Converting Spectrum programs to run on the Enterprise

1) Introduction

This document, together with the attached program listing, should be all that the experienced programmer needs to convert and run his/her Spectrum programs on the Enterprise. Throughout this document, it is assumed that that the programmer has no previous experience of Enterprise hardware, and so only those aspects that are relevant to the conversion are outlined. The heart of the Spectrum emulation is the Enterprise's extremely versatile video chip that can mimic most video chips on the market today. Thus, apart from a slight difference in attributes format, only the I/O ports are different. Even these have been made as similar as possible, only the port numbers are changed.

Using a Spectrum ROM emulator, several Spectrum programs have been loaded and run directly on the Enterprise with no modifications. It is perhaps ironic that those programs that will not run on the Spectrum 128 will more than probably run directly on a converted Enterprise with ROM emulation.

2) Enterprise Memory configuration.

Before describing the Spectrum emulator in detail, it is worthwhile describing how the Enterprise memory map works. You have probably heard that the Enterprise can address up to 4Mbytes of memory, so how does it do it? The Z80 address space is split up into four pages as follows:

```
page 0  Z80 addresses   0000  => 3FFFh
page 1  Z80 addresses   4000h => 7FFFh
page 2  Z80 addresses   8000h => BFFFh
page 3  Z80 addresses   C000h => FFFFh
```

Associated with each of these pages is an 8-bit I/O port. This selects which one of 256 16K memory segments is to reside in that page (256 * 16K = 4096K). Port number 0B0h selects which segment is in page 0, port 0B1h selects which segment is in page 1, and ports 0B2h and 0B3h select which segments go into pages 2 and 3 respectively. Any 16K segment can be in any page or pages. Throughout this document, page 0 will refer to the Space where the ROM would reside in the Spectrum, page 1 to the 16K where the video RAM is and so on.

3) The Enterprise setup program

The attached program first sets up the video chip to simulate the Spectrum video hardware, then sets up the sound chip and interrupts accordingly. Two spare RAM segments are placed into pages 2 and 3.

Because of copyright considerations, it is not possible to supply a complete copy of the Spectrum ROM, and so a RAM segment is left paged into page 0. Any programs that rely on Spectrum ROM calls should first load a series of routines to provide ROM emulation into page 0. Note that the user can now insert a jump to his own interrupt routine at 0038h for fast interrupt response. After every interrupt, the Enterprise interrupt latch must be reset by writing 30h to port 0B4h.

4) Speed differences between the Spectrum and Enterprise.

The Enterprise uses a Z80A processor running at a full 4MHz. The Spectrum uses a Z80A running at 3.5MHz. So, normally, there is a speed increase of about 15%. Note, however, that the RAM in page 1 is shared by the processor and the video chip, and so the Z80 is subject to clock stretching on all memory accesses in this segment, slowing it down by about 15 to 20%. It is recommended, therefore, that any time-critical routines should be assembled above 7FFFh.

5) Sound on the Enterprise.

The port number 0A8h on the Enterprise is configured by the setup program as a 6 bit D/A ladder network, outputting to the tape output port and internal speaker. Alternately writing 0 and 3Fh to this port will toggle this port in the same way as toggling bit 4 on port 0FEh of the Spectrum. This port is effectively a volume control, with numbers less than 3Fh producing progressively quieter sounds. Note also that this is only the left-hand output. Port 0ACh is identical, but for the right-hand output. Plug a pair of walkman-style headphones into the tape output port and you have stereo sound!

6) Colours on the Enterprise.

The macro LINE in the attached listing defines the video parameters for one scanline (one row of pixels). The eight colours shown are only a suggestion and are by no means fixed. These colours match directly with Spectrum colours 0 to 7, and so for direct conversions are quite adequate. The next section describes how to define your own colours for the Enterprise.

The Enterprise video chip can generate up to 256 colours, but the attributes mode only uses 16 of these. So the video chip allows you to select any eight out of the 256 colours available to form a 'palette'. The eight colours shown in the macro define palette colours 0 to 7 respectively. Each colour is defined by one byte with bits weighted as follows:

b7	b6	b5	b4	b3	b2	b1	b0
G	R	B	G	R	B	G	R

Where Total red = (4*b0 + 2*b3 + b6)/7
Total green = (4*b1 + 2*b4 + b7)/7
Total blue = (2*b2 + b5)/4

The palette colours 8 to 15 are defined by writing to bits b0..b4 of port 080h (BIAS). These bits are substituted directly into bits b3..b7 to determine the colour. Bits b0..b2 are taken from the palette number ie. 000 = colour 8, 001 = colour 9, 010 = colour 10 etc. With a bias of 0, the following colours are obtained for palette numbers 8 to 15:

black,red,green,yellow,blue,magenta,cyan,white

Altering the bias adds an offset of varying amounts of red, green and blue to these colours, rather like placing a coloured filter over the screen.

