

# 3 OSBYTE CALLS

OSBYTE calls are a powerful and flexible way of invoking many of the available operating system facilities.

OSBYTE calls are specified by the contents of the accumulator (A register) in the 6502. This means that up to 256 different calls can be made.

The command line interpreter (see section 2.11) performs OSBYTE calls in response to \*FX commands. This enables the user to make OSBYTE calls from the keyboard or within BASIC programs. It should be noted however that no results are returned by a \*FX call and so it is inappropriate to use certain OSBYTES in this way.

## **OSBYTE    Miscellaneous OS functions specified by the contents of the accumulator.**

Call address &FFF4 Indirected through &20A

On entry:

    A selects an OSBYTE routine.

    X contains an OSBYTE parameter.

    Y contains an OSBYTE parameter.

All calls are made to the OSBYTE subroutine at address &FFF4. This is then indirected through the vector at &20A (which means that user programs can intercept the OSBYTE calls before they get to the operating system if so desired). The selected function is determined by the accumulator contents. Two parameters can be passed to and from OSBYTE routines by putting the values to be passed in the X and Y registers respectively.

## Example

Using OSBYTE 4 to disable cursor editing.

From BASIC this would be typed as:

```
*FX 4,1
```

From assembly language it could be performed as:

```
LDA    #4    \Load accumulator with 4
LDX    #1    \Select cursor disabled option
JSR    &FFF4 \Make OSBYTE call
```

If an OSBYTE is not recognised by the Electron, it will be offered to any fitted paged ROMs (see chapters 8 to 11). The OSBYTE will then usually be claimed by the relevant expansion module's ROM. When OSBYTE is called directly, if none of the paged ROMs claim it then the call returns with the overflow flag set. If the OSBYTE itself was initiated by a \*FX command then the \*FX handler will generate the 'Bad command' error.

When OSBYTE calls are used in a second processor only a limited amount of information is returned. For low numbered OSBYTE calls (0 to 127) only the X register is returned and for high numbered OSBYTE calls only the X and Y registers, and the carry flag are returned.

All the OSBYTE calls recognised by the operating system are described on the following pages. The description for each call includes details of the entry parameters required and the state of the registers on exit. All OSBYTE calls may be made using the \*FX command, but it is not always appropriate to do so (i.e. those calls returning values in the X and Y registers). Where it is appropriate to use a \*FX command this has been indicated. Preceding the full OSBYTE descriptions is a complete summary of the OSBYTE calls in a list.

# OSBYTE/\*FX Call Summary

**dec.    hex.    function**

0	0	Print operating system version.
1	1	Set the User flag.
2	2	Select input stream.
3	3	Select output stream.
4	4	Enable/disable cursor editing.
5	5	Select printer destination.
6	6	Set character ignored by printer.
7	7	Set RS423 baud rate for receiving data.
8	8	Set RS423 baud rate for data transmission.
9	9	Set flashing colour mark state duration.
10	A	Set flashing colour space state duration.
11	B	Set keyboard auto-repeat delay interval.
12	C	Set keyboard auto-repeat rate.
13	D	Disable events.
14	E	Enable events.
15	F	Flush selected buffer class.
16	10	Select ADC channels to be sampled.
17	11	Force an ADC conversion.
18	12	Reset soft keys.
19	13	Wait for vertical sync.
20	14	Explode soft character RAM allocation.
21	15	Flush specific buffer.
22	16	Increment paged ROM polling semaphore
23	17	Decrement paged ROM polling semaphore
24	18	Change sound system.

OSBYTE/\*FX calls 25 (&19) to 114 (&72) are not used by OS 1.00.

115	73	Blank/restore palette.
116	74	Reset internal sound system.
117	75	Read VDU status.
118	76	Read keyboard status.
119	77	Close any SPOOL or EXEC files.
120	78	Write to two-key-roll-over locations.

121	79	Perform keyboard scan.
122	7A	Perform keyboard scan from 16 (&10).
123	7B	Inform OS, printer driver going dormant.
124	7C	Clear ESCAPE condition.
125	7D	Set ESCAPE condition.
126	7E	Acknowledge detection of ESCAPE condition.
127	7F	Check for EOF on an open file.
128	80	Read ADC channel or get buffer status.
129	81	Read key with time limit or key depression.
130	82	Read machine high order address.
131	83	Read top of OS RAM address (OSHWMM).
132	84	Read bottom of display RAM address (HIMEM).
133	85	Read bottom of display address for a given MODE.
134	86	Read text cursor position (POS and VPOS).
135	87	Read character at cursor position.
136	88	Perform *CODE.
137	89	Perform *MOTOR.
138	8A	Insert value into buffer.
139	8B	Perform *OPT.
140	8C	Perform *TAPE.
141	8D	Perform *ROM.
142	8E	Enter language ROM.
143	8F	Issue paged ROM service request.
144	90	Perform *TV (not implemented).
145	91	Get character from buffer.
146	92	Read from FRED, 1 MHz bus.
147	93	Write to FRED, 1 MHz bus.
148	94	Read from JIM, 1 MHz bus.
149	95	Write to JIM, 1 MHz bus.
150	96	Read from SHEILA, 1 MHz bus.
151	97	Write to SHEILA, 1 MHz bus.
152	98	Examine buffer status.
153	99	Insert character into input buffer.
154	9A	Reset video flash cycle.
155	9B	Reserved.
156	9C	Read/write 6850 control register and copy.
157	9D	'Fast Tube BPUT.'
158	9E	Read from speech processor.
159	9F	Write to speech processor.
160	A0	Read VDU variable value.

OSBYTE/\*FX calls 161 (&A1) to 165 (&A5) are not used by OS 1.00 and are reserved for future expansion.

166	A6	Read start address of OS variables (low byte).
167	A7	Read start address of OS variables (high byte).
168	A8	Read address of ROM pointer table (low byte).
169	A9	Read address of ROM pointer table (high byte).
170	AA	Read address of ROM information table (low byte).
171	AB	Read address of ROM information table (high byte).
172	AC	Read address of key translation table (low byte).
173	AD	Read address of key translation table (high byte).
174	AE	Read start address of OS VDU variables (low byte).
175	AF	Read start address of OS VDU variables (high byte).
176	B0	Read/write filing system timeout counter.
177	B1	Read/write input source.
178	B2	Undefined
179	B3	Read/write primary OSHWM.
180	B4	Read/write current OSHWM.
181	B5	Read/write RS423 mode.
182	B6	Read character definition explosion state.
183	B7	Read/write cassette/ROM filing system switch.
184	B8	Undefined.
185	B9	Read/write timer paged ROM service call semaphore.
186	BA	Read/write ROM number active at last BRK (error).
187	BB	Read/write number of ROM socket containing BASIC.
188	BC	Read current ADC channel.
189	BD	Read/write maximum ADC channel number.
190	BE	Read ADC conversion type.
191	BF	Read/write RS423 use flag.
192	CO	Read RS423 control flag.
193	C1	Read/write flash counter.
194	C2	Read/write space period count.
195	C3	Read/write mark period count.
196	C4	Read/write keyboard auto-repeat delay.
197	CS	Read/write keyboard auto-repeat period.
198	C6	Read/write *EXEC file handle.
199	C7	Read/write *SPOOL file handle.
200	C8	Read/write ESCAPE, BREAK effect.
201	C9	Read/write Econet keyboard disable.
202	CA	Read/write keyboard status byte.

