PropPrint.bas

This proportional printing routine is ideal for making text more readable. Do NOT use it for tables - letter positions move around within the lines depending what the text is. For tables, use a fixed width routine - either the system 32 chars per line, or the 42 chars per line version listed in the library.

usage

```
propPrint(xPosition, yPosition, SizeOfSpaceCharacter, GapBetweenLetters, "Text")
```

> Note: X and Y are pixel positions. Note that unlike Print and Print42, this routine in in the form (x,y), not (y,x). That can easily be changed in the declaration below. [Just swap x and y].

`SizeOfSpaceCharacter` is the width of a space. `GapBetweenLetters` can be as small as zero - though the letters crash into each other, so it doesn't look wonderful. 1 pixel is probably the most common setting for this. Text is anything that's valid as a string - either a `String` literal, or a variable.

```
' Proportional printing anywhere on screen.
' Original routine by Christoph Odenthal (Odin)
' ZXBC modification by Leszek Chmielewski (LCD)
' Thanks to Boriel and Britlion for help and hints

sub propPrint(x as ubyte,y as ubyte,spacesize as ubyte, xgap as ubyte,txt$ as string)
dim b$ as string '
b$=txt$+chr(0)
poke Uinteger @PropPrintTxtadr,PEEK(Uinteger, @b$)+2 'textadr
poke @PropPrintTxtadr+2,x
poke @PropPrintTxtadr+3,y
poke @PropPrintTxtadr+4,spacesize
poke @PropPrintTxtadr+5,xgap
PropPrint:
Asm
;' --------------------------------
;'  Free Size N Place Text Print
;' --------------------------------
;' (c) 14.09.2002 C.Odenthal
;'
;' Last Modifications: 15.10.2002
;' Modification for Boriels ZXBC
;' By Leszek Chmielewski 02.06.2010
;' --------------------------------
;' ORG 32000
;' DUMP 45000
;' --------------------------------
;' Entry point
;' --------------------------------
Start:
   JR Start2
;' --------------------------------
;' Parameter
;' --------------------------------
end asm
PropPrintTxtadr:
asm
text_addr:
   DEFW 0 ;' Addr of text
x_pos:
   DEFB 4 ;' Pos of text
y_pos:
   DEFB 4
spc_size:
   DEFB 3 ;' Width of space char
x_gap:
   DEFB 1 ;' Gap between chars
y_gap:
   DEFB 1 ;' Gap between lines
end_symb:
   DEFB 0 ;' End of line char code
x_start:
   DEFB 0 ;' Start of window
y_start:
   DEFB 0
x_end:
   DEFB 255 ;' End of window
y_end:
   DEFB 191
x_zoom:
   DEFB 1 ;' Zooming factor
y_zoom:
   DEFB 1

;' --------------------------------
;' Main routine
;' --------------------------------

Start2:
   LD A,(x_pos) ;' Calc. scr addr.
   LD B,A
   LD A,(y_pos)
```

```
   LD C,A
   CALL PosToScr
   LD (scr_ad),DE ;' Store scr addr.
   LD A,L
   LD (pix_pos),A
   LD A,(end_symb) ;' Get line ending char
   LD (poke_here+1),A ;' Poke it into memory below
   LD HL,(text_addr)
Main_Loop:
   LD A,(HL);' Get char
poke_here:
   CP 0 ;' End of text? (poked!)
   JP Z,Exit1
   INC HL
   PUSH HL
   CP 32 ;' Replace ctrl chars
   JP NC,No_Ctrl
   LD A,32
No_Ctrl:
   CALL Copy_Chr ;' Copy char-gfx to work-buffer
   CALL Measure ;' Measure left rim + width
   PUSH BC ;' (Save results)
   LD A,C ;' No pixel set in char?
   AND A
   JP Z,Space_Chr
   LD A,(pix_pos);' size+bitpos (=1..15)
   ADD A,C
   CP 9 ;' > 8 ?
   JP NC,Overlap ;' Char overlaps
;' Calculate 8 bit rotation
   LD A,(pix_pos) ;' bitpos-l_rim (=-7..7)
   SUB B
   AND A ;' = 0 ?
   JP Z,PrintIt8
   JP C,Neg_8 ;' < 0 ?
;' --------------------------------
   CP 5 ;' Not in 1..4 ?
   JP NC,Left_8
;' --------------------------------
   LD B,A
   CALL Chr_Rgt8 ;' Rotate right 8 bit
   JP PrintIt8
;' --------------------------------
Left_8:
   NEG ;' = 8 - A
   ADD A,8
   LD B,A
   CALL Chr_Left8 ;' Rotate left 8 bit
   JP PrintIt8
;' --------------------------------
Neg_8:
   NEG
   CP 5 ;' Not in 1..4 ?
   JP NC,Right_8
;' --------------------------------
   LD B,A
   CALL Chr_Left8 ;' Rotate left 8 bit
   JP PrintIt8
;' --------------------------------
Right_8:
   NEG ;' = 8 - A
   ADD A,8
   LD B,A
   CALL Chr_Rgt8 ;' Rotate right 8 bit
   JP PrintIt8
;' --------------------------------
;' Calculate 16 bit rotation
;' --------------------------------
Overlap:
   LD A,(pix_pos);' bitpos-l_rim (=-7..7)
   SUB B
   AND A ;' = 0 ?
```

```
   JP Z,PrintIt8
;' ------------------------------
   LD B,A
   CALL Chr_Rgt16 ;' Rotate right 16 bit
PrintIt16:
   LD DE,(scr_ad);' Check for screen end
   LD A,E
   AND 31
   CP 31 ;' (2nd byte outside ?)
   JP Z,Outside
   CALL Print16 ;' Display char (16 bit)
   JP Next_Loc
;' ------------------------------
PrintIt8:
   LD DE,(scr_ad);' Display char (8 bit)
   CALL Print8
   JP Next_Loc
;' ------------------------------
Space_Chr:
   LD A,(spc_size);' Skip pixels
   LD DE,(scr_ad);' Get screen addr.
   POP BC ;' Throw away values
   LD B,0
   LD C,A
   JP Next_Loc2
;' ------------------------------
Next_Loc:
   POP BC ;' Move to next char location
Next_Loc2:
   LD A,(pix_pos);' =bitpos+x_gap+size
   LD L,A
   LD A,(x_gap)
   ADD A,L
   ADD A ,C
   LD L ,A
   AND 7 ;' New pix_pos
   LD (pix_pos),A
   LD H,0 ;' =result/8
   SRL L
   SRL L
   SRL L
   ADD HL,DE ;' New byte pos
   LD A ,E ;' Check for screen end
   AND 31
   LD E,A
   LD A,L
   AND 31
   CP E ;' New pos smaller than old ?
   JP C ,Exit2 ;' -> End of printing
   LD (scr_ad),HL ;' Store new scr ad.
   POP HL ;' Restore text pointer
   JP Main_Loop
;' ------------------------------
Outside:
   POP BC ;' Stop printing
   POP HL ;' Char not printed!
   DEC HL
   JP Exit1
;' ------------------------------
Exit2:
   POP HL ;' Return nr of printed chars
Exit1:
   LD DE,(text_addr)
   XOR A
   SBC HL,DE
   LD B,H ;' Return value in BC to Basic
   LD C,L
    jp PropPrintTxtadr2
;' ------------------------------
;' Calc. scr adr from x,y
;' ------------------------------
;' In : B  = x / C = y (preserved)
```