Bits b5..b7 of port 080h should normally be left as zero, but by setting bit 7, the internal speaker can be disabled.

The attributes byte format is as follows:

b7	b6	b5	b4	b3	b2	b1	b0
paper colour				ink colour			

The border colour can be set by simply writing an 8-bit number to port 081h. The colour is determined in the same way as for palette colours.

7) Scanning the Enterprise keyboard matrix.

The figure below shows the layout of the Enterprise keyboard matrix. The keyboard is scanned by first writing the row number to port 0B5h, and then reading back from this port. If a key on that row is being held down, then the appropriate bit loaded into A will be reset. So a routine for checking the STOP key could look like:

```
CHECK_STOP:     LD      A,07            ; Row number to read from
                OUT     (0B5h),A        ; Send row no. to kbd.
                IN      A,(0B5h)        ; Fetch keys
                RRA                     ; Put STOP in CY
                RET                     ; Return with CY reset if
                                        ; STOP key pressed.
```

Also, by writing to port 0B5h, and reading back from port 0B6h, The two external joysticks can be polled. Only one bit (b0) is used on each row for the joysticks, with rows 0 to 4 returning results from joystick port 1, and 5 to 9 from port 2. The following routine will poll the external joystick ports (a bit is reset if joystick moved).

```
READ_JOYS:      LD      L,2             ; Read both joysticks
                LD      C,0             ; First row
JOY_LOOP1:      LD      B,5             ; Read four directions+fire
                LD      E,D             ; On 2nd pass, put joy1 in E
JOY_LOOP2:      LD      A,C             ; Row number
                INC     C               ; Next row
                OUT     (0B5h),A        ; Scan row and get result
                IN      A,(0B6h)        ; Get each bit in turn and
                RRA                     ;  shift result into D
                RL      D               ;
                DJNZ    JOY_LOOP2       ;
                DEC     L               ; Have we done joy 2 yet?
                JR      NZ,JOY_LOOP1    ; No.
                RET
```

									ROW NUMBER
L.SHFT	Z z	X x	V v	C c	B b	| \	N n	JOY1 FIRE	0
CTRL	A a	S s	F f	D d	G g	LOCK	H h	JOY1 UP	1
TAB	W w	E e	T t	R r	Y y	Q q	U u	JOY1 DOWN	2
ESC	" 2	£ 3	% 5	$ 4	& 6	! 1	' 7	JOY1 LEFT	3
F1	F2	F7	F5	F6	F3	F8	F4	JOY1 RIGHT	4
	ERASE	~ ^	_ 0	= -	) 9		(8	JOY2 FIRE	5
	}]	* :	L l	+ ;	K k		J j	JOY2 UP	6
ALT	ENTER	JOY LEFT	HOLD	JOY UP	JOY RIGHT	JOY DOWN	STOP	JOY2 DOWN	7
INS	SPACE	R.SHFT	> .	? /	< ,	DELETE	M m	JOY2 LEFT	8
		{ [	P p	` @	O o		I i	JOY2 RIGHT	9
b7	b6	b5	b4	b3	b2	b1	b0	b0 Port 0B6h	

Read port 0B5h

8) Tape I/O on the Enterprise.

The programmer can load his/her programs on one of two different ways. The simplest way is to go through EXOS, the Enterprise's operating system. Each call to the operating system is a RST 030h instruction, followed by a single byte defining the type of call. All channel calls require a channel number in A. A status code is returned in A, with flags set. This is normally zero if the call was successful. So a routine to load a program from tape would look like:

```
open        EQU     1           ; operating system call
close       EQU     3           ;  function codes.
rdblk       EQU     6           ;
;
CHANNEL     MACRO   chan,fn             ; This MACRO defines a single
            LD      A,chan              ; call to the operating system
            RST     030h                ; with the channel number in A
            DB      fn                  ; and the type of call after the
            JP      NZ,ERROR            ; RST. ERROR is user's error
            ENDM                        ; trapping routine.
;
```

```
        LD      DE,FILENAME             ; open an inupt file
        CHANNEL 5,open                  ; from tape as channel 5
        LD      DE,START_ADDRESS        ; and read a block of
        LD      BC,LENGTH_OF_FILE       ; data from this channel
        CHANNEL 5,rdblk                 ;
        CHANNEL 5,close                 ; close the channel
        RET
;
FILENAME:   DB      6                   ; The length of the name
            DB      'TAPE:'             ; Load the first file you find
                                        ; on the tape.
;
```

For further information, consult the Enterprise Technical Specification available from Enterprise computers Ltd.

