HRprint.bas

The High Resolution printing routine allows standard sized characters to be positioned anywhere on the screen.

Usage

There is a an example program that uses this at the end of the page.

```
HRPrint(x,y,character,attribute,over)
```

Where * x is the x value in pixel co-ordinates * y is the y value in pixel co-ordinates * character is the memory address of the UDG style bytes for the character being printed. To print standard characters, use the address of the character from the ROM table. * attribute is the attribute byte value (As you'd get from the ATTR function) * over is 0 or 1, and chooses whether to print in OVER 0 or OVER 1 mode.

Prints the character to the screen at the given pixel co-ordinates.

> NOTE: The ZX Spectrum's attribute system is encoded into the hardware as a 32 character grid. HRPrint does its best, but changing the paper/bright/flash colour from the background is likely to look imperfect as the attribute blocks cannot line up well with the character printed unless it overlaps - as a result, the colour of attribute squares nearby is likely to be changed.

CODE

```
SUB HRPrint (x as uByte, y as uByte, char as uInteger, attribute as uByte, overprint as uByte)
'High res Printing, based on code produced, with thanks, by Turkwel over on the WOS boards.
'Brought to ZX Basic by Britlion, June 2010.
'For overprint use:
'      0 -> Nop. Direct write
'      0xB6 -> OR  (i.e. Normal "sprite" write)
'      0xAE -> XOR (i.e. Like PRINT OVER 1)

Asm
      ld a,(IX+13) ; Get overprint value
      LD (HRPOver1),a
      LD (HRPOver2),a

      ld b,(IX+7)
      ld c,(IX+5)
      push bc ; save our co-ordinates.

;print_char:
      ld d,(IX+09)
      inc d
      dec d
      jr z, HRPrint_From_Charset
      ld e,(IX+08)
      ex de, hl
      jp HR_Print


HRPrint_From_Charset:
      ld  de,(23606)
      ld  h,0
      ld  l,(IX+8) ; character
      add  hl,hl
      add  hl,hl
      add  hl,hl
      add  hl,de

HR_Print:
      call HRPat

;convert the Y and X pixel values to the correct Screen Address  - Address in DE
      ld a,8

;set counter to 8 - Bytes of Character Data to put down
HRPrint0:
      push af
      ;save off Counter

      ld a,b
      cp 192
      jr c,HRprint1
      pop af
      jp HRPrintEnd

;don't print character if  > 191 - off the bottom of the screen - restore AF and exit Print routine
;[this can be removed if you are keeping tight control of your Y values]
HRprint1:
      push hl
      push de
      push de

;save off Address of Character Data, Screen Address, Screen Address
      ld a,c
      and 7
      ld d,a

;get lowest 3 bits of Screen address
      ld e,255

;set up E with the Mask to use - 11111111b = All On
      ld a,(hl)
      jr z,HRprint3
```

```
;get a Byte of Character Data to put down - but ignore the following Mask shifting
;if the the X value is on an actual Character boundary i.e. there's no need to shift anything
HRprint2:
      rrca
      srl e
      dec d
      jp nz,HRprint2

;Rotate the Character Data Byte D times - and Shift the Mask Byte as well, forcing Zeroes into the
;Left hand side. The Mask will be used to split the Rotated Character Data over a Character boundary
HRprint3:
      pop hl
;POP one of the Screen Addresses (formerly in DE) into HL
      ld d,a
      ld a,e
      and d

HRPOver1:
      or (hl)
      ld (hl),a

;take the Rotated Character Data, mask it with the Mask Byte and the OR it with what's already on the
;this takes care of the first part of the Byte
;[remove the OR (HL) if you just want a straight write rather than a merge]
      inc l
      ld a,l
      and 31
      jr z,HRprint4

;Increment the Screen Address and check to see if it's at the end of a line,
;if so then there's no need to put down the second part of the Byte
      ld a,e
      cpl
      and d

HRPOver2:
      or (hl)
      ld (hl),a

;Similar to the first Byte, we need to Invert the mask with a CPL so we can put down the second part o
;in the next Character location
;[again, remove the OR (HL) if you just want a straight write rather than a merge]
HRprint4:
      pop de
      inc d
      inc b

;get the Screen Address back into DE, increment the MSB so it points the the Address immediately below
;it and Increment the Y value in B as well
      ld a,b
      and 7
      call z,HRPat

;now check if the Y value has gone over a Character Boundary i.e. we will need to recalculate the Scre
;Address if we've jumped from one Character Line to another - messy but necessary especially for lines
      pop hl
      inc hl

;get the Address of the Character Data back and increment it ready for the next byte of data
      pop af
      dec a
      jp nz,HRPrint0

;get the Counter value back, decrement it and go back for another write if we haven't reached the end
      jp HRPrintAttributes

;HRPAT is a subroutine to convert pixel values into an absolute screen address
;On Entry - B = Y Value C = X Value   On Exit - DE = Screen Address
HRPat:
      ld a,b
      srl a
      srl a
```

```
      srl a
      ld e,a
      and 24
      or 64
      ld d,a
      ld a,b
      and 7
      add a,d
      ld d,a
      ld a,e
      and  7
      rrca
      rrca
      rrca
      ld e,a
      ld a,c
      srl a
      srl a
      srl a
      add a,e
      ld  e,a
      ret

HRPrintAttributes:
      pop bc ; recover our X-Y co-ordinates.
      ld d,0
      ld a,(IX+11) ; attribute
      and a
      jr z, HRPrintEnd  ; if attribute=0, then we don't do attributes.
      ld e,a ; pass to e

;transfer Attribute Byte to e for easier use
      ld a,b
      cp 192
      jr nc, HRPrintEnd

;check Y position and exit if off bottom of screen
      push bc

;save off Y and X values for later
      and 248
      ld h,22
      ld l,a
      add hl,hl
      add hl,hl
      srl c
      srl c
      srl c
      ld b,d
      add hl,bc

;calculate the correct Attribute Address for the Y\X values
      ld (hl),e

;set the Attribute - this is ALWAYS set no matter what the valid Y\X values used
      pop bc

;get the Y and X values back into BC
      ;call print_attribute2

;call the subroutine to see if an adjacent Horizontal Attribute needs to be set
print_attributes1:
      ld a,c
      cp 248
      jr nc,endPrintAttributes1

;check to see if we are at Horizontal character 31 - if so then no need to set adjacent Horizontal Att
      and 7
      jr z, endPrintAttributes1

;and don't set the adjacent Horizontal Attribute if there's no need to
      inc l
```

```
      ld (hl),e
      dec l

;increment the Attribute address - set the adjacent horizontal Attribute - then set the Attribute Addr
endPrintAttributes1:
      ld a,b
      cp 184
      jr nc, HRPrintEnd

;check to see if we are at Vertical character 23 - if so then no need to set adjacent Vertical Attribu
      and 7
      jr z, HRPrintEnd

;and don't set the adjacent Vertical Attribute if there's no need to & Exit routine
      ld a,l
      add a,32
      ld l,a
      ld a,d
      adc a,h
      ld h,a
      ld (hl),e

;set the Attribute address to the line below  - and set the adjacent Vertical Attribute
;
;drop through now into adjacent Horizontal Attribute subroutine - all RETs will now Exit the routine c
;
HRPrintAttribute2:   ld a,c
      cp 248
      jr nc, HRPrintEnd

;check to see if we are at Horizontal character 31 - if so then no need to set adjacent Horizontal Att
      and 7
      jr z, HRPrintEnd

;and don't set the adjacent Horizontal Attribute if there's no need to
      inc l
      ld (hl),e
      dec l

;increment the Attribute address - set the adjacent horizontal Attribute - then set the Attribute Addr
      ;ret

HRPrintEnd:
End Asm
END SUB
```

