Picture this

Christmas comes early as Auntie John Kennedy shows how 8K can be as good as 16K.

How would you like to halve the size of your screen files? No fancy data compression routines, just a guaranteed 8K screen file? Well of course this is completely impossible. Or is it?

If you use a lot of digitised pictures then perhaps you are in luck. Digitised pictures are so full of little bits of detail that losing half of them will still make next to no difference. If you don't believe me, take a look at the two wonderful pictures of my good self. One image takes 16K and the other takes 8K. You should be able to tell them apart – now hold them at arm's length. Can you still see 8K worth of difference? Remember, although half the information is missing, the handsome chap is still instantly recognisable.

All that has happened is that of the two hundred lines of screen data, only one hundred have been stored and the other hundred just copied from the line above. Simple, but effective. And the machine code program to Create and Display the data is very simple indeed. The routine HALFSCREEN will copy half the screen data to address &4000, so to save it after compression just type: SAVE "SMALLSCR",b,&40000,&2000

To reconstruct a screen image, the routine DRAWSCREEN should be called after placing some suitable data at address &4000.

There are only a few tricky bits in the program so far. You must remember to switch the upper ROM off if you are looking at the screen memory which occupies the area underneath. It is also good manners to switch this ROM on again after we have played with it. Once again, we make the little Z80 instruction LDIR to do all the work. LDIR must rate as my favourite instruction of all time. It performs the incredibly useful task of copying memory from one address to another, and works thus:

Load DE with the address where the data is to go,

Load HL with the address where the data is already,

Load BC with the number of bytes to be moved.

For example,

```
LD DE,&C000
LD HL,&4000
Ld BC,&0005
LDIR
```

will move five bytes from &4000 to &c000

```
nolist
;
; Screen Compressor for Digitised Pictures
;
; Copyright Auntie John for Amstrad Computer User
;
; Routines:
;
; HALFSCREEN
; Take a normal screen display and store half of it from address
; &4000 and onwards.
;
; DRAWSCREEN
; Take half a screen stored starting at &4000 and re-draw it.
;
; SYNTHSCREEN
; Take half a screen stored starting at &4000 and synthesize a
; full image from it.
;
;

prev_line equ &bc29
next_line equ &bc26

Upper_rom_off equ &b903
Upper_rom_on equ &b900

Top_of_screen equ &c000
Store_address equ &4000

org &3000                ;Start of code - must be before &4000

.HALFSCREEN
        call Upper_rom_off              ;Turn off top ROM to make sure
                                        ;we examine the screen RAM
        ld hl,Top_of_screen             ;Top of Screen Ram
        ld de,Store_address             ;Start of storage

        ld b,100                        ;Half the 200 screen lines
.loop1  push hl:push bc                 ;Preserve registers
        ld bc,80                        ;Move 80 bytes
        ldir
        pop bc
        pop hl:call nexthl:call nexthl
        djnz loop1                      ;Repeat for all lines

        call Upper_rom_on               ;Turn on Rom

        ret


.DRAWSCREEN
        ld de,Top_of_screen             ;Top of Screen RAM
        ld hl,Store_address             ;Start of screen store
        ld b,99
.loop2  push bc:push de                 ;Preserve registers
        ld bc,80:ldir                   ;Draw first line
        pop de:call nextde
        push hl:push de                 ;Draw it again, but down
        ld bc,80:ldir                   ;one screen line.
        pop de:pop hl:pop bc
        call nextde
        djnz loop2
        ret


.SYNTHSCREEN

;Stage One - Draw every other line

        ld de,Top_of_screen             ;Top of Screen RAM
        ld hl,Store_address             ;Start of screen store
        ld b,99
.loop3  push bc:push de                 ;Preserve registers
        ld bc,80:ldir                   ;Draw a screen line
        pop de:pop bc
        call nextde:call nextde         ;Move down two lines
        djnz loop3

;Stage Two - Make up the data for in-between lines

        call Upper_rom_off              ;Turn off top ROM to make sure
                                        ;we examine the screen RAM

        ld de,Top_of_screen
        call nextde                     ;Move one line down from top
        ld b,98
.loop4  ld c,80
```