/

```
;' Out: DE = scr adr / L = pixpos
;' Usd: A, BC, DE, L
;' -------------------------------
PosToScr:
   LD A,C ;' Range check
   CP 185
   JP C,ValOk
   LD C,184
ValOk:
   LD A,B ;' Pix pos
   AND 7
   LD L,A
   LD E,B
   SRL E
   SRL E
   SRL E
   LD A,C ;' Scr pos
   AND 7
   LD D,A
   LD A,C
   AND 56
   RLA
   RLA
   OR E
   LD E,A
   LD A,C
   AND 192
   RRA
   RRA
   RRA
   OR D
   OR 64
   LD D,A
   RET
;' -------------------------------
;' Copy char into buffer
;' -------------------------------
;' In : A = Char
;' Out: -
;' Usd: A, HL, DE, BC
;' -------------------------------
Copy_Chr:
   LD DE,(23606);' SysVar CHARS
   LD H,0
   LD L,A
   ADD HL,HL ;' * 8
   ADD HL,HL
   ADD HL,HL
   ADD HL,DE ;' + Chartable
   EX DE,HL
   LD HL,Chr_Buf
   LD B,8
Copy_Loop:
   LD A,(DE);' Double to 16 pixel/row
   INC DE
   LD (HL),A ;' Low-byte in memory!
   INC HL
   LD (HL),0 ;' High-byte in memory!
   INC HL
   DJNZ Copy_Loop
   RET
;' -------------------------------
;' Measure left border and width
;' -------------------------------
;' In : -
;' Out: B = left rim / C = width
;' Usd: A, HL, BC
;' -------------------------------
Measure:
   LD HL,Chr_Buf ;' "OR" together all 8 bytes
   LD B,8
   XOR A
```