203	CB	Read/write the ULA interrupt mask.
204	CC	Read/write Firm key pointer.
205	CD	Read/write length of current firm key string.
206	CE	Read/write Econet OS call interception status.
207	CF	Read/write Econet OSRDCH interception status.
208	D0	Read/write Econet OSWRCH interception status.
209	D1	Read/write speech suppression status.
210	D2	Read/write sound suppression status.
211	D3	Read/write BELL channel.
212	D4	Read/write BELL (CTRL G) sound information.
213	D5	Read/write BELL frequency.
214	D6	Read/write BELL duration.
215	D7	Read/write startup message and !BOOT options.
216	D8	Read/write length of soft key string.
217	D9	Read/write number of lines printed since last page.
218	DA	Read/write number of items in VDU queue.
219	DB	Read/write External sound flag.
220	DC	Read/write ESCAPE character value.
221	DD	Read/write i/p buffer code interpretation status.
222	DE	Read/write i/p buffer code interpretation status.
223	DE	Read/write i/p buffer code interpretation status.
224	E0	Read/write i/p buffer code interpretation status.
225	E1	Read/write function key status.
226	E2	Read/write firm key status.
227	E3	Read/write firm key status.
228	E4	Read/write CTRL+SHIFT+function key status.
229	E5	Read/write ESCAPE key status.
230	E6	Read/write flags determining ESCAPE effects.
231	E7	Reserved.
232	E8	Sound semaphore.
233	E9	Soft key pointer.
234	EA	Read flag indicating Tube presence.
235	EB	Read flag indicating speech processor presence.
236	EC	Read/write write character destination status.
237	ED	Read/write cursor editing status.
238	EE	Read/write OS workspace bytes.
239	EF	Read/write OS workspace bytes.
240	F0	Read country code.
241	F1	Read/write user flag location.
242	F2	Read RAM copy of &FE07.
243	F3	Read timer switch state.
244	F4	Read/write soft key consistency flag.

245	F5	Read/write printer destination flag.
246	F6	Read/write character ignored by printer.
247	F7	Read/write first byte of BREAK intercept code.
248	F8	Read/write second byte of BREAK intercept code.
249	F9	Read/write third byte of BREAK intercept code.
250	FA	Read/write OS workspace locations.
251	FB	Read/write OS workspace locations.
252	FC	Read/write current language ROM number.
253	FD	Read/write last BREAK type.
254	FE	Read/write available RAM.
255	FF	Read/write start up options.

## OSBYTE &00 (0)

Identify OS version

See OSBYTE &81 for more information regarding OS identification.

Entry parameters:

X=0	Execute BRK with a message giving the OS
version	
X<>0	RTS with OS version returned in X

On exit:

X=0, 05 1.00 or Electron OS 1.00  
X=1, OS 1.20 or American OS  
A and Y are preserved  
C is undefined

## OSBYTE &01 (1)

### Set the user flag

Entry parameters:

The user flag is replaced by X

On exit:

X=old value

This call uses OSBYTE with A=&F1 (241). This OSBYTE call is left free for user applications and is not used by the operating system. The user flag has a default value is 0.

## OSBYTE &02 (2)

### Select input stream

In the Electron any call with  $X \neq 0$  will result in an *unknown OSBYTE* service call being made to the paged ROMs unless a previous such call was recognised and thus changed the input source.

Entry parameters:

X determines input device(s)

\*FX 2,0    X=0    keyboard selected, RS423 disabled

\*FX 2,1    X=1    RS423 selected and enabled

\*FX 2,2    X=2    keyboard selected, RS423 enabled

Default: \*FX 2,0

On exit:

X=0 if previous input was from the keyboard

X=1 if previous input was from RS423

A is preserved

Y and C are undefined



## OSBYTE &03 (3)

### Select output stream

If RS423 output is selected in the Electron, paged ROM service calls are issued. In the absence of a suitable response this output is sunk (thrown away). The same applies to printer output if selected.

Bit 3 should not be used to enable the printer as this may conflict with the Econet protocol of claiming the printer.

Entry parameters:

X determines output device(s)

bit     o/p selected if bit is set

0	Enables RS423 driver
1	Disables VDU driver
2	Disables printer driver
3	Enables printer, independent of CTRL B or C
4	Disables spooled output
5	Not used
6	Disables printer driver unless the character is preceded by a VDU 1 (or equivalent)
7	Not used

\*FX 3,0 selects the default output options which are :

RS423 disabled

VDU enabled

Printer enabled (if selected by VDU 2)

Spooled output enabled (if selected by \*SPOOL)

On exit:

A is preserved

X contains the old output stream status

Y and C are undefined

## OSBYTE &04 (4)

Enable/disable cursor editing

Entry parameters:

X determines the status of the editing keys

- |         |     |  |
|---------|-----|--|
| *FX 4,0 | X=0 | Enable cursor editing (default setting)  |
| *FX 4,1 | X=1 | Disable cursor editing and make them return normal ASCII values like the other keys. |

The cursor control keys will return the following codes :

COPY	&87 (13S)
LEFT	&88 (136)
RIGHT	&89 (137)
DOWN	&8A (138)
UP	&8B (139)

- |         |     |   |
|---------|-----|---|
| *FX 4,2 | X=2 | Disable cursor editing and make the keys act as soft keys with the following soft key associations: |
|---------|-----|---|

COPY	11
LEFT	12
RIGHT	13
DOWN	14
UP	15

On exit:

A is preserved

X contains the previous status of the editing keys

Y and C are undefined

## OSBYTE &05 (5)

Select printer destination

Entry parameters:

X determines print destination

*FX 5,0	X=0	Printer sink (printer output ignored)
*FX 5,1	X=1	Parallel output
*FX 5,2	X=2	RS423 output (sink if RS423 enabled)
*FX 5,3	X=3	User printer routine (see section 6.5)
*FX 5,4	X=4	Net printer (see section 6.5)
*FX 5,5 to *FX5,255		User printer routine (see section 6.5)

Default setting: \*FX 5,0

On Exit:

A is preserved

X contains the previous \*FX 5 setting

Y and C are undefined

Interrupts are enabled by this call

This call is not reset to default by a soft break

## OSBYTE &06 (6)

Set character ignored by printer

Entry parameters:

X contains the character value to be ignored

*FX 6,10	X=10	This prevents LINE FEED characters being sent to the printer, unless preceded by VDU 1 (this is the default setting)
----------	------	--

On exit:

A is preserved

X contains the previous \*FX 6 setting

Y and C are undefined

This is not reset by soft BREAK.

## OSBYTE &07 (7)

Set RS423 baud rate for receiving data

This routine is not implemented on the unexpanded Electron. If this OSBYTE is used on the Electron an *unknown OSBYTE* service call is made to the paged ROMs.

This call is reserved for future expansion.

## OSBYTE &08 (8)

Set RS423 baud rate for data transmission

This routine is not implemented on the unexpanded Electron. If this OSBYTE is used on the Electron an *unknown OSBYTE* service call is made to the paged ROMs.

This call is reserved for future expansion.

