DoubleSizePrint.bas

This routine will print out Strings at double size. It will stop if it reaches the end of the screen, however - so you have to deal with new rows on your own! (143 bytes long for main routine)

The core subroutine is capable of printing one character on screen at specified location, in double size. Of course, this is the basis for the second subroutine, that takes a string and prints it all.

Both routines are capable of printing from the User Definable Graphics locations.

Usage:

```
REM Requires ascii code of character to print.
doubleSizePrintChar(y, x, code("a"))
```

```
REM This version calls the first one, and will print a whole string.
doubleSizePrint(y, x, thingToPrint$)
```

Code:

```
SUB doubleSizePrintChar(y as uByte, x as uByte, thingToPrint as uByte)
' Prints a single character double sized.
' Takes X and Y values as character positions, like print.
' takes an ascii code value for a character.
' By Britlion, 2012.

ASM
LD A,(IX+5)     ;' Y value
CP 22
JP NC, doubleSizePrintCharEnd

;' A=y value
LD E,A
AND 24           ; calculate
OR 64            ; screen
LD H,A           ; address
LD A,E           ; for
AND 7            ; row
OR a             ; Y
RRA
RRA
RRA
RRA
LD E,A

LD A,(IX+7)     ;' X Value
CP 30
JP NC, doubleSizePrintCharEnd

ADD A,E         ;' correct address for column value. (add it in)
LD L,A
EX DE,HL        ;' Save it in DE

LD A,(IX+9)     ;'Character

CP 164      ;' > UDG "U" ?
JP NC, doubleSizePrintCharEnd

CP 32       ;' < space+1?
JP C, doubleSizePrintCharEnd

CP 144      ;' >144?
JP NC, doubleSizePrintCharUDGAddress

LD L,A
LD H,0

ADD HL,HL
ADD HL,HL
ADD HL,HL   ;' multiply by 8.
LD BC,(23606)   ;' Chars
ADD HL,BC   ;' Hl -> Character data.
EX DE,HL    ;' DE -> character data, HL-> screen address.
JP doubleSizePrintCharRotateLoopCharStart

doubleSizePrintCharUDGAddress:
LD HL,(23675)    ;'UDG address
SUB 144
ADD A,A         ;multiply by 8.
ADD A,A
ADD A,A
ADD A,L
LD L,A

JR NC, doubleSizePrintCharUDGAddressNoCarry
INC H
doubleSizePrintCharUDGAddressNoCarry:

;' At this point HL -> Character data in UDG block.
EX DE,HL ;' DE -> character data, HL-> screen address.

doubleSizePrintCharRotateLoopCharStart:
```

```
LD C,2 ;' 2 character rows.
doubleSizePrintCharRotateLoopCharRowLoopOuter:
LD b,4 ;' 4 source bytes to count through per character row.
doubleSizePrintCharRotateLoopCharRowLoopInner:
   PUSH BC

   LD A,(DE) ;' Grab a bitmap.
   PUSH DE

   LD B,4
   LD C,A ; Copy byte so we can put two into the big version.
   doubleSizePrintCharRotateLoop1:
      RRA  ; one bit into carry
      RR E ; one bit into result
      RR C ; same bit into carry again
      RR E ; duplicated bit into result
   DJNZ doubleSizePrintCharRotateLoop1

   LD B,4
   doubleSizePrintCharRotateLoop2:
       RRA
       RR D ; Other register for other half of big 16 bit line.
       RR C
       RR D
   DJNZ doubleSizePrintCharRotateLoop2

   LD (HL),D    ;' Output first byte
   INC HL   ;' Move right
   LD (HL),E    ;' Second half.
   DEC HL   ;' Move left
   INC H    ;' Move down
   LD (HL),D    ;' Output second row (copy of first), first byte.
   INC HL   ;' Move right
   LD (HL),E    ; Output second row, second byte
   DEC HL   ; Move left
   INC H    ; Move down.
   POP DE
   INC DE
   POP BC

DJNZ doubleSizePrintCharRotateLoopCharRowLoopInner
; CALL __DECY+2     ;'Jump into the DRAW next_line_down routine, at a convenient point (accounting for
; Can't seem to call to this at the moment! Here in longhand form:

ld a, h
and 7
jr nz, doubleSizePrintCharRotateNextCharRow
ld a, l
add a, 32
ld l, a
jr c, doubleSizePrintCharRotateNextCharRow
ld a, h
sub 8
ld h, a

doubleSizePrintCharRotateNextCharRow:

DEC C
JR NZ, doubleSizePrintCharRotateLoopCharRowLoopOuter

doubleSizePrintCharEnd:
END ASM
END SUB
```

```
SUB doubleSizePrint(y as uByte, x as uByte, thingToPrint$ as String)
'Uses doubleSizePrintChar subroutine to print a string.
'By Britlion, 2012

   DIM n as uByte
   for n=0 to LEN thingToPrint - 1
      doubleSizePrintChar(y,x,CODE thingToPrint$(n) )
      x=x+2
   next n

END SUB
```

Example:

```
cls
doubleSizePrintChar(0,0,145)
doubleSizePrint(10,0,"Hello World")
```