/

```
Msr_Loop:
   OR (HL)
   INC HL
   INC HL
   DJNZ Msr_Loop
   LD BC,0
   AND A ;' Check if zero
   RET Z
Msr_Loop2:
   INC B ;' Measure left border
   RLCA
   JP NC,Msr_Loop2
   RRCA
   DEC B
   LD C,9 ;' Measure width
Msr_Loop3:
   DEC C
   RRCA
   JP NC,Msr_Loop3
   RET
;' --------------------------------
;' Move char to left, 8 bit
;' --------------------------------
;' In : B = Nr of bits to shift
;' Out: -
;' Usd: A, B, HL
;' --------------------------------
Chr_Left8:
   PUSH BC
   LD HL,Chr_Buf ;' Rotate char left 8 bit
   LD C,B ;' 1st row
   LD A,(HL )
Chr_LLp1:
   RLCA
   DJNZ Chr_LLp1
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 2nd row
   LD A,(HL)
Chr_LLp2:
   RLCA
   DJNZ Chr_LLp2
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 3rd row
   LD A ,(HL)
Chr_LLp3:
   RLCA
   DJNZ Chr_LLp3
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 4th row
   LD A,(HL)
Chr_LLp4:
   RLCA
   DJNZ Chr_LLp4
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 5th row
   LD A,(HL)
Chr_LLp5:
   RLCA
   DJNZ Chr_LLp5
   LD (HL),A
```


/

```
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 6th row
   LD A,(HL)
Chr_LLp6:
   RLCA
   DJNZ Chr_LLp6
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 7th row
   LD A,(HL)
Chr_LLp7:
   RLCA
   DJNZ Chr_LLp7
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 8th row
   LD A,(HL)
Chr_LLp8:
   RLCA
   DJNZ Chr_LLp8
   LD (HL),A
   LD B,C
   POP BC
   RET
;' --------------------------------
;' Move char to right, 8 bit
;' --------------------------------
;' In : B = Nr of bits to shift
;' Out: -
;' Usd: A, B, HL
;' --------------------------------
Chr_Rgt8:
   PUSH BC
   LD HL,Chr_Buf ;' Rotate char right 8 bit
   LD C,B ;' 1st row
   LD A,(HL)
Chr_RLp1:
   RRCA
   DJNZ Chr_RLp1
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 2nd row
   LD A,(HL)
Chr_RLp2:
   RRCA
   DJNZ Chr_RLp2
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 3rd row
   LD A,(HL)
Chr_RLp3:
   RRCA
   DJNZ Chr_RLp3
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 4th row
   LD A,(HL)
Chr_RLp4:
   RRCA
   DJNZ Chr_RLp4
```


/

```
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 5th row
   LD A,(HL)
Chr_RLp5:
   RRCA
   DJNZ Chr_RLp5
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 6th row
   LD A ,(HL )
Chr_RLp6:
   RRCA
   DJNZ Chr_RLp6
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 7th row
   LD A,(HL)
Chr_RLp7:
   RRCA
   DJNZ Chr_RLp7
   LD (HL),A
   LD B,C
   INC HL
   INC HL
   LD C,B ;' 8th row
   LD A,(HL)
Chr_RLp8:
   RRCA
   DJNZ Chr_RLp8
   LD (HL),A
   LD B,C
   POP BC
   RET
;' --------------------------------
;' Move char to right 16 bit
;' --------------------------------
;' In : B = Nr of bits to shift
;' Out:
;' Usd: A, B, HL
;' --------------------------------
Chr_Rgt16:
   LD HL,(Chr_Buf);' Rotate char right 16 bit
   LD A,B ;' 1st row
Chr_R2Lp1:
   SRL L
   RR H ;' Insert carry
   DJNZ Chr_R2Lp1
   LD B,A
   LD (Chr_Buf),HL
   LD HL,(Chr_Buf+2);' 2nd row
   LD A,B
Chr_R2Lp2:
   SRL L
   RR H ;' Insert carry
   DJNZ Chr_R2Lp2
   LD B,A
   LD (Chr_Buf+2),HL
   LD HL,(Chr_Buf+4);' 3rd row
   LD A,B
Chr_R2Lp3:
   SRL L
   RR H ;' Insert carry
   DJNZ Chr_R2Lp3
   LD B,A
   LD (Chr_Buf+4),HL
```


