This document defines various constants which relate to EXOS version 2.1. This includes the addresses of certain system variables and all codes which are used by EXOS calls, including error codes. Many of these constants also appear in the EXOS kernel specification but are gathered together here for completeness.

1. System Segment Addresses

The top of the system segment (segment 0FFh) contains certain defined addresses which may be accessed by the user of EXOS. These are:

0BFFFh	USR_P3	\ These are the four segments which
0BFFEh	USR_P2	\ were paged in when EXOS was last
0BFFDh	USR_P1	/ called. Undefined when not in EXOS.
0BFFCh	USR_P0	/

0BFFA/Bh	STACK_LIMIT	lower limit of the system stack which devices may use. Allows a safety margin for interrupts.
0BFF8/9h	RST_ADDR	Warm reset address, must be in page zero.
0BFF6/7h	ST_POINTER	Z-80 address of status line RAM.
0BFF4/5h	LP_POINTER	Z-80 address of start of video line parameter table.
0BFF3h	PORTB5	Current contents of Z-80 port 0B5h, used if any bits are modified.
0BFF2h	FLAG_SOFT_IRQ	Flag set to non-zero value to cause a software interrupt. The value will become the software interrupt code.
0BFF0/1h	SEC_COUNTER	16-bit seconds counter, never read by EXOS or built in devices.
0BFEEh	CR_DISP	Flag can be set non-zero by an extension ROM when it is called to be initialised (action code 8) to suppress the initial flashing ENTERPRISE display.
0BFED/Eh	USER_ISR	Address of user interrupt routine, must be in page zero. 0000h for no user interrupt routine.

2. EXOS Function Codes

Below are listed all the EXOS function codes. These are the values which follow the "RST 30h" instruction in an EXOS call. Any values other than these will give a .IFUNC error code. Details of parameters for these calls can be found in the EXOS kernel specification.

0	Reset EXOS
1	Open channel
2	Create channel
3	Close channel
4	Destroy channel
5	Read character
6	Read block
7	Write character
8	Write block
9	Read status
10	Set channel status
11	Special function
16	Set/read/toggle EXOS variable
17	Capture channel
18	Redirect channel
19	Set default device
20	Return system status
21	Link user device
22	Read EXOS boundary
23	Set USER boundary
24	Allocate segment
25	Free segment
26	Scan system extensions
27	Allocate channel buffer (device call only)
28	Explain error message
29	Load module
30	Load relocatable module
31	Set system time
32	Read system time
33	Set system date
34	Read system date

3. Error Codes

All EXOS calls return a status code which is zero to indicate success and negative to indicate that an error occurred. Positive non-zero codes are warning codes and are not widely used by EXOS, indeed there is only one warning code produced by EXOS and it is included here.

In the documentation all error codes are referred to by there names which by convention begin with a full stop. The names are listed here along with the values and meanings. The strings inside double quotes are the error messages produced by the EXOS "explain error code" function, where applicable. Those error codes which have no corresponding message are marked with an asterisk.

3.1 Warning Codes

* 07Fh .SHARE Shared segment allocated, only returned by an 'allocate segment' call.

3.2 General Kernel Errors

These errors are generated inside the EXOS kernel and generally user devices should not return these error codes.

0FFh .IFUNC "Invalid EXOS function code"

0FEh .ILLFN "EXOS function call not allowed"
This occurs for example if a device tries to open a channel, the user tries to allocate a channel buffer, or an interrupt routine makes an EXOS call.

0FDh .INAME "Invalid EXOS string"
Occurs when an invalid character is detected in a device or filename, or the string is too long.

0FCh .STACK "EXOS stack overflow"
Can occur from almost any EXOS call. Generally does not occur unless very deeply nested inside multiple devices.

0FBh .ICHAN "Channel does not exist"
Attempt to access a non-existent channel for reading, writing etc.

0FAh .NODEV "Device does not exist"
Attempt to open a channel to a non-existent device.

0F9h .CHANX "Channel exists"
Attempt to open a channel using a channel number which is already open.

* 0F8h .NOBUF
Occurs during an open channel if the device fails to make an 'allocate channel buffer' call, or makes one with unreasonable parameters (more than 64k total RAM requested for example).

0F7h .NORAM "Insufficient memory"
Occurs if there is not enough free memory for an operation. Can occur with open channel, link device, load system extension or load new application.

0F6h .NOVID "Insufficient video memory"
Occurs when there is sufficient RAM available for an operation, but moving the channel buffers would cause a video channel to move out of video RAM.

* 0F5h .NOSEG
No segments available when an 'allocate segment' call is made.

* 0F4h .ISEG
Attempt to free a segment which is not allocated to the user, or device. Also if attempt to set user boundary when there is no shared segment.

* 0F3h .IBOUND
Attempt to set the user boundary above the EXOS boundary in the shared segment.