## OSBYTE &09 (9)

Set duration of the mark state of flashing colours

(Duration of first named colour)

Entry parameters:

X determines duration

- |         |     |  |
|---------|-----|--|
| *FX 9,0 | X=0 | Sets mark duration to infinity<br>Forces mark state if space is set to 0 |
| *FX 9,n | X=n | Sets mark duration to n VSYNC periods<br>(n=25 is the default setting)   |

On exit:

A is preserved

X contains the old mark duration

Y and C are undefined

## OSBYTE &0A (10)

Set duration of the space state of flashing colours

(Duration of second named colour)

Entry parameters:

X determines duration

- \*FX 10,0 X=0      Sets space duration to infinity  
                         Forces space state if mark is set to 0
- \*FX 10,n X=n      Sets space duration to n VSYNC periods  
                         (n=25 is the default setting)

On exit:

A is preserved

X contains the old space duration

Y and C are undefined

## OSBYTE &0B (11)

Set keyboard auto-repeat delay

Entry parameters:

X determines delay before repeating starts

- \*FX 11,0 X=0      Disables auto-repeat facility
- \*FX 11,n X=n      Sets delay ton centiseconds  
                         (n=50 is the default setting)

After call,

A is preserved

X contains the old delay setting

Y and C are undefined

## OSBYTE &0C (12)

### Set keyboard auto-repeat period

Entry parameters:

X determines auto-repeat periodic interval

\*FX 12,0 X=0 Resets delay and repeat to default vals

\*FX 12,n X=n Sets repeat interval to n centiseconds  
(n=8 is the default value)

On exit:

A is preserved

X contains the old \*FX 12 setting

Y and C are undefined

## OSBYTE &0D (13)

### Disable events

Entry parameters: X contains the event code, Y=0

\*FX 13,0 X=0 Disable output buffer empty event

\*FX 13,1 X=1 Disable input buffer full event

\*FX 13,2 X=2 Disable character entering buffer event

\*FX 13,3 X=3 Disable ADC conversion complete event

\*FX 13,4 X=4 Disable start of vertical sync event

\*FX 13,5 X=5 Disable interval timer crossing 0 event

\*FX 13,6 X=6 Disable ESCAPE pressed event

\*FX 13,7 X=7 Disable RS423 RX error event

\*FX 13,8 X=8 Disable network error event

\*FX 13,9 X=9 Disable user event

See section 6.4 for information on event handling.

On exit:

A is preserved

X contains the old enable state (0= disabled)

Y and C are undefined

## OSBYTE &0E (14)

### Enable events

Entry parameters:

X contains the event code, Y=0

*FX 14,0	X=0	Enable output buffer empty event
*FX 14,1	X=1	Enable input buffer full event
*FX 14,2	X=2	Enable character entering buffer event
*FX 14,3	X=3	Enable ADC conversion complete event
*FX 14,4	X=4	Enable start of vertical sync event
*FX 14,S	X=S	Enable interval timer crossing 0 event
*FX 14,6	X=6	Enable ESCAPE pressed event
*FX 14,7	X=7	Enable RS423 RX error event
*FX 14,8	X=8	Enable network error event
*FX 14,9	X=9	Enable user event

After call,

A is preserved

X contains the old enable state (>0= enabled)

C is undefined

See section 6.4 for information on event handling.

## OSBYTE &0F (15)

### Flush selected buffer class

Entry parameters:

X value selects class of buffer

X=0                All buffers flushed

X=1                Input buffer flushed only

See OSBYTE call &16/\*FX 21

On exit,

Buffer contents are discarded

A is preserved

X, Y and C are undefined

## **OSBYTE &10 (16)**

Select ADC channels which are to be sampled

This routine is not implemented on the unexpanded Electron but is passed on to paged ROMs as an *unknown OSBYTE* paged ROM service call.

## **OSBYTE &11 (17)**

Force an ADC conversion

This routine is not implemented on the unexpanded Electron but is passed on to paged ROMs as an *unknown OSBYTE* paged ROM service call.

## **OSBYTE &12 (18)**

### **Reset soft keys**

This call clears the soft key buffer so the character strings are no longer available.

No parameters

On exit:

- A and Y are preserved
- X and C are undefined



## OSBYTE &13 (19)

### Wait for vertical sync

No parameters

This call forces the machine to wait until the start of the next frame of the display. This occurs 50 times per second on the UK Electron. Its main use is to help produce flicker free animation on the screen. The flickering effect is often due to changes being made on the screen halfway through a screen refresh. Using this OSBYTE call graphics manipulation can be made to coincide with the flyback between screen refreshes.

N.B. User trapping of IRQ1 may stop this call from working.

On exit:

A is preserved

X, Y and C are undefined

## OSBYTE &14 (20)

### Explode soft character RAM allocation

Entry parameters : X value explodes/implodes memory allocation

In the default state 32 characters may be user defined using the VDU 23 statement from BASIC (or the OSWRCH call in machine code). These characters use memory from &COO to &CFF. Printing ASCII codes in the range 128 (&80) to 159 (&9F) will cause these user defined characters to be printed up (these characters will also be printed out for characters in the range &A0-&BF, &C0-&DF, &E0-&FF), In this state the character definitions are said to be *imploded*.

If the character definitions are exploded then ASCII characters 128 (&80) to 159 (&9F) can be defined as before using VDU 23 and memory at &C00. Exploding the character set definitions enables the user to uniquely define characters 32 (&20) to 255 (&FF) in steps of 32 extra characters at a time. The operating system must allocate memory for this which it does using memory

starting at the ‘operating system high-water mark’ (OSHW). This is the value to which the BASIC variable PAGE is usually set and so if a totally exploded character set is to be used in BASIC then PAGE must be reset to OSHWM + &600 (i.e. PAGE=PAGE+&600).

ASCII characters 32 (&20) to 128 (&7F) are defined by memory within the operating system ROM when the character definitions are imploded.

See OSBYTE &83 (131) for details about reading OSHWM from machine code.

The memory allocation for ASCII codes in the expanded state is as follows :-

		ASCII code	Memory allocation
*FX 20,0	X=0	&80 - &8F	&C00 to &CFF (imploded)
*FX 20,1	X=1	&A0 - &BF	OSHW to OSHWM+&FF (+above)
*FX 20,2	X=2	&C0 - &DF	OSHW+&100 to OSHW+&1FF (+above)
*FX 20,3	X=3	&E0 - &FF	OSHW+&200 to OSHW+&2FF (+above)
*FX 20,4	X=4	&20 - &3F	OSHW+&300 to OSHW+&3FF (+above)
*FX 20,5	X=5	&40 - &5F	OSHW+&400 to OSHW+&4FF (+above)
*FX 20,6	X=6	&60 - &7F	OSHW+&500 to OSHW+&5FF (+above)

The explosion state can be determined using OSBYTE &B6.

Before the OSHWM is changed during a font explosion a service call is made to the paged ROMs warning them of the impending change.

On exit:

A is preserved

X contains the new OSHWM (high byte)

Y and C are undefined

## OSBYTE &15 (21)

### Flush specific buffer

While the unexpanded Electron only has a single sound channel the operating system has been designed to enable the implementation of an external sound system. Each time any of the sound buffers are flushed a paged ROM service call is issued with A=&17. In the unexpanded Electron there is a single effective buffer which may be addressed as any of the four channels. Thus flushing any of the four buffers will extinguish any sound being produced at that time.

See section 10.1 for more information regarding the Electron sound paged ROM service calls.

Entry parameters:

X determines the buffer to be cleared

*FX 21,0	X=0	Keyboard buffer emptied
*FX 21,1	X=1	RS423 input buffer emptied
*FX 21,2	X=2	RS423 output buffer emptied
*FX 21,3	X=3	Printer buffer emptied
*FX 21,4	X=4	Sound channel 0 buffer emptied
*FX 21,5	X=5	Sound channel 1 buffer emptied
*FX 21,6	X=6	Sound channel 2 buffer emptied
*FX 21,7	X=7	Sound channel 3 buffer emptied
*FX 21,8	X=8	Speech buffer emptied

See also OSBYTEs &0F (\*FX1S) and &80 (128).

On exit:

A and X are preserved

Y and C are undefined

## **OSBYTE &16 (22)**

### **Increment paged ROM polling semaphore**

This call increments the semaphore which when non-zero makes the operating system issue a paged ROM service call with A=&15 at centi-second intervals.

See paged ROM service call &15, chapter 10.

Entry parameters:

None

On exit:

A and X are preserved

Y and C are undefined

Semaphore is incremented once per call.

