PutTile

putTile.bas

This subroutine takes a 2X2 tile of data from the given address and copies it to the screen co-ordinates at (x, y) - x and y in character addresses, where 0 <= x <= 31 and 0 <= y < =23.

Note that this uses pushes and pops to move the data, using the fastest known data moving algorithm for the Z80. As a consequence, while active it uses ALL the registers, including alternates and IY and IX as well as the Stack Pointer SP. It is kind enough to put these back for the purposes of exiting the subroutine though - ZX BASIC uses that register quite extensively.

Also, interrupts are disabled while the copying is happening. Considering that the stack pointer is likely pointing either at the screen or the tile data, an interrupt would be disastrous. If interrupts were enabled when the SUB is called, it should re-enable them again on exit.

Note the data format is across the tile - 2 bytes for the top row, then 2 bytes for the second row...and so on until there are 2 bytes for the 16th row. Then two bytes for the top two attributes, and 2 for the bottom. It uses 36 bytes of data, starting at the address given.

```
' Routine to place a 16 pixel by 16 pixel "Tile" onto the screen at character position x,y from adddre
' Data must be in the format of 16 bit rows, followed by attribute data.
' (c) 2010 Britlion, donated to the ZX BASIC project.
' Thanks to Boriel, LCD and Na_than for inspiration behind this.

' This routine could be used as the basis for a fast sprite system, provided all sprites can be in 4 c
' It can also be used to clean up dirty background (erase sprites), or put backgrounds from tiled bloc

' Note the comments about Self Modifying code should be ignored. This has been updated with IX+n metho
' (They would have to be accessed to change the memory anyway - may as well just access them directly

SUB putTile(x as uByte, y as uByte, graphicsAddr as uInteger)

ASM
JP pt_start

ptstackSave:
defb 0,0

pt_start:
ld a,i
push af ; Save interrupt status.

; Routine to save the background to the buffer

         DI ; we really, really, REALLY can NOT be having interrupts while the stack and IX and IY are

         PUSH IX
         PUSH IY
         ;ld      HL, 65535 ; Self modifying code should load this with the graphics address.
         LD D,(IX+9)
         LD E,(IX+8)
         EX DE,HL

;; Print sprites routine
         LD (ptstackSave), SP ; Save Stack Pointer

         LD SP,HL   ; now SP points at the start of the graphics.

       ; This function returns the address into HL of the screen address
         ld      a,(IX+5) ; Load in x - note the Self Modifying value
         ld      IYH, a ; save it
         ld      l,a
         ld      a,(IX+7) ; Load in y - note the Self Modifying value
         ld      IYL, a ; save it
         ld      d,a
         and     24
         add     a,64
         ld      h,a
         ld      a,d
         and     7
         rrca
         rrca
         rrca
         or      l
         add     a,2   ; Need to be to the right so backwards writing pushes land properly.
         ld      l,a

         ; SO now, HL -> Screen address, and SP -> Graphics. Time to start loading.

         POP BC    ; Row 0
         POP DE    ; row 1
         EX AF,AF'
         POP AF    ; row 2
         EX AF,AF'
         EXX
         POP BC    ; row 3
         POP DE    ; row 4
         POP HL    ; row 5
         EXX

         ; All right. We're loaded. Time to dump!
```

```
        LD IX,0
        ADD IX,SP  ; Save our stack pointer into IX

        LD SP,HL  ; point at the screen.
        PUSH BC   ; row 0

        INC H
        LD SP,HL
        PUSH DE   ; row 1

        INC H
        LD SP,HL
        EX AF,AF'
        PUSH AF   ; row 2

        INC H
        LD SP,HL
        EXX
        PUSH BC   ; row 3
        EXX

        INC H
        LD SP,HL
        EXX
        PUSH DE   ; row 4
        EXX

        INC H
        LD SP,HL
        EXX
        PUSH HL   ; ROW 5
        EXX

        ; We're empty. Time to load up again.

        LD SP,IX
        POP BC    ; ROW 6
        POP DE    ; ROW 7
        EX AF,AF'
        POP AF    ; ROW 8
        EX AF,AF'
        EXX
        POP BC    ; ROW 9
        POP DE    ; ROW 10
        POP HL    ; ROW 11
        EXX

        ; and we're loaded up again! Time to dump this graphic on the screen.

        LD IX,0
        ADD IX,SP ; save SP in IX

        INC H
        LD SP,HL
        PUSH BC   ; ROW 6

        INC H
        LD SP,HL
        PUSH DE   ; ROW 7

        DEC HL
        DEC HL

        ; Aha. Snag. We're at the bottom of a character. What's the next address down?
        ld   a,l
        and  224
        cp   224
        jp   z,ptSameThird3

ptNextThird3:
        ld   de,1760
```

```
        and  a
        sbc  hl,de
        jp ptAddrDone3

ptSameThird3:

        ld   de,32
        and  a
        adc  hl,de

ptAddrDone3:

         INC HL
         INC HL

         LD SP,HL
         EX AF,AF'
         PUSH AF  ; ROW 8

         INC H
         LD SP,HL
         EXX
         PUSH BC  ; ROW 9
         EXX

         INC H
         LD SP,HL
         EXX
         PUSH DE  ; ROW 10
         EXX

         INC H
         LD SP,HL
         EXX
         PUSH HL  ; ROW 11
         EXX

        ; Okay. Registers empty. Reload time!
        LD SP,IX
        POP BC    ; ROW 12
        POP DE    ; ROW 13

        EXX
        POP BC    ; ROW 14
        POP DE    ; ROW 15
        POP HL    ; Top Attrs
        EXX

        EX AF,AF'
        POP AF    ; Bottom Attrs
        EX AF,AF'

        ; and the last dump to screen

        INC H
        LD SP,HL
        PUSH BC

        INC H
        LD SP,HL
        PUSH DE

        INC H
        LD SP,HL
        EXX
        PUSH BC
        EXX

        INC H
        LD SP,HL
        EXX
        PUSH DE
```

```
        EXX

        ; Pixels done. Just need to do the attributes.
        ; So set HL to the attr address:

        ld      a,IYL        ;ypos

        rrca
        rrca
        rrca                ; Multiply by 32
        ld      l,a         ; Pass to L
        and     3           ; Mask with 00000011
        add     a,88        ; 88 * 256 = 22528 - start of attributes.
        ld      h,a         ; Put it in the High Byte
        ld      a,l         ; We get y value *32
        and     224         ; Mask with 11100000
        ld      l,a         ; Put it in L
        ld      a,IYH       ; xpos
        adc     a,l         ; Add it to the Low byte
        ld      l,a         ; Put it back in L, and we're done. HL=Address.
        INC HL              ; we need to be to the right of the ATTR point as pushes write backwards.
        INC HL

        ; attr
        LD SP,HL
        EXX
        PUSH HL             ; top row
        EXX

        LD HL,34            ; we need to move down to the next row. We already backed up 2, so we add
        ADD HL,SP
        LD SP,HL
        EX AF,AF'           ; bottom row
        PUSH AF

ptNextSprite2:
        ; done. Cleanup.
        LD SP,(ptstackSave) ;  put our stack back together.

        ; done all 4 final clean up

        POP IY
        POP IX

        POP AF  ; recover interrupt status
        JP PO, pt_nointerrupts
        EI      ; Okay. We put everything back. If you need interrupts, you can go with em.

pt_nointerrupts:
END ASM
END SUB
```

Usage

Example:

```
putTile (10,10,@sprite)
```

Will copy a tile of data to print position 10,10 from address at label sprite.