0F2h .IVAR "Unknown EXOS variable number"
Returned by 'read/write/toggle EXOS variable' if the variable number is not recognised by the EXOS kernel or any system extension.

0F1h .IDESC "Invalid device descriptor"
Occurs if the device descriptor given to a 'link device' EXOS call is invalid.

0F0h .NOSTR "Unrecognised command string"
Occurs if a string passed to a 'scan system extensions' call is not recognised by and extension.

0EFh .ASCII "Invalid file header"
Occurs if the first byte of a file read by 'load module' EXOS call is non-zero, or if the first two bytes are zero. Indicates that the file is not an Enterprise module format file.

0EEh .ITYPE "Unknown module type"
Occurs if the type of a module header read by 'load module' is not recognised by EXOS or any system extension.

0EDh .IREL "Invalid relocatable module"
Occurs if an invalid bit stream is encountered by the 'load relocatable module' call, or by the 'load module' call if it is loading a relocatable system extension.

* 0ECh .NOMOD
Returned by 'load module' if it finds a module header of type $$EOF, or if it gets a .EOF error when trying to read the first byte of a module header. Indicates that the file has finished in a controlled manner.

0EBh .ITIME "Invalid date or time value"
Returned by 'set date' or 'set time' if the values given are not valid.

3.3 General Device Errors

These errors are generated by device drivers but are not specific to any particular device. They can be returned by any device where it seems appropriate.

0EAh .ISPEC "Invalid special function call"
Results from a special function call to a device which does not recognise the given sub-function number.

0E9h .2NDCH "Device in use"
Attempt to open a channel to a device which only allows one channel, while there is already a channel open to it.

0E8h .IUNIT "Invalid unit number"
Returned by devices which interpret unit numbers if the unit number in an open or create channel call is unreasonable for that device.

0E7h .NOFN "Call not supported by this device"
Not all devices support all the EXOS channel calls (function codes 1...11). This is the error returned if for example an attempt is made to write to a keyboard channel.

0E6h .ESC "Invalid escape sequence"
Several devices interpret escape sequences to control various functions. If one of these devices receives an escape sequence which it doesn't understand then it returns this error.

0E5h .STOP "STOP key pressed"
This error is returned by a device if it aborts its current operation to service a STOP key software interrupt. The software interrupt will have occurred before the user gets this error code.

0E4h .EOF "End of file"
Returned by any file device if an attempt is made to read beyond the end of a file.

0E3h .PROT "Protection violation"
Allows device to provide some sort of file protection. The only built in device which supports this is the cassette driver.

3.4 Device Specific Errors

The following errors are generally specific to a single device driver and are only returned by that one device. Full explanations of the error codes can be found in the individual device driver specifications.

0E2h	.KFSPC	"Function key string too long"	(KEYBOARD)
0E1h	.SENV	"Envelope too big"	(SOUND)
0E0h	.SENBF	"Envelope storage full"	(SOUND)
0DFh	.SQFUL	"Sound queue full"	(SOUND)
0DEh	.VSIZE	"Invalid video page size"	(VIDEO)
0DDh	.VMODE	"Invalid video mode"	(VIDEO)
0DCh	.VDISP	"Invalid display parameters"	(VIDEO)
0DBh	not used		
0DAh	.VROW	"Invalid row number to scroll"	(VIDEO)
0D9h	.VCURS	"Invalid cursor coordinates"	(VIDEO,EDITOR)
0D8h	.VBEAM	"Invalid beam position"	(VIDEO)
0D7h	.SEROP	"Cannot use bot serial and network"	(SERIAL,NET)

0D6h	.NOADR	"Network address not set"	(NET)
0D5h	.NETOP	"Network link exists"	(NET)
0D4h	.EVID	"Editor video channel error"	(EDITOR)
0D3h	.EKEY	"Editor keyboard channel error"	(EDITOR)
0D2h	.EDINV	"Editor load file error"	(EDITOR)
0D1h	.EDBUF	"Editor load file too big"	(EDITOR)
0D0h	.CCRC	"Cassette CRC error"	(CASSETTE)

4. EXOS Variable Numbers

Below is a list of all the EXOS variables supported by the EXOS kernel and built in devices. More details of their functions can be found in the appropriate device or kernel specifications.

0 - IRQ_ENABLE_STATE
- b0 - set to enable sound IRQ.
- b2 - set to enable 1Hz IRQ.
- b4 - set to enable video IRQ.
- b6 - set to enable external IRQ.

b1,3,5 & 7 must be zero.

1 - FLAG_SOFT_IRQ. This is the byte set non-zero by a device to cause a software interrupt. It could also be set by the user to cause a software interupt directly. This variable is also available at a fixed address given in an earlier section.

2 - CODE_SOFT_IRQ. This is the copy of the flag set by the device and is the variable that should be inspected by a software interrupt service routine to determine the reason for the interrupt.