## **OSBYTE &17 (23)**

### **Decrement paged ROM polling semaphore**

This call decrements the semaphore which when non-zero makes the operating system issue a paged ROM service call with A=&15 at centi-second intervals.

See paged ROM service call &15, chapter 10.

Entry parameters:

None

On exit:

A and X are preserved

Y and C are undefined

Semaphore is decremented once per call.

## OSBYTE &18 (24)

### Select external sound system

This call is used to select a sound system which is implemented by an external hardware/software sound system.

Entry parameters:

X contains an undefined parameter

On exit:

A is preserved

All other registers are undefined

## OSBYTE &73 (115)

### Blank/restore palette

This call is used to temporarily turn all colours in the palette black. It should be useful for NMI users who want to generate NMIs with a high resolution screen display. This will ensure that there is no *snow* seen on the screen.

Entry parameters:

X=0 Restores the palette

X<>0 Set palette to all black if in high res. mode

On exit:

All registers undefined

## OSBYTE &74 (116)

### Reset internal sound system

This call can be used to reset the internal sound system.

Entry parameters:

X contains an undefined parameter

On exit:

All registers are undefined

## **OSBYTE &75 (117)**

### **Read VDU status**

No entry parameters

On exit the X register contains the VDU status. Information is conveyed in the following bits :

Bit 0	Printer output enabled by a VDU 2
Bit 1	Scrolling disabled e.g. during cursor editing
Bit 2	Paged scrolling selected
Bit 3	Software scrolling selected i.e. text window
Bit 4	reserved
Bit 5	Printing at graphics cursor enabled by VDU 5
Bit 6	Set when input and output cursors are separated (i.e. cursor editing mode).
Bit 7	Set if VDU is disabled by a VDU 21

On exit:

A and Y are preserved  
C is undefined

## **OSBYTE &76 (118)**

Reflect keyboard status in keyboard LEDs

This routine is hardware dependent and is implemented differently on the BBC microcomputer and the Electron. This call should not be used on either machine.

## **OSBYTE &77 (119)**

Close any SPOOL or EXEC files

This call closes any open files being used as \*SPOOLed output or \*EXECed input to be closed. This call is first offered to paged ROMs via a service call with A=&10. If the call is claimed then the operating system takes no further action. If the call is not claimed by a paged ROM the operating system closes any EXEC or SPOOL files itself. This call should be made by filing systems if they are deselected.

On exit:

A is preserved

X, Y and C are undefined

## **OSBYTE &78 (120)**

### **Write current keys pressed information**

This call should only be made by filing systems which have recognised a key pressed with BREAK and are initialising accordingly (see paged ROM service call with A=&03, section 10.1). This call should be used to write the old key pressed value to prevent its entry into the keyboard buffer.

The operating system operates a two key roll-over for keyboard input (recognising a second key press even when the first key is still pressed). There are two zero page locations which contain the values of the two key-presses which may be recognised at any one time. If no keys are pressed, location &EC contains 0 and location &ED contains 0. If one key is pressed, location &EC contains the internal key number+ 128 (see table below for internal key numbers) and location &ED contains 0. If a second key is pressed while the original key is held down, location &EC contains the internal key number+ 128 of the most recent key pressed and location &ED contains the internal key number + 128 of the first key pressed.

## Internal Key Numbers

hex.	dec.	key	hex.	dec.	key
&00	0	SHIFT	&40	64	CAPS LOCK
&01	1	CTRL	&41	65	A
&02	2	bit 0	&42	66	X
&03	3	bit 1	&43	67	F
&04	4	bit 2	&44	68	Y
&05	5	bit 3	&45	69	J
&06	6	bit 4	&46	70	K
&07	7	bit 5	&47	71	@
&08	8	bit 6	&48	72	:
&09	9	bit 7	&49	73	RETURN
&10	16	0	&50	80	SHIFTLOCK
&11	17	3	&51	81	5
&12	18	4	&52	82	C
&13	19	5	&53	83	G
&14	20	f4	&54	84	H
&15	21	8	&55	85	N
&16	22	f7	&56	86	L
&17	23	-	&57	87	;
&18	24	↑	&58	88	]
&19	25	left cursor	&59	89	DELETE
&20	32	f0	&60	96	TAB
&21	33	W	&61	97	Z
&22	34	E	&62	98	SPACE
&23	35	T	&63	99	V
&24	36	7	&64	100	B
&25	37	9	&65	101	M
&26	38	I	&66	102	,
&27	39	0	&67	103	.
&28	40	_	&68	104	/
&29	41	down cursor	&69	105	COPY
&30	48	1	&70	112	ESCAPE
&31	49	2	&71	113	f1
&32	50	D	&72	114	f2
&33	51	R	&73	115	f3
&34	52	6	&74	116	f4
&35	53	U	&75	117	f6
&36	54	0	&76	118	f8
&37	55	P	&77	119	f9
&38	56	[	&78	120	\
&39	57	up cursor	&79	121	rightcursor



Bits 0 to 7 refer to the the start up option byte. See OSBYTE &FF for further information about this byte.

To convert these internal key numbers to the INKEY numbers they should be EOR (Exclusive ORed) with &FF (255).

Entry parameters :

X and Y contain values to be written

Value in X is stored as the old key information.

Value in Y is stored in the new key information.

See also OSBYTE calls with A=&AC and A=&AD.

On exit:

A, X and Y are preserved

C is undefined

## **OSBYTE &79 (121)**

Keyboard scan

The keyboard is scanned in ascending numerical order. This call returns information about the first pressed key encountered during the scan. Other keys may also be pressed and a further call or calls will be needed to complete the entire keyboard scan.

Entry parameters:

X determines the key to be detected and also determines the range of keys to be scanned.

Key numbers refer to internal key numbers in the table above (see OSBYTE &78).

To scan a particular key:

X=key number EOR &80

On exit X<0 if the key is pressed

To scan the matrix starting from a particular key number:

X=key number

On exit X=key number of any key pressed or &FF if no key pressed

On exit:

A is preserved

X contains key value (see above)

Y and C are undefined

## **OSBYTE &7A (122)**

### **Keyboard scan from 16 decimal**

No entry parameters

Internal key number (see table above) of the key pressed is returned in X.

This call is directly equivalent to an OSBYTE call with A=&79 and X=16.

On exit:

A is preserved

X contains key number or zero if none pressed

Y and C are undefined

## **OSBYTE &7B (123)**

### **Inform operating system of printer driver going dormant**

Entry parameters:

X should contain the value 3 (printer buffer i.d.)

This OSBYTE call should be made by user printer drivers when they go dormant. The operating system will need to wake up the printer driver if more characters are placed in the printer buffer (see section 6.5).

On exit:

A and X are preserved

Y is preserved

C is undefined

## **OSBYTE &7C (124)**

### **Clear ESCAPE condition**

No entry parameters

This call clears any ESCAPE condition without any further action.

See OSBYTE &7E also.

On exit:

A, X and Y are preserved

C is undefined

## **OSBYTE &7D (125)**

### **Set Escape condition**

No entry parameters

This call partially simulates the ESCAPE key being pressed. The Tube is informed (if active). An ESCAPE event is not generated.

On exit:

A, X and Y are preserved  
C is undefined

## **OSBYTE &7E (126)**

### **Clear ESCAPE condition with side effects**

No entry parameters

This call attempts to clear the ESCAPE condition. All active buffers will be flushed, any open EXEC files closed, the VDU paging counter will be reset and the VDU queue will be reset.

See OSBYTE &E6 (230) also.