Note that once the Spectrum setup routine has been executed, parts of the system have had their control removed from the operating system, and so care should be taken when using EXOS calls after this has happened. An alternative method for loading from tape is to poll the hardware directly. Any software house that has written custom loaders for the Spectrum will find it fairly easy to modify these for use on the Enterprise. Also, from a copyright protection point of view, this is probably the best method. Bit b7 of port 0B6h can be polled in exactly the same way as the tape input on the Spectrum. Bit b6 of this port also provides an average level input for implementing tape level meters for setting tape volume. Tape players with external remote facilities can be controlled from the Enterprise by writing to bit b6 of port 0B5h. Bit 5 of this port controls the cassette sound feedthrough to the internal speaker. For accurate tape timing, the Enterprise sound chip can be configured to generate interrupts at rates of up to 250KHz. For further information, consult the Enterprise technical manual. Saving to tape is fairly easy also, by writing to the D/A port used for the sound output.

9) Program Enhancements on the Enterprise.

The above sections are all that is required for direct Spectrum conversions. For those of you who wish to enhance your programs to take advantage of the some of the Enterprise's more advanced features, here are a few possible tips.

By writing 0FEh to (say) port 0B3h, the Line parameter table that defines the video screen can be paged in. If it is paged in to page 3 (as above), then the table is accessible from 0C100h onwards. Each block of 16 bytes defines the parameters of one scan line, so bytes 0C100h to 0C10Fh define the top pixel row, 0C110h to 0C11Fh define the second row, and so on. Each row has its own set of eight palette colours, and so by changing these numbers, all 256 colours of the Enterprise can be used, with each row of pixels having its own set of eight. For row 1, ink 0 is located at 0C108h, ink 1 at 0C109h and so on, for row 2, ink 0 is located at 0C118h, ink 1 at 0C119h etc. The other parameters in each block should not be of any interest to the programmer apart from the interupt bit. Bit b7 of the second byte in each block can be set to generate a video interrupt at the start of that line. This could be used (for example), for modifying the bias register to provide multiple bias values. There has to be at least one normal row of pixels between each

video interrupt. The line parameter table is initially set up to provide a video interrupt immediately after the last row of pixels on the screen, for maximum interrupt service time. For further information about the video chip, consult the Nick chip specification at the back of the Enterprise technical manual.

After the line parameter table has been modified, the previous segment should be paged back in. The setup program configures the four pages as follows:

port 0B0h (page 0) = 0F8h
port 0B1h (page 1) = 0FDh
port 0B2h (page 2) = 0F9h
port 0B3h (page 3) = 0FAh

The Enterprise sound chip provides three seperate tone channels plus noise in full stereo, with ring modulation and filtering sound effects. Those programmers that wish to take advantage of this should read the Dave chip specification at the end of the Enterprise Technical manual.

The setup program provided assumes that the Enterprise is the 128K version. There are, however, still a number of 64K Enterprises around. The Spectrum emulation could check for this by looking for segment 0FCh in page 0. If the Enterprise is a 64, then only segments 0FCh to 0FFh can be paged in. this means that the video line parameter table cannot be paged out, and so should be put in page 0. Note also that all RAM will be classed as video RAM, with the appropriate reduction in execution speed.

