Putchars.bas

Putchars

> **WARNING: This subroutine does not check to see if it's writing over the edge of the screen. This is done for speed, but it is the user's job to make sure that all data will fit on the screen!**

Usage

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

```
putChars(x as uByte,y as uByte, width as uByte, height as uByte, dataAddress as uInteger)
```

Where

- x is the x value in character co-ordinates
- y is the y value in character co-ordinates
- width is the width in characters
- height is the height in characters
- dataaddress is the memory address of the UDG style bytes for the character being printed. 8 Bytes to a character. The order is top left to bottom right, first column, then second column, and so forth.

Prints the graphics data to the screen at the given character co-ordinates.

```
SUB putChars(x as uByte,y as uByte, width as uByte, height as uByte, dataAddress as uInteger)
' Copyleft Britlion. Feel free to use as you will. Please attribute me if you use this, however!

Asm
    BLPutChar:
             LD      a,(IX+5)
             ;AND     31
             ld      l,a
             ld      a,(IX+7) ; Y value
             ld      d,a
             AND     24
             add     a,64 ; 256 byte "page" for screen - 256*64=16384. Change this if you are working
             ld      h,a
             ld      a,d
             AND     7
             rrca
             rrca
             rrca
             OR      l
             ld      l,a

    PUSH HL ; save our address

    LD E,(IX+12) ; data address
    LD D,(IX+13)
    LD B,(IX+9) ; width
    PUSH BC ; save our column count

    BLPutCharColumnLoop:

    LD B,(IX+11) ; height

    BLPutCharInColumnLoop:

    ; gets screen address in HL, and bytes address in DE. Copies the 8 bytes to the screen
    ld a,(DE) ; First Row
    LD (HL),a

    INC DE
    INC H
    ld a,(DE)
    LD (HL),a ; second Row

    INC DE
    INC H
    ld a,(DE)
    LD (HL),a ; Third Row

    INC DE
    INC H
    ld a,(DE)
    LD (HL),a ; Fourth Row

    INC DE
    INC H
    ld a,(DE)
    LD (HL),a ; Fifth Row

    INC DE
    INC H
    ld a,(DE)
    LD (HL),a ; Sixth Row

    INC DE
    INC H
    ld a,(DE)
    LD (HL),a ; Seventh Row

    INC DE
    INC H
    ld a,(DE)
    LD (HL),a ; Eigth Row
```

v: latest

```
    INC DE ; Move to next data item.

    DEC B
    JR Z,BLPutCharNextColumn
    ;The following code calculates the address of the next line down below current HL address.
    PUSH DE ; save DE
             ld   a,l
             and  224
             cp   224
             jp   z,BLPutCharNextThird

    BLPutCharSameThird:
             ld   de,-1760
             ;and  a
             add  hl,de
             POP DE ; get our data point back.
             jp BLPutCharInColumnLoop

    BLPutCharNextThird:
             ld   de,32
             ;and  a
             add  hl,de
             POP DE ; get our data point back.
    JP BLPutCharInColumnLoop

    BLPutCharNextColumn:
    POP BC
    POP HL
    DEC B
    JP Z, BLPutCharsEnd

    INC L   ; Note this would normally be Increase HL - but block painting should never need to increa
    PUSH HL
    PUSH BC
    JP BLPutCharColumnLoop


BLPutCharsEnd:

End Asm
END SUB
```

Paint

Prints the colour data to the screen at the given character co-ordinates.

Syntax

```
paint (x as uByte,y as uByte, width as uByte, height as uByte, attribute as ubyte)
```

Where * x is the x value in character co-ordinates * y is the y value in character co-ordinates * width is the width in characters * height is the height in characters * attribute is the byte value of the attribute to paint to the given co-ordinates. (As one would get from the ATTR function)

Usage

v: latest

There is a an example program after the source code.

```
SUB paint (x as uByte,y as uByte, width as uByte, height as uByte, attribute as ubyte)
REM Copyleft Britlion. Feel free to use as you will. Please attribute me if you use this, however!

Asm
    ld      a,(IX+7)   ;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. Change this if you are working with a
    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,(IX+5)   ; xpos
    add     a,l        ; Add it to the Low byte
    ld      l,a        ; Put it back in L, and we're done. HL=Address.

    push HL            ; save address
    LD A, (IX+13)      ; attribute
    LD DE,32
    LD c,(IX+11)       ; height

    BLPaintHeightLoop:
    LD b,(IX+9)        ; width

    BLPaintWidthLoop:
    LD (HL),a          ; paint a character
    INC L              ; Move to the right (Note that we only would have to inc H if we are crossing t
    DJNZ BLPaintWidthLoop

    BLPaintWidthExitLoop:
    POP HL             ; recover our left edge
    DEC C
    JR Z, BLPaintHeightExitLoop

    ADD HL,DE          ; move 32 down
    PUSH HL            ; save it again
    JP BLPaintHeightLoop

    BLPaintHeightExitLoop:
end asm
END SUB
```

PaintData

Copies the colour data to the screen at the given character co-ordinates. The order here is Rows and then Columns; so first row, then second row and so on. While this may be awkward, being the other way around to the pixel data, these orders are the most efficient speedwise.

Where * x is the x value in character co-ordinates * y is the y value in character co-ordinates * width is the width in characters * height is the height in characters * address is the address of the data to copy to the screen's attribute area.

Usage

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

v: latest

```
paintData (x as uByte,y as uByte, width as uByte, height as uByte, address as uInteger)
```

```
SUB paintData (x as uByte,y as uByte, width as uByte, height as uByte, address as uInteger)
REM Copyleft Britlion. Feel free to use as you will. Please attribute me if you use this, however!

Asm
    ld      a,(IX+7)   ;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. Change this if you are working with a
    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,(IX+5)   ; xpos
    add     a,l        ; Add it to the Low byte
    ld      l,a        ; Put it back in L, and we're done. HL=Address.

    push HL            ; save address
    LD D, (IX+13)
    LD E, (IX+12)
    LD c,(IX+11)       ; height

    BLPaintDataHeightLoop:
    LD b,(IX+9)        ; width

    BLPaintDataWidthLoop:
    LD a,(DE)
    LD (HL),a          ; paint a character
    INC L              ; Move to the right (Note that we only would have to inc H if we are crossing t
    INC DE
    DJNZ BLPaintDataWidthLoop

    BLPaintDataWidthExitLoop:
    POP HL             ; recover our left edge
    DEC C
    JR Z, BLPaintDataHeightExitLoop
    PUSH DE
    LD DE,32
    ADD HL,DE          ; move 32 down
    POP DE
    PUSH HL            ; save it again
    JP BLPaintDataHeightLoop

    BLPaintDataHeightExitLoop:
End Asm
END SUB
```

Example Program

v: latest

```
goto start

datapoint:
Asm
    defb 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32
    defb 33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63
    defb 65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95
    defb 97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120
End Asm

start:
cls
putChars(10,10,3,3,@datapoint)
paint(10,10,3,3,79)
```

v: latest