On exit:

X=&FF            if the ESCAPE condition cleared  
X=0             if no ESCAPE condition found

A is preserved  
Y and C are undefined

## OSBYTE &7F (127)

Check for end-of-file on an opened file

Entry parameters:

X contains file handle

On exit:

X<>0                    If end-of-file has been reached

X=0                     If end-of-file has not been reached

A and Y are preserved (Y not passed across Tube)

C is undefined

## OSBYTE &80 (128)

### Read ADC channel (ADVAL) or get buffer status

On the Electron this call will generate an *unknown OSBYTE* paged ROM service call when passed a positive value in the X register. If this service call is not claimed then the values in page 2 of memory allocated to storing ADC information are returned.

Otherwise this call is implemented identically on the BBC microcomputer and the Electron. Information about those buffers not used on the unexpanded Electron will be meaningless; these buffers have been implemented for expansion capability.

Entry parameters:

X determines action and buffer or channel number

If X=0 on entry:

Y returns channel number (range 1 to 4) showing which channel was last used for ADC conversion, Note that OSBYTE calls with A=&10 (16) and A=&11 (17) set this value too. A value of 0 indicates that no conversion has been completed. Bits 0 and 1 of X indicate the status of the two 'fire buttons'.

If X=1 to 4 on entry:

X and Y contain the 16 bit value (X-low, Y-high) read from channel specified by X.

If X<0 and Y=&FF on entry:

If X contains a negative value (in 2' s complement notation) then this call will return information about various buffers.

X=255	(&FF)	keyboard buffer
X=254	(&FE)	RS423 input buffer
X=253	(&FD)	RS423 output buffer
X=252	(&FC)	printer buffer
X=251	(&FB)	sound channel 0
X=250	(&FA)	sound channel 1
X=249	(&F9)	sound channel 2
X=248	(&F8)	sound channel 3
X=249	(&F7)	speech buffer

For input buffers X contains the number of characters in the buffer and for output buffers the number of spaces remaining.

On exit:

A is preserved  
C is undefined

## **OSBYTE &81 (129)**

### **Read key with time limit (INKEY)**

This call is functionally equivalent to the BASIC statement INKEY, It can be used to get a character from the keyboard within a time limit, scan the keyboard for a particular key press or return information about the OS type.

(a) Read key with time limit

Entry parameters:

X and Y specify time limit in centiseconds

If a time limit of n centiseconds is required,

$X = n \text{ AND } \&\text{FF (LSB)}$

$Y = n \text{ DIV } \&100 \text{ (MSB)}$

Maximum time limit is  $\&7\text{FFF}$  centiseconds (5.5 minutes aprox.)

On exit:

If key press detected,  $X = \text{ASCII key value}$ ,  $Y = 0$  &  $C = 0$

If key press not detected by timeout then  $Y = \&\text{FF}$  &  $C = 1$

If Escape is pressed then  $Y = \&1\text{B}$  (27) and  $C = 1$

(b) Scan keyboard for key press

Entry parameters:

X=negative INKEY value for key to be scanned

$Y = \&\text{FF}$

On exit:

$X = Y = \&\text{FF}$ ,  $C = 1$  if the key being scanned is pressed.

$X = Y = 0$ ,  $C = 0$  if key is not pressed.

(c) Return information about OS type

Entry parameters:

$X = 0$

$Y = \&\text{FF}$

On exit:

$X = 0$  BBC OS0.1

$X = 1$  Electron 05 1.00

$X = \&\text{FF}$  BBC 05 1.00 or 05 1.20

$X = \&\text{FE}$  US BBC OS1.20

## **OSBYTE &82 (130)**

### **Read machine high order address**

No entry parameters

This call yields the high order address required for the most significant 16 bits of the 32 bit addresses used for filing systems. The high order address is different in a second processor to that in an i/o processor. The Tube operating system intercepts this call to return the second processor high order address.

On exit:

X and Y contain the address (X-high, Y-low)

A is preserved

C is undefined

## **OSBYTE &83 (131)**

### **Return current OSHWM**

The OSHWM (operating system high water mark) represents the top of memory used by the operating system. This value is set after the paged ROMs have claimed workspace and any font explosion carried out. On a second processor this value represents the OSHWM on the i/o processor.

The OSHWM indicates the start of user memory and so this call is made by BASIC to initialise the value of PAGE.

No entry parameters

On exit:

X and Y contain the OSHWM address (X= low-byte , Y = high-byte)

A is preserved

C is undefined



## **OSBYTE &84 (132)**

### **Return HIMEM**

HIMEM is an address indicating the top of the available user RAM. This is usually the bottom of screen memory address. On a second processor this will be the bottom address of any code copied across from the I/O processor and executed.

No entry parameters

On exit:

X and Y contain the HIMEM address (X-low, Y-high)

A is preserved

C is undefined

## **OSBYTE &85 (133)**

### **Read bottom of display RAM address for a specified mode**

This call may be used to investigate the consequences of an intended MODE change. This enables languages to determine whether the selection of a new MODE should be allowed.

Entry parameters:

X determines mode number

On exit:

X and Y contain the address (X-low byte, Y-high byte)

A is preserved

C is undefined

## **OSBYTE &86 (134)**

### **Read text cursor position (POS and VPOS)**

When in cursor editing mode this call returns the position of the input cursor not the output cursor.

No entry parameters

On exit:

X contains horizontal position of the cursor (POS)

Y contains vertical position of the cursor (VPOS)

A is preserved

C is undefined

## **OSBYTE &87 (135)**

### **Read character at text cursor position and screen MODE**

No entry parameters

On exit:

X contains character value

(0 if character not recognised)

Y contains graphics MODE number

A is preserved

C is undefined

## **OSBYTE &88 (136)**

### **Execute code indirected via USERV (\*CODE equivalent)**

This call JSRs to the address contained in the user vector (USERV &200). The X and Y registers are passed on to the user routine.

See \*CODE, section 6.1.

## **OSBYTE &89 (137)**

**Switch cassette relay (\*MOTOR equivalent)**

Entry parameters:

X=0 relay off

X=1 relay on

The cassette filing system calls this routine with Y=0 for write operations and Y=1 for read operations. This enables the implementation of a dual cassette system with additional hardware and software

On exit:

A is preserved

X, Y and C are undefined

## **OSBYTE &8A (138)**

**Insert value into buffer**

Entry parameters:

X identifies the buffer (See OSBYTE &15)

Y contains the to be value inserted into buffer

On exit:

C=0 if value successfully inserted

C= 1 if value not inserted e.g. if buffer full

A is preserved

## **OSBYTE &8B (139)**

**Select file options (\*OPT equivalent)**

Entry parameters:

X contains file option number

Y contains the option value required

On exit:

A is preserved

C is undefined

## **OSBYTE &8C (140)**

**Select tape filing system (\*TAPE equivalent)**

No entry parameters

On exit:

A is preserved

C is undefined

## **OSBYTE &8D (141)**

**Select ROM filing system (\*ROM equivalent)**

No entry parameters

On exit:

A is preserved

X, Y and C are undefined

## **OSBYTE &8E (142)**

**Enter language ROM**

Entry parameters:

X determines which paged ROM is entered

The language ROM is entered via its entry point with A= 1.  
Locations &FD and &FE in zero page are set to point to the  
copyright message in the ROM.

There is no exit from this call.

## **OSBYTE &8F (143)**

### **Issue paged ROM service call**

See Service ROMs section 10.1

Entry parameters:

X=reason code

Y=parameter passed with service call

On exit:

Y may contain return argument (if appropriate) X=0 if a paged ROM claimed the service call

A is preserved

C is undefined

## **OSBYTE &90 (144)**

### **Alter display parameters (\*TV equivalent)**

On the Electron this call is not implemented and returns with registers preserved.