******* END OF DOCUMENT *******

```
;
;                              Spectrum emulator
;                              =================
;
;-------------------------------------------------------------------------
;
;       This  program  configures  the Enterprise hardware to resemble as
;       closely  as  possible  the  hardware  of  the 48K Spectrum. It is
;       loaded  in at 0100h, and automatically runs once loaded. Spectrum
;       programs can then be loaded and run on the Enterprise with little
;       or no modifications. Read the companion notes to this program for
;       further information.
;
;-------------------------------------------------------------------------
;
                aseg
                .z80
;
                db      0,5                     ; EXOS module header
                dw      PROG_LEN
                db      0,0,0,0,0,0,0,0,0,0,0,0
;
                org     0100h                   ; New applications program
;
;-------------------------------------------------------------------------
;
;               EQUates
;
fixbias         EQU     80h                     ; Nick chip ports
border          EQU     81h                     ;
lpl             EQU     82h                     ;
lph             EQU     83h
;
sync            EQU     0A7h                    ;
page0           EQU     0B0h                    ; Dave chip ports
page1           EQU     0B1h                    ;
page2           EQU     0B2h                    ;
page3           EQU     0B3h                    ;
irq_mask        EQU     0B4h
;
vint            EQU     10000000b               ; Video chip parameters
vres            EQU     00010000b               ;
attr            EQU     00000100b               ;
pixel           EQU     00000010b               ;
reload          EQU     00000001b
;
LP_TAB          EQU     080F0h                  ; Address of new Line Parameter Table
                                                ; (LPT) in Nick chip address space.
;
black           EQU     00000000b               ; Video chip colours
red             EQU     01001001b               ;
green           EQU     10010010b               ;
blue            EQU     00100100b               ;
yellow          EQU     11011011b               ;
magenta         EQU     01010101b               ;
cyan            EQU     10110110b               ;
white           EQU     11111111b
;
;-------------------------------------------------------------------------
;
```

```
;       This MACRO can define any number of video scanlines from
;       1 to 256 lines. It defines the video mode, the left and right
;       margins, and the palette of colours available.
;
LINE            MACRO   numlines,mode,left,right
                db      -numlines,mode,left,right
                dw      0
                dw      0
                db      black,blue,red,magenta,green,cyan,yellow,white
                ENDM
;
;-------------------------------------------------------------------------
;
;               Entry point.
;
START:
                di                      ;
                ld      sp,STACK        ;
                ld      a,0FEh          ; Make Z80 LPT address=Nick chip
                out     (page2),a       ;  address.
                dec     a               ;
                out     (page1),a       ; RAM segment for new video RAM.
                ld      de,LP_TAB       ; Address of new LPT
                ld      hl,topline      ; First 30 scanlines are border only
                ld      a,193
;
LPT_SETUP:
                ld      bc,16           ;
                ldir                    ;
                ld      hl,sc           ; One LP block per scanline
                dec     a               ;
                jr      nz,LPT_SETUP
;
                ld      hl,vsync        ; Append vsync to LPT
                ld      bc,vsync_len    ;
                ldir
;
;-------------------------------------------------------------------------
;
;               Now set attribute addresses
;
                ld      hl,8104h        ; First attr destination.
                ld      de,5800h        ; Address of attributes
                ld      c,24            ; Number of attribute rows.
ATTR_LOOP2:
                ld      b,8             ; Each row controls 8 scanlines
ATTR_LOOP1:
                ld      (hl),e          ; Set attribute address
                inc     hl              ;
                ld      (hl),d          ;
                ld      a,l             ;
                add     a,15            ; HL+16 (next LP block)
                ld      l,a             ;
                djnz    ATTR_LOOP1      ; Same attributes for every 8 lines
                ld      a,0             ; Overflow from ADD A,15
                adc     a,h             ;
                ld      h,a             ;
                push    hl              ;
                ld      hl,32           ; Next attribute row
                add     hl,de           ;
                ex      de,hl           ;
                pop     hl              ;
                dec     c               ; Repeat for all attribute rows.
                jr      nz,ATTR_LOOP2
;
```

```
;       how to go about doing this, see the attached document.
;
;-------------------------------------------------------------------------
;
;               Data area
;
topline:        LINE    30,pixel,63,0
;
sc:             LINE    1,attr,15,47
;
vsync:          LINE    30,vint+pixel,63,0
                LINE    14,vres+pixel,63,0
                LINE    3,vres,63,0
                LINE    4,vres,06,63
                LINE    1,vres,63,32
                LINE    4,vres+pixel,6,63
                LINE    34,vres+pixel+reload,63,0
;
vsync_len       EQU     $-vsync
;
STACK           EQU     $+100h
;
PROG_LEN        EQU     $-100h
;
                db      0,0Ah,0,0,0,0,0,0       ; EXOS end-of-file module
                db      0,0,0,0,0,0,0,0         ; header
;
                end
;
```