zero

On entry A=252, Read/write current language ROM
The number of the current language ROM is stored in this variable

On entry A=253, Read/write last reset type
Can be used to determine what caused the last reset
For 0=a soft reset
For 1=a power on reset
For 2=a hard reset

On entry A=254, Read/write base key value of numeric keypad keys
This value will be added to each of the numbers printed on the key tops, default &30.
On the BBC Micro this location contained a value denoting how much RAM was installed, &40 or &80 for 16K and 32K - this behaviour is not used here as the PS/2 keyboard contains a numeric keypad and the BBC Master used this location this way.

On entry A=255, Read/write startup options
Not currently used, returns 15.

OSWord

Change a system setting or effect requiring more than 2 parameters

Entry point &FFF1
Indirected via &20C
On entry A=setting to change
X, Y=point to a block containing other parameters
On exit A preserved
X,Y undefined
NZCV undefined

All OSWords (including the built in ones) are first offered to EconetV before other processing. If not claimed by EconetV the built in OSWords will then be handled, with unknown OSWords being passed to the paged ROMs and OSWords with A ⩾ &E0 being passed to UserV instead.

On entry A=0, Read a line from the current input stream
XY+0=16 bit address for the resulting string
XY+2=maximum line length to accept
XY+3=minimum accepted ASCii value
XY+4=maximum accepted ASCii value
On exit
C=1 if it was terminated by Escape
C=0 if it was terminated by Return and Y=length of string (including Return)

On entry A=1, Read the system clock
On exit
XY+0=5 byte system clock

On entry A=2, Write the system clock
XY+0=5 byte value to write
On exit
XY+0=unchanged

On entry A=3, Read interval timer
On exit
XY+0=5 byte event timer read

On entry A=4, Write interval timer
XY+0=5 byte event timer to write
On exit
XY+0=unchanged

On entry A=5, Read IO processor memory
XY+0=LSB of address
:
XY+3=MSB of address
The two high bytes of the 32 bit address should be &FFFF
On exit
XY+4=value read

On entry A=6, Write IO processor memory
XY+0=LSB of address
:
XY+3=MSB of address
XY+4=byte to write
The two high bytes of the 32 bit address should be &FFFF
On exit
XY+0=unchanged

On entry A=7, 8
XY+0=unimportant
On exit
XY+0=unchanged
V=0 the call was recognised but does nothing at present

On entry A=9, Read pixel value
XY+0=LSB of X coordinate
XY+1=MSB of X coordinate
XY+2=LSB of Y coordinate
XY+3=MSB of Y coordinate
On exit
XY+4=logical colour of coordinate or &FF if invalid
As there are no bitmap graphics this call always returns &FF.

On entry A=10, 11
XY+0=unimportant
On exit
XY+0=unchanged
V=0 the call was recognised but does nothing at present

On entry A=12, Write palette
XY+0=physical colour
XY+1=logical colour
XY+2=0
XY+3=0
XY+4=0
On exit
XY+0=unchanged
As there is no palette, this call does nothing

On entry A=13, Read last two graohics coordinates
On exit
XY+0=0
:
XY+7=0
As there are no bitmap graphics this call always returns a pair of 0's

On entry A=14, Read the real time clock

XY+0=0 return the time as a string
XY+0=1 return the time as BCD
XY+0=2 convert the following BCD time to a string

On exit

XY+0=of the form "Fri,31 Dec 1999.23:59:59"+CHR$13

or

XY+0=years
XY+1=months
XY+2=day of month
XY+3=day of week (1=Sunday)
XY+4=hours
XY+5=minutes
XY+6=seconds

On entry A=15, Write the real time clock

XY+0=8 set the time from a string of the form "HH:MM:SS"
XY+0=15 set the date from a string of the form "Ddd,DD Mmm YYYY"
XY+0=24 set the date and time from a string as returned by OSWord 14

On exit

XY+0=unchanged

OSFind

Get or release file handles for a given file

Entry point	&FFCE
Indirected via	&21C
On entry	A=0 to close a file
	Y=handle as assigned by OSFind to close a specific file
	Y=0 to close all open files
	A=&40 to open a file for input
	A=&80 to open a file for output
	A=&C0 to open a file for both input and output
	X, Y=point to the filename in memory
On exit	A=preserved when closing a file
	A=handle when opening a file (or zero if not found)
	X, Y preserved
	NZCV undefined

The underlying filing system will determine which reason codes are accepted or acted upon.

OSFSC

Miscellaneous filing system control

Entry point	No entry point
Indirected via	&21E
On entry	A=operation to perform
	X, Y=other parameters dependent on the value in A
On exit	A, X, Y as defined by the operation
	NZCV undefined

The underlying filing system will determine which reason codes are accepted or acted upon.

OSFile

Perform an operation on an entire file

Entry point &FFDD
Indirected via &212
On entry A=action to perform
X, Y=point to a block of the form
XY+0=16 bit address of the filename terminated by &0D
XY+2=load address of the file
XY+6=execution address of the file
XY+10=start address of data to save, length otherwise
XY+14=end address of data to save, attributes otherwise
On exit A=type of object found
A=0=nothing found
A=1=file found
A=2=directory found
X, Y preserved
NZCV undefined

The underlying filing system will determine which reason codes are accepted or acted upon

OSArgs

Change an open file's attributes

Entry point &FFDA
Indirected via &214
On entry Y=0
A=0 to read the currently active filing system id
A=1 to read the address of the tail of the last *RUN command
A=255 to flush all buffers to the media
Y=handle assigned by OSFind
A=0 to read PTR
A=1 to write PTR
A=2 to read EXT
A=255 to flush this file to the media
X=points to a 4 byte block in the IO processor
On exit A=preserved (except for A=0 Y=0)
X, Y preserved
NZCV undefined

The underlying filing system will determine which reason codes are accepted or acted upon.