## **OSBYTE &91 (145)**

Get character from buffer

Entry parameters:

X contains buffer number (see OSBYTE &1S)

On exit:

Y contains the extracted character.

If the buffer was empty then C=1 otherwise C=0.

A is preserved

## OSBYTEs &92 to &97 (146 to 151)

### Read or Write to mapped I/O

Entry parameters:

X contains offset within page

Y contains byte to be written (for write calls)

OSBYTE call		Memory addressed	Name
read	write		
&92 (146)	&93 (147)	&FC00 to &FCFF	FRED
&94 (148)	&95 (149)	&FD00 to &FDFF	JIM
&96 (150)	&97 (151)	&FE00 to &FEFF	SHEILA

Refer to the hardware section for details about these 1 MHz buses.

On exit:

Read operations return with the value read in the Y register

A is preserved

C is undefined

## OSBYTE &98 (152)

### Examine Buffer status

Entry parameters: X contains buffer number

On exit:

Y=character value read from buffer if buffer not empty

Y is preserved if buffer empty

C=1 if buffer empty otherwise C=0

A and X are preserved

## **OSBYTE &99 (153)**

### **Insert character into input buffer, checking for ESCAPE**

Entry parameters:

X contains buffer number (0 or 1 only) Y contains the character value

X=0 keyboard buffer

X=1 RS423 input

If the character is an ESCAPE character and ESCAPEs are not protected (using OSBYTE &C8/\*FX 200 or OSBYTE &E5/\*FX229) then an ESCAPE event is generated instead of the keyboard event.

On exit:

A is preserved

X, Y and C are undefined

## **OSBYTE &9A (154)**

### **Reset flash cycle**

This call resets the flash cycle to the beginning of the mark state (i.e. to the first named colour of the pair) by manipulating the ULA registers.

There are no entry parameters.

On exit:

All registers are undefined

## **OSBYTE &9B (155)**

### **Write to video ULA palette register and OS copy (BBC micro)**

On the Electron this call is ignored by immediately executing an RTS instruction.

## **OSBYTE &9C (156)**

### **Read/update 6850 control register and OS copy (BBC micro)**

On the Electron this call causes the operating system to issue an *unknown OSBYTE* paged ROM service call but makes no further actions.

## **OSBYTE &9D (157)**

### **Fast Tube BPUT**

The byte to be output is channelled through the standard BPUT routine.

Entry parameters:

X = byte to be output

Y = file handle

On exit:

A is preserved

X, Y and C are undefined

## **OSBYTE &9E (158)**

### **Read from speech processor**

On the Electron this call causes the operating system to issue an *unknown OSBYTE* paged ROM service call but makes no further actions.

## **OSBYTE &9F (159)**

### **Write to speech processor**

On the Electron this call causes the operating system to issue an unknown OSBYTE paged ROM service call but makes no further actions.



## **OSBYTE &A0 (160)**

Read VDU variable value

This call is implemented on the Electron but is officially undefined and may change in future issues of the operating system software.

Entry parameters:

X contains the number of the variable to be read

On exit:

X=low byte of variable

Y=high byte of variable

A is preserved

C is undefined

## **OSBYTEs &A6 (166) and &A7 (167)**

Read start address of OS variables

<NEW VALUE>=(<OLD VALUE> ANDY) EOR X

The old value is returned in X. The contents of the next location are returned in Y.

This call returns the start address of the memory used by the operating system to store its internal variables.

On exit:

X=low byte

Y=high byte

A is preserved

C is undefined

## **OSBYTEs &A8 (168) and &A9 (169)**

### **Read address of ROM pointer table**

This call is implemented on the BBC microcomputer and the Electron. When used across the Tube the address returned refers to the I/O processor's memory.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This table of extended vectors consists of 3 byte vectors in the form Location (2 bytes), ROM no. (1 byte). See Paged ROM section 10.3 for a complete description of extended vectors.

On exit:

X=low byte

Y=high byte

A is preserved

C is undefined

## **OSBYTEs &AA (170) and &AB (171)**

### **Read address of ROM information table**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This call returns the origin of a 16 byte table, containing one byte per paged ROM. This byte contains the ROM type byte contained in location & 8006 of the ROM or contains 0 if a valid ROM is not present.

On exit:

X=low byte

Y=high byte

A is preserved

C is undefined

## **OSBYTEs &AC (172) and &AD (173)**

Read address of keyboard translation table

This call is implemented on the BBC microcomputer and the Electron. However it should be noted that this call is hardware specific due to the different keyboard matrix layout on different machines, When used across the Tube the address returned refers to the I/O processor's memory.

Use of this call is not recommended.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

On exit:

X=low byte

Y=high byte

## **OSBYTEs &AE (174) and &AF (175)**

Read VDU variables origin

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This call returns with the address of the table of internal VDU variables.

On exit:

X=low byte

Y=high byte

## OSBYTE &B0 (176)

Read/write CFS timeout counter

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This counter is decremented once every vertical sync pulse (50 times per second) which is also used for OSBYTE &13/\*FX 19. The timeout counter is used to time interblock gaps and leader tones.

## OSBYTE &B1 (177)

**Read input source flags**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location should contain 0 for keyboard input and 1 for RS423 input (i.e. contains buffer no.) and is used for OSBYTE &2. OSBYTE &2 should be used to change the input source as writing the flag with this call does not enable the relevant interrupts.

## OSBYTE &B2 (178)

This call is undefined on the Electron.

## **OSBYTE &B3 (179)**

### **Read/write primary OSHWM (for imploded font)**

This call should not be used as it has been re-allocated on other products in the Acorn-BBC range.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains the OSHWM page value for an imploded font (even when character definition RAM explosion has been selected) but after paged ROM workspace allocation has been made.

See OSBYTE &B4 and OSBYTE &14.

## **OSBYTE &B4 (180)**

### **Read OSHWM (similar to OSBYTE &83)**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

This call returns the page number of OSHWM in X.

This location is updated by any character definition RAM explosion which may have been selected and returns with the high byte of the OSHWM address (the low byte always being 0).

See OSBYTE &14.

## OSBYTE &B5 (181)

### Read/write RS423 mode

On the unexpanded Electron this call will have no effect unless a suitable hardware and software expansion has been performed to implement RS423.

<NEW VALUE>=(<OLD VALUE> AND Y) EOR X

The old value is returned in X. The contents of the next location are returned in Y.

Flag=0      ESCAPEs are recognised soft keys are expanded  
             character entering input buffer event generated  
             cursor editing performed

Flag=1      All characters enter input buffer  
(default)   character entering buffer event not generated

## OSBYTE &B6 (182)

Read character definition explosion state

Use of this call is not recommended as this OS BYTE has been reallocated on other products in the Acorn BBC range.

<NEW VALUE>=(<OLD VALUE> AND Y) EOR X

The old value is returned in X. The contents of the next location are returned in Y.

This location contains the state of font explosion as set by OSBYTE call with A=&14/\*FX 20.

## **OSBYTE &B7 (183)**

### **Read cassette/ROM filing system flag**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains 0 for \*TAPE selection and 2 for \*ROM selection. Other values are meaningless, and should not be used.

## **OSBYTE &B8 (184)**

This call is undefined on the Electron.

## **OSBYTE &B9 (185)**

### **Read/write timer paged ROM service call semaphore**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains a semaphore. If the contents of this location are non-zero the operating system will generate a paged ROM service call with a reason code of &15. This semaphore should only be read using this call. See OSBYTES &16 and &17 for information about setting semaphore and service ROMs chapter 10 for information about the paged ROM service call.

## **OSBYTE &BA (186)**

### **Read ROM number active at last BRK (error)**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains the ROM number of the paged ROM that was in use at the last BRK.