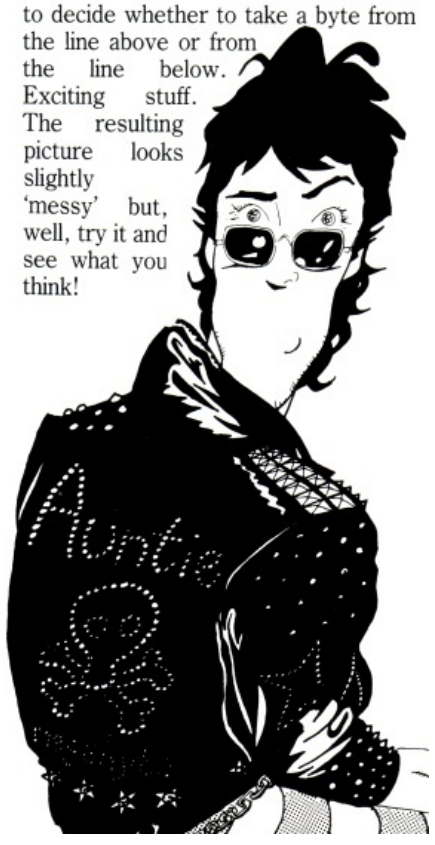
After playing around with this program three more ideas presented themselves to me. The first was why not do the same with the horizontal resolution, with a result that 4k is needed for each picture? This technique results in a definite 'de-rezzed' look, but for a quarter of the memory this trade-off will sometimes be worthwhile. The program is left to you as an exercise, because it is really not all that difficult to write.

The second idea was to combine conventional data compression techniques with the special halved and quartered images. Potentially this means the pictures could now take 4K and 2K to store. Again, Run Length

Compression is something I've written dozens of programs about, so have a go yourself. Try looking through some back issues of ACU.

The third idea was a way of creating a slightly better quality picture from a halved picture. Instead of just copying the screen line above, why not synthesize a new line, by taking elements from the line above and the line below? This seemed a fun little program to write, needing only a new PRINTHALF routine. So I had a bath to think about it, and wrote it.

It uses the magic 'r' register, which to all intents and purposes returns a random number. We use this number to decide whether to take a byte from the line above or from the line below. Exciting stuff. The resulting picture looks slightly 'messy' but, well, try it and see what you think!



```
.loop5  push bc

        push de:pop hl

        ld a,r                          ;Sneak a random number
        bit 2,a                         ;Check one bit of it
        jr z,lookup
        jr nz,lookdown
.back   ld a,(hl):ld (de),a             ;Draw in byte of pixels
        inc hl:inc de                   ;Move to next byte in line

        pop bc

        dec c:ld a,c
        cp 0:jr nz,loop5
        pop de

        call nextde:call nextde         ;Move down 2 lines

        dec b:ld a,b
        cp 0:jr nz,loop4

        call Upper_rom_on               ;Turn on Rom

        ret

.lookup
        call prev_line
        jr back

.lookdown
        call next_line
        jr back

.nexthl
        call next_line
ret


 .nextde
         push hl
         ex hl,de
         call nexthl
         ex hl,de
         pop hl
ret
```

Compressed 8K Image.

Compressed 16K Image.

If you are wondering where I get my digitised pictures from, the answer lies in a little black box called VIDI made by a company called Rombo - famous for their ROM boxes. Apparently the price of this little electronic gem is coming down, so make sure you check it out. It takes a video image from a video-recorder or camera, and produces a practically instant image on your CPC. Of course, I can't mention Rombo without mentioning the delightful Verona and Leslie. Well, I could I suppose. But I won't.

Bye. ●