WindowPaint

paintWindow

> **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!**

This subroutine changes the attribute map without actually changing the bitmap that's in it. You can combine this with other routines, such as clearbox, to clear a screen area and reset the attributes, as well as fast plot and draw routines that don't deal with attributes themselves. Also ideal (and originally designed for) use with `putChars`, which is a fast graphics print routine, that also doesn't do attributes directly. Sprites can be worked up from this basis.

Usage

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

`windowPaint` calls paint with the required single attribute byte - it's perfectly reasonable to call it directly, if you have the full attribute value ready. windowPaint is really there to make it simpler to construct this byte.

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

```
SUB windowPaint(x as uByte,y as uByte, width as uByte, height as uByte, inkCol as ubyte, paperCol as u
 paint(x,y,width,height,(isFlash<<7+isBright<<6+paperCol<<3+inkCol))
END SUB


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
```