## **OSBYTE &BB (187)**

### **Read number of ROM socket containing BASIC**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

BASIC is recognised by the fact that it is a language ROM which does not possess a service entry. This ROM is then selected by the \*BASIC command. If no BASIC ROM is present then this location contains &FF.

## **OSBYTE &BC (188)**

### **Read current ADC channel**

This call is not implemented in the unexpanded Electron.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains the number of the ADC channel currently being converted. This call should not be used to force ADC conversions, use OSBYTE &11/\*FX 17.



## OSBYTE &BD (189)

### Read maximum ADC channel number.

This call is not implemented in the unexpanded Electron.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

The maximum channel number to be used for ADC conversions in the range 0 to 4. Set by OSBYTE &16/\*FX 10.

## OSBYTE &BE (190)

### Read ADC conversion type, 12 or 8 bits.

This call is not implemented in the unexpanded Electron.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

Set to &00, default (12 bit)  
Set to &08, 8 bit conversion  
Set to &0C, 12 bit conversion

## OSBYTE &BF (191)

### Read/write RS423 use flag.

This location is reserved for expansion software on the Electron.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

## **OSBYTE &C0 (192)**

### **Read RS423 control flag**

This location is reserved for expansion software on the Electron.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

## **OSBYTE &C1 (193)**

### **Read/write flash counter.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains the number of 1/50th sec. units until the next change of colour for flashing colours.

## **OSBYTE &C2 (194)**

### **Read/write space period count.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

Similar to OSBYTE &0A.

## **OSBYTE &C3 (195)**

**Read/write mark period count.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

Similar to OSBYTE &09.

## **OSBYTE &C4 (196)**

**Read/write keyboard auto-repeat delay.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This call is used by OSBYTE &0B.

## **OSBYTE &C5 (197)**

**Read/write keyboard auto-repeat period (rate).**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This call is used by OSBYTE &0C.

## **OSBYTE &C6 (198)**

**Read \*EXEC file handle.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This call should be used only to read this location as writing to it may have undefined effects. This location contains zero if no file handle has been allocated by the operating system.

## **OSBYTE &C7 (199)**

**Read \*SPOOL file handle.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This call should be used to read this location only. This location contains the file handle of the current SPOOL file or zero if not currently spooling.

## OSBYTE &C8 (200)

### Read/write ESCAPE, BREAK effect

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

bit 0 = 0	Normal ESCAPE action
bit 0 = 1	ESCAPE disabled unless caused by OSBYTE &7D/125
bits 1 to 7 = 0	Normal BREAK action
bits 1 to 7 = 1	Memory cleared on BREAK

e.g. A value 0000001x (binary) will cause memory to be cleared on BREAK.

## OSBYTE &C9 (201)

### Read/write keyboard disable.

This call should only be made by the Econet filing system.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

If this location contains 0 then the keyboard is scanned normally otherwise lock keyboard (all keys ignored except BREAK).

This call is used by the \*REMOTE Econet facility.

## OSBYTE &CA (202)

**Read/write keyboard status byte.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

bit 4=0	if CAPS LOCK active
bit 5=1	if Fn active
bit 6=1	if SHIFT active
bit 7=1	if CTRL active

All bits except the CAPS LOCK bit will only change transiently and are subsequently unlikely to be of use.

See also OSBYTE with A=&76 (118).

## OSBYTE &CB (203)

**Read/write the ULA Interrupt Mask**

See chapter 7 for a description of the interrupt handling routine.

## OSBYTE &CC (204)

**Read/write Firm Key Pointer**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

The value contained in this location is a pointer into the currently expanding firm key. For more information about the firm keys see language ROMs section 9.2.

## **OSBYTE &CD (205)**

**Read/write Length of current Firm key string.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains the length of the string currently being expanded from a Firm key. For more information about Firm keys see language ROMs section 9.2.

## **OSBYTE &CE (206)**

**Read/write Econet OS call interception status.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

If bit 7 of this location is set then all OSBYTE and OS WORD calls (except those sent to paged ROMs) are indirected through the Econet vector (&224) to the Econet. Bits 0 to 6 are ignored.

## **OSBYTE &CF (207)**

**Read/write Econet read character interception status.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

If bit 7 of this location is set then input is pulled from the Econet vector.

## **OSBYTE &D0 (208)**

**Read/write Econet write character interception status.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

If bit 7 of this location is set then output is directed to the Econet. Output may go through the normal write character on return from the Econet code.

## **OSBYTE &D1 (209)**

**Read/write speech suppression status.**

This location is not used in the unexpanded Electron and is reserved for future expansion.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

## **OSBYTE &D2 (210)**

**Read/write sound suppression status.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

Setting X to zero allows sound to be generated. Setting X non-zero will prevent any further sound being produced.

The old value is returned in X. The contents of the next location are returned in Y.



## OSBYTE &D3 (211)

**Read/write BELL (CTRL G) channel.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains the channel number to be used for the BELL sound. Default value is 3.

## OSBYTE &D4 (212)

**Read/write BELL (CTRL G) SOUND information.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains a byte which determines either the amplitude or the ENVELOPE number to be used by the BELL sound. If an ENVELOPE is specified then the value should be set to  $(\text{ENVELOPE no.} - 1) * 8$ . Similarly an amplitude in the range 15 to 0 must be translated by subtracting 1 and multiplying by 8.

The least significant three bits of this location contain the H and 5 parameters of the SOUND command (see User Guide).

Note that the internal sound system on the Electron will not allow the amplitude of the sound to be varied.

Default value 144 (&90) on the Electron.

## **OSBYTE &D5 (213)**

**Read/write bell (CTRL G) frequency.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This value contains the pitch parameter (as used by SOUND command third parameter) used for the BELL sound.

Default value 101 (&65) on the Electron.

## **OSBYTE &D6 (214)**

**Read/write bell (CTRL G) duration.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This value contains the duration parameter (as for SOUND command) used for the BELL sound.

Default value 6 on the Electron.

## **OSBYTE &D7 (215)**

**Read/write start up message suppression and !BOOT option status.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

- bit 0 If clear then ignore OS startup message. If set then print up OS startup message as normal.
- bit 7 If set then if an error occurs in a !BOOT file in \*ROM, carry on but if an error is encountered from a disc !BOOT file because no language has been initialised the machine locks up.  
If clear then the opposite will occur, i.e. locks up if there is an error in \*ROM.

This can only be over-ridden by a paged ROM on initialisation or by intercepting BREAK, see OSBYTE calls &F7 to &F9.

## OSBYTE &D8 (216)

**Read/write length of soft key string.**

<NEW VALUE>=(<OLD VALUE> AND Y) EOR X

The old value is returned in X. The contents of the next location are returned in Y.

This location contains the number of characters yet to be read from the soft key buffer of the current soft key. To clear input buffer use \*FX 15/OSBYTE &0F.

## OSBYTE &D9 (217)

**Read/write number of lines since last halt in page mode.**

<NEW VALUE>=(<OLD VALUE> AND Y) EOR X

The old value is returned in X. The contents of the next location are returned in Y.

This location contains the number of lines printed since the last page halt. This value is used by the operating system to decide whether to halt scrolling when paged mode has been selected. This location is set to zero during OSWORD call &00 to prevent a scrolling halt occurring during input.

## **OSBYTE &DA (218)**

**Read/write number of items in the VDU queue.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ ANDY}) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This contains the 2' s complement negative number of bytes still required for the execution of a VDU command.

Writing 0 to this location can be a useful way of abandoning a VDU queue, otherwise writing to this location is not recommended.

## **OSBYTE &DB (219)**

**Read/write External sound flag**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ ANDY}) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains a flag indicating that an external sound system has been selected using OSBYTE &18.