3 - DEF_TYPE Type of default device
- 0 => non file handling device (eg. TAPE)
- 1 => file handling device (eg. DISK)

4 - DEF_CHAN Default channel number. This channel number will be used whenever a channel call is made with channel number 255.

5 - TIMER 1Hz down counter. Will cause a software interrupt when it reaches zero and will then stop.

6 - LOCK_KEY Current keyboard lock status

7 - CLICK_KEY
- 0 => Key click enabled

8 - STOP_IRQ
- 0 => STOP key causes soft IRQ
- <>0 => STOP key returns code

9 - KEY_IRQ
- 0 => Any key press causes soft IRQ, as well as returning a code

10 - RATE_KEY Keyboard auto-repeat rate in 1/50 second

11 - DELAY_KEY Delay 'til auto-repeat starts
- 0 => no auto-repeat

12 - TAPE_SND
- 0 => Tape sound enabled

13	-	WAIT_SND	0 => Sound driver waits when queue full <>0 => returns .SQFUL error .. .. ..
14	-	MUTE_SND	0 => internal speaker active <>0 => internal speaker disabled
15	-	BUF_SND	Sound envelope storage size in 'phases'
16	-	BAUD_SER	Defines serial baud rate
17	-	FORM_SER	Defines serial word format
18	-	ADDR_NET	Network address of this machine
19	-	NET_IRQ	0 => Data received on network will cause a software interrupt
20	-	CHAN_NET	Channel number of network block received
21	-	MACH_NET	Source machine number of network block
22	-	MODE_VID	Video mode \ These variables select
23	-	COLR_VID	Colour mode \ the characteristics of
24	-	X_SIZ_VID	X page size / a video page when it
25	-	Y_SIZ_VID	Y page size / is opened
26	-	ST_FLAG	0 => Status line is displayed
27	-	BORD_VID	Border colour of screen
28	-	BIAS_VID	Colour bias for palette colours 8...16
29	-	VID_EDIT	Channel number of video page for editor
30	-	KEY_EDIT	Channel number of keyboard for editor
31	-	BUF_EDIT	Size of edit buffer (in 256 byte pages)
32	-	FLG_EDIT	Flags to control reading from editor
33	-	SP_TAPE	Non-zero to force slow tape saving
34	-	PROTECT	Non-zero to make cassette write out protected file
35	-	LV_TAPE	Controls tape output level
36	-	REM1	\ State of cassette remote controls,
37	-	REM2	/ zero is off, non-zero is on
38	-	SPRITE	Controls external sprite colour priority
39	-	RANDOM_IRQ	Incremented on every interrupt. Can be used as a source of random numbers provided it is only accessed infrequently.

5. Software Interrupt Codes

A software interrupt with a certain code can be caused by storing that code in the variable FLAG_SOFT_IRQ. The codes which are used by the built in devices are listed here. Details of enabling and disabling these can be found in the appropriate device driver specification.

10h...1Fh	(KEYBOARD)	Function keys 1 to 8 and also shifted function keys 1 to 8.
20h	(KEYBOARD)	STOP key pressed.
21h	(KEYBOARD)	Any normal key pressed (this is disabled by default).
30h	(NET)	Data block received on a network channel.
40h	(KERNEL)	The TIMER EXOS variable has counted down to zero.

6. Special Function Sub-Function Codes

All special function calls pass a sub-function number in register B. This defines what special function is required. The codes for the functions supported by the built in devices are listed here, along with which device they apply to. Details of the action of these calls can be found in the appropriate device driver specifications.

1	(VIDEO)	Display video page on screen
2	(VIDEO)	Return page size and mode
3	(VIDEO)	Return page size
4	(VIDEO)	Re-initialise character font
8	(KEYBOARD)	Program a function key
9	(KEYBOARD)	Read joystick (internal or external)
16	(NET)	Flush buffer
17	(NET)	Clear buffers
24	(EDITOR)	Set margin positions
25	(EDITOR)	Load a document file
26	(EDITOR)	Save a document file

7. Action Codes for System Extensions

Below is a list of the action codes which are passed around system extensions. All other values are reserved for future extensions. The meaning of these codes is explained in the EXOS kernel specification.

0	Do nothing
1	Cold reset
2	User command string
3	User help string
4	Unknown EXOS variable
5	Explain error code
6	Load file module
7	RAM allocation
8	Initialise system extension

8. Module Types for Load Files

This section lists all currently defined module types. Theses codes must appear in the second byte of the sixteen byte module header. Some of those defined below are not supported at all by the internal software but are used by IS-BASIC or other cartridge software. Those which are supported by EXOS or the built in devices are marked with an asterisk.

	0	ASCII file
	1	reserved for IS-FORTH
*	2	User relocatable module
	3	Multiple IS-BASIC program
	4	Single IS-BASIC program
*	5	New applications program
*	6	Absolute system extension
*	7	Relocatable system extension
*	8	Editor document file
	9	reserved for IS-LISP
*	10	End of file module

++++++++++ END OF DOCUMENT ++++++++++