OSBGet

Get a byte from the file handle in Y

Entry point	&FFD7
Indirected via	&216
On entry	Y=handle assigned by OSFind
	A, X unimportant
On exit	A=byte read
	X, Y preserved
	C=1=EOF reached, the byte read is not valid
	NZV undefined

The underlying filing system will determine which reason codes are accepted or acted upon.

OSBPut

Put a byte to the file handle in Y

Entry point	&FFD4
Indirected via	&218
On entry	Y=handle assigned by OSFind
	A=byte to put
	X unimportant
On exit	A, X, Y preserved
	NZCV undefined

The underlying filing system will determine which reason codes are accepted or acted upon.

OSGBPB

Read or write a group of bytes

Entry point	&FFD1
Indirected via	&21A
On entry	A=operation to perform
	X, Y=point to a block containing other parameters
	XY+0=handle as assigned by OSFind
	XY+1=pointer to data
	XY+5=number of bytes to transfer
	XY+9=sequential pointer to be used
On exit	A, X, Y preserved
	C=1=the operation could not be completed
	NZV undefined

The underlying filing system will determine which reason codes are accepted or acted upon.

Other notes

The following paragraphs do not yet merit a section of their own but are included here as an aid to users who may wish to experiment anyway.

Vectors

The vectors in page &2 are present at their usual addresses, and extended vector entry into paged ROMs is also available for any paged ROMs wishing to intercept a vector for their use.

Memory usage

The OS memory usage is broadly as per the BBC micro, but no assumption should be made about "magic" workspace locations unless documented here:

Page 0

Special locations due to the 6502 addressing mode

&00-&8F	allocated to the current language
&90-&9F	Econet
&A0-&A7	current NMI owner
&A8-&AF	OS workspace
&B0-&BF	filing system and OS scratch area
&C0-&CF	current filing system
&D0-&ED	OS workspace
&EE	RAM copy of the 1MHz bus paging register
&EF	value of A of last OSByte/OSWord
&F0	value of X of last OSByte/OSWord
&F1	value of Y of last OSByte/OSWord
&F2-&F3	pointer to string used for OS commands
&F4	RAM copy of the ROM latch
&F5	ROM filing system ROM number
&F6-&F7	ROM filing system ROM pointer
&F8-&FB	OS workspace
&FC	value of A at the last IRQ
&FD-&FE	pointer to last error message block
&FF	escape pressed flag, bit 7 set to signify an escape is pending

Page 1

The 6502 stack

The bottom of this page might also be used to copy error messages from paged ROM

Page 2

OS workspace

&200-&235	vectors
&236-&2FF	OS workspace

Page 3

VDU and OS workspace

&300-&3FF	OS workspace

Pages 4-7

Language workspace

&400-&7FF	free for use by the current language

Pages 8-12
OS buffers and workspace
&800-&CFF OS workspace
Page 13
ROM workspace
&D00-&D9E current NMI owner, NMIs will branch to &D00
&D9F-&DEF extended vectors for paged ROMs
&DF0-&DFF workspace for the installed ROMs, one byte each
Pages 14-127
Application workspace
The rest of RAM is left for applications, or further ROM workspace if claimed at reset
Pages 128-191
Paged ROM
This is the main flash ROM device which is reprogrammable in circuit
Pages 192-255
Operating system
This image also appears aliased in slot 15 of the ROM

Interrupts

When an interrupt occurs it will first be despatched either to BRKV if it was a software interrupt or to IRQ1V if it was a hardware interrupt.

All of the hardware interrupts generated by the onboard hardware are processed by the MiniB OS, and any which it does not expect or know how to handle will be passed to IRQ2V for the user to trap. Note though that the 6522 logical AND mask as set by OSByte 231 and 233 is currently ignored - you may not intercept interrupts coming from sources which MiniB OS handles as it will always handle them internally.

Filing systems & paged ROMs

A filing system may install itself in place of the default OS vectors which are described in OSFind/OSFile/ OSArgs/OSBput/OSBget/OSGBPB and OSFSC. When appropriate commands decoded by the OS (for example a *CAT command) the filing system will be called through these vectors to take action.

The paged ROMs will also be called at appropriate points through their service entry points.

ROM filing system

MiniB OS contains a default filing system which will be used when no other filing systems are present in paged ROM (or when it is selected with *ROM or equivalent means).

It allows fast access with error checking to programs stored serially in paged ROMs - up to 112k worth in total - and can be booted at startup by holding down the shift key.

The implementation is slightly enhanced compared with the ROM filing system present in the normal BBC Model B, in summary:

OSArgs	Reports the filing system identity
	Unlike the BBC Model B, the value of BASIC's PTR is readable
	Unlike the BBC Model B, the address of the command tail is readable
OSFile	Only A=255 (load) is possible
OSFSC	Extra commands (A=3) always cause an error
	Unlike the BBC Model B, the handle range (A=7) is also readable

OSFind	One input file is supported at once
	Attempts to open a file for update (OPENUP) are taken as OPENIN
OSBGet	One input file is supported at once, opened with OSFind
OSBPut	Not supported, always causes an error
OSGBPB	Not supported, always returns with C=1

RS423

There is no serial hardware nor software support. The 1MHz bus interface can be used for the addition of asynchronous serial port(s) instead.