## **OSBYTE &DC (220)**

### **Read/write Escape character.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains the ASCII character (and key) which will generate an ESCAPE condition or event.

e.g. \*FX 220,32 will make the SPACE bar the ESCAPE key.  
Default value &1B (27).

## **OSBYTEs &DD (221) to &EO (224)**

### **Read/write I/P buffer code interpretation status.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

These locations determine the effect of the character values &CO (192) to &FF (255) when placed in the input buffer. See OSBYTEs &E1 (225) to &E4 (228) for details about the different effects which may be selected. Note that these values cannot be inserted into the input buffer from the keyboard. RS423 input or a user keyboard handling routine may place these values into the input buffer.

OSBYTE &DD affects interpretation of values &C0 to &BF  
OSBYTE &DE affects interpretation of values &D0 to &CF  
OSBYTE &DF affects interpretation of values &E0 to &EF  
OSBYTE &E0 affects interpretation of values &F0 to &FF

Default values &01 ,&DO,&EO and &FO (respectively)

## **OSBYTE &E1 (225)**

**Read/write function key status (soft keys/codes/null).** Input buffer characters &80 to &8F.

## **OSBYTE &E2 (226)**

**Read/write firm key status (soft key or code).**

Input buffer characters &90 to &9F.

## **OSBYTE &E3 (227)**

**Read/write firm key status (soft key or code).**

Input buffer characters &AO to &AF.

## **OSBYTE &E4 (228)**

**Read/write CTRL+SHIFT+function key Status (soft key or code).**

Input buffer characters &BO to &BF.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

These locations determine the action taken by the operating system when a function key is pressed.

value 0	totally ignore key.
value 1	expand as normal soft key.
value 2 to &FF	add n (base) to soft key number to provide 'ASCII' code.

The default settings are:-

fn keys alone	&01	expand using soft key strings
fn keys+ SHIFT	&01	expand using firm key strings
fn keys+CTRL	&01	expand using firm key strings
fn keys+SHIFT+CTRL	&00	key has no effect

When the BREAK key is pressed a character of value &CA is entered into the input buffer. The effect of this character may be set independently of the other soft keys using OSBYTE &DD (221). One of the other effects of pressing the BREAK key is to reset this OSBYTE call and so the usefulness of this facility is limited.

## OSBYTE &E5 (229)

**Read/write status of ESCAPE key (escape action or ASCII code).**

<NEW VALUE>=(<OLD VALUE> AND Y) EOR X

The old value is returned in X. The contents of the next location are returned in Y.

If this location contains 0 then the ESCAPE key has its normal action. Otherwise treat currently selected ESCAPE key as an ASCII code.

## **OSBYTE &E6 (230)**

**Read/write flags determining ESCAPE effects.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

If this location contains 0 then when an ESCAPE is acknowledged (using OSBYTE &7E/\*FX 126) then :

- EXEC file is closed (if open)
- Purge all buffers (including input buffer)
- Reset paging counter (lines since last halt)
- Reset VDU queue
- Any current soft key expansion is cleared

If this location contains any value other than 0 then ESCAPE causes none of these.

## **OSBYTE &E7 (231)**

**Read/write IRQ bit mask for the user 6522 (BBC micro)**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location is reserved for future Acorn expansion on the Electron.

## **OSBYTE &E8 (232)**

**Read/write sound semaphore**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

This location contains the sound semaphore.



## **OSBYTE &E9 (233)**

### **Read/write soft key pointer**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

This location contains the soft key pointer.

## **OSBYTE &EA (234)**

### **Read flag indicating Tube presence.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains 0 if a Tube system is not present and &FF if Tube chips and software are installed.

No other values are meaningful or valid.

## **OSBYTE &EB (235)**

### **Read flag indicating speech processor presence.**

This location is used differently on the BBC micro and the Electron.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location is reserved for future Acorn expansion on the Electron.

## **OSBYTE &EC (236)**

**Read/write write character destination status.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ BOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This call is used by OSBYTE &3/\*FX 3.

## **OSBYTE &ED (237)**

**Read/write cursor editing status.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This call is used by OSBYTE &4/\*FX 4.

## **OSBYTEs &EE (238) and &EF (239)**

**Read/write OS workspace bytes.**

These locations are reserved for future Acorn expansion on the Electron.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

## OSBYTE &F0 (240)

### Read country code

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location contains a value indicating the country for which this version of the operating system has been written.

country code	country
0	United Kingdom
1	United States

## OSBYTE &F1 (241)

### Read/write User flag location.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This call is not used by the operating system and is unlikely to be used by later issues either. This location is reserved as a user flag for use with \*FX 1.

Default value 0.

## OSBYTE &F2 (242)

Read RAM copy of location &FEO7

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ BOR } X$

This location contains a RAM copy of the last value written to the ULA at address &FEO7.

The old value is returned in X. The contents of the next location are returned in Y.

## OSBYTE &F3 (243)

**Read timer switch state.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

The operating system maintains two internal clocks which are updated alternately. As the operating system alternates between the two clocks it toggles this location between values of 5 and 10. These values represent offsets within the OS workspace where the clock values are stored. This OS workspace location should not be interfered with.

## OSBYTE &F4 (244)

**Read/write soft key consistency flag.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

If this location contains 0 then the soft key buffer is in a consistent state. A value other than 0 indicates that the soft key buffer is in an inconsistent state (the operating system does this during soft key string entries and deletions). If the soft keys are in an inconsistent state during a soft break then the soft key buffer is cleared (otherwise it is preserved).

## **OSBYTE &F5 (245)**

**Read/write printer destination flag.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This call is used by OSBYTE &5/\*FX 5. Using this call does not check for the printer previously selected being inactive or inform the user printer routine. See section 6.1.

## **OSBYTE &F6 (246)**

**Read/write character ignored by printer.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This call is used by OSBYTE &6/\*FX 6.

## **OSBYTEs &F7 (247), &F8 (248) and &F9 (249)**

**Read/write BREAK intercept code.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

The contents of these locations must be a JMP instruction for BREAKs to be intercepted (the operating system identifies the presence of an intercept by testing the first location contents equal to &4C - JMP). This code is entered twice during each break. On the first occasion C=0 and is performed before the reset message is printed or the Tube initialised. The second call is made with C= 1 after the reset message has been printed and the Tube initialised.

## **OSBYTEs &FA (250) and &FB (251)**

### **Read/write OS workspace locations.**

These locations are reserved for future Acorn expansions on the Electron.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

## **OSBYTE &FC (252)**

### **Read/write current language ROM number.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location is set after use of OSBYTE &8E/\*FX 126. This ROM is entered following a soft BREAK or a BRK (error).

## **OSBYTE &FD (253)**

### **Read hard/soft BREAK.**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y

This location contains a value indicating the type of the last BREAK performed.

value 0 - soft BREAK  
value 1 - power up reset  
value 2 - hard BREAK

## **OSBYTE &FE (254)**

### **Read/write available RAM (BBC micro)**

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

This location is reserved for future Acorn expansion.

Default value 0 in the unexpanded Electron.

## **OSBYTE &FF (255)**

Read/write start up options.

$\langle \text{NEW VALUE} \rangle = (\langle \text{OLD VALUE} \rangle \text{ AND } Y) \text{ EOR } X$

The old value is returned in X. The contents of the next location are returned in Y.

On the Electron the default value of this location is &FF (255) and this OSBYTE is the only way of resetting the start up options.

bits 0 to 2    screen MODE selected following reset. (MODE number = 3 bit value)

bit 3            if clear reverse action of SHIFT+BREAK.

bits 4 to 7    not used (reserved for future applications)