Usage Example

Here is a short program that demonstrates the HRPrint routine in use - in this case to animate a sprite. Four calls to the routine put the four characters of the sprite onto the screen, and similarly erase it.

```
#include "hrprint.bas"

CLS

DIM x,y AS UBYTE
DIM xd,yd AS BYTE
x=100
y=10
xd=1
yd=1

DO
    PAUSE 2
    HRPrint(x,y,32,56,0): REM Erase with a space (CHR$ 32)
    HRPrint(x+8,y,32,56,0)
    HRPrint(x,y+8,32,56,0)
    HRPrint(x+8,y+8,32,56,0)

    x=x+xd
    y=y+yd

    HRPrint(x,y,@gentle,76,0)
    HRPrint(x+8,y,@gentle+8,76,0xAE)
    HRPrint(x,y+8,@gentle+16,76,0)
    HRPrint(x+8,y+8,@gentle+24,76,0xAE)

    IF x<=0 OR x>=247 THEN xd=-xd
    IF y<=0 OR y>=184 THEN yd=-yd
LOOP
END

gentle:
ASM
    defb 15,15,15,15,15,15,13,15
    defb 240,144,208,208,240,240,176,240
    defb 15,14,63,0,0,12,26,30
    defb 176,112,252,0,0,24,104,120
END ASM
```