/

```
   LD HL,(Chr_Buf+6);' 4th row
   LD A,B
Chr_R2Lp4:
   SRL L
   RR H ;' Insert carry
   DJNZ Chr_R2Lp4
   LD B,A
   LD (Chr_Buf+6),HL
   LD HL,(Chr_Buf+8);' 5th row
   LD A,B
Chr_R2Lp5:
   SRL L
   RR H ;' Insert carry
   DJNZ Chr_R2Lp5
   LD B,A
   LD (Chr_Buf+8),HL
   LD HL,(Chr_Buf+10);' 6th row
   LD A,B
Chr_R2Lp6:
   SRL L
   RR H ;' Insert carry
   DJNZ Chr_R2Lp6
   LD B,A
   LD (Chr_Buf+10),HL
   LD HL,(Chr_Buf+12);' 7th row
   LD A,B
Chr_R2Lp7:
   SRL L
   RR H ;' Insert carry
   DJNZ Chr_R2Lp7
   LD B,A
   LD (Chr_Buf+12),HL
   LD HL,(Chr_Buf+14);' 8th row
   LD A,B
Chr_R2Lp8:
   SRL L
   RR H ;'Insert carry
   DJNZ Chr_R2Lp8
   LD B,A
   LD (Chr_Buf+14),HL
   RET
;' --------------------------------
;' Print 8 bit wide char on screen
;' --------------------------------
;' In : DE = screen adr.
;' Out: -
;' Usd: A, HL, DE, B
;' --------------------------------
Print8:
   LD HL,Chr_Buf
   PUSH DE ;' save scr ad.
   EX DE,HL
   LD B,8 ;' 8 lines
Prt8_L:
   LD A,(DE);' set 1 byte
   XOR (HL)
   LD (HL),A
   INC DE ;' skip 1 byte
   INC DE ;' next byte
   INC H ;' calc. next line
   LD A,H
   AND 7
   JP NZ,Prt8_C
   LD A,L
   ADD A,32
   LD L,A
   JR C,Prt8_C
   LD A,H
   SUB 8
   LD H,A
Prt8_C:
   DJNZ Prt8_L ;' next round
```

```
   POP DE ;' restore scr ad.
   RET
;'-------------------------------
;' Print 16 bit wide char on screen
;' -------------------------------
;' In : DE = screen adr.
;' Out: -
;' Usd: A, HL, DE, B
;' -------------------------------
Print16:
   LD HL,Chr_Buf
   PUSH DE ;' save scr ad.
   EX DE,HL
   LD B,8 ;' 8 lines
Prt16_L:
   LD A,(DE);' set 1 byte
   XOR (HL)
   LD (HL),A
   INC HL ;' next scr pos
   INC DE ;' next byte
   LD A,(DE);' set 1 byte
   XOR (HL)
   LD (HL),A
   DEC HL ;' prev scr pos
   INC DE ;' next byte
   INC H ;' calc. next line
   LD A,H
   AND 7
   JP NZ,Prt16_C
   LD A,L
   ADD A,32
   LD L,A
   JR C,Prt16_C
   LD A,H
   SUB 8
   LD H,A
Prt16_C:
   DJNZ Prt16_L ;' next round
   POP DE ;' restore scr ad.
   RET
;' -------------------------------
;'  Variables
;' -------------------------------
Chr_Buf:
   DEFS 16
scr_ad:
   DEFW 0
pix_pos:
   DEFB 0
;' -----------------
PropPrintTxtadr2:
end asm
end sub
```