PRINT#-3

Theodore (Alex) Evans
varmfskii@gmail.com

November 25, 2021

Complete code for this can be found at: `https://github.com/varmfskii/coco/tree/main/print-3/code`

Extended Color BASIC supports devices -2 to 0 with -2 being the printer, -1 the cassette, and 0 being the basic text screen. With Disk BASIC adds devices 1-15 as handles to open files on the disk drive. Super Extended Color Basic on the CoCo 3 this didn't change this other than device 0 getting a set of three modes (WIDTH 32, 40, and 80) to support the old and new text modes. There have been various things done to repurpose some of these devices, most notably allowing text display on the PMODE 4 screen to allow 53, 64, or 85 column text.

I am doing something a little different. I am adding device number -3 to support a new serial channel. The method is demonstrative, and not particularly practical, though someone may find a use for the particular case. Some potential uses for this could be an easy way to simultaneously use the second display provided by hardware like the Wordpak 2+ or SuperSprite FM+ which still having easy access to the original text screen.

There are two common ways to modify/extend the Coco's BASIC. You can patch the 25 vectors that run through addresses $015E to $01A8 or, on a machine with 64k or more RAM, directly patch BASIC. I will do a combination of the two.

For all of the vector code.

If we have device -3:

- Remove one return address from the stack to inhibit BASIC's handling of the vector.
- Perform our processing for device -3
- Return

If we do not have device -3, we call the original code.

Things to do:

- Change device 0 to not use all of screen.
- Validate device number: RVEC1

- Set print parameters: RVEC2
- Console out: RVEC3

1 Modify device 0

Change device 0 to not use all of screen: Printing on device 0 uses a routing for PRINT@ which has no hook, so for this part we are going to modify BASIC directly. We are patching where the screen ends.

To do this first make sure the machine is in all RAM mode with the contents of the ROMS copied to ram.

```
allram:
        ;;  check if already in all RAM mode
        lda $8000
        tfr a,b
        com $8000
        eora $8000
        beq rom
        stb $8000
        rts

        ;; copy ROMs to RAM
rom:
        pshs cc
        orcc #$50               ; disable interrupts
        sts $fe
        lds #$8000
loop@:
        clr RAMROM
        puls a,b,x,y,u
        clr RAMRAM
        pshs u,y,x,a,b
        leas 8,s
        cmps #$ff00
        ble loop@
        lds $fe
        puls cc,pc
```

Then perform the actual patches.

```
print0:
        ;; modify PRINT@ limit
        ldx #255
        stx $a558 ; limit
        leax VIDRAM,x ; adjust back to memory location
        stx $a55f ; adjusts
```

```
        ;; modify CLS end
        stx $a932 ; cls end
        ;; modify CHROUT limit
        stx $a347 ; memory location limit
        leax -31,x ; beginning of last line
        stx $a354 ; scroll limit
        ;; set cursor at top of screen
        ldx #VIDRAM
        stx CURPOS
        rts
```

If we don't need to change the existing device, we can add our vectors without using all RAM mode. We also had the option of using some of the other vectors like the PRINT vector (RVEC9) and the CLS vector (RVEC22).

2 Valildate device number

Vector RVEC1 takes a value in the B register and either returns on success or calls the error handler on a failure. In our case we want to check for device -3, if we have it we return success otherwise we pass control to the original routine. exrepl st

```
        ;; Validate device number
rvec1_3:
        leas 2,s
        rts
        cmpb #-3
        bne ovec1
        leas 2,s
        rts
ovec1:  rmb 3
```

3 Set print parameters

Vector RVEC2 checks what the current device is in DEVNUM and assigns the proper values to: DEVCFW (tab zone width), DEVLCF (beginning of last tab zone), DEVPOS (current column), and DEVWID (total number of columns)

```
        ;; set print parameters
rvec2_3:
        pshs a                  ;
        lda DEVNUM
        cmpa #-3
        puls a
```

```
        beq cont@
ovec2:  rmb 3
cont@:
        leas 2,s
        pshs x,b,a
        clr PRTDEV
        ;; a=current column
        lda curpos3+1
        anda #cols-1
        ;; b=number of columns
        ldb #cols
        std DEVPOS
        ;; xhi=tab size, xlo=last tab column
        ldx #$0818
        stx DEVCFW
        puls a,b,x,pc
```

4 Console out

Vector RVEC3 takes the output character in register A and prints it to the selected device.

```
        ;; output to device
rows:   equ 8
cols:   equ 32
scrstt: equ $0500
bs:     equ $08
cr:     equ $0d
space:  equ $20

curpos3:
        fdb $0500
rvec3_3:
        pshs a
        lda DEVNUM
        cmpa #-3
        puls a
        beq cont@
ovec3:  rmb 3
cont@:
        leas 2,s
        pshs x,b,a
        ldx curpos3
        cmpa #$20
```

```
        blo cntl
        cmpa #$40
        blo wrtchr
        cmpa #$80
        bhs wrtchr
        cmpa #$60
        bhs lcase
ucase:
        anda #$3f
wrtchr:
        sta ,x+
may_scroll:
        cmpx #scrstt+rows*cols
        beq scroll
save:
        stx curpos3
exit:
        puls a,b,x,pc
lcase:
        suba #$20
        bra wrtchr

scroll:
        ldx #scrstt
lp1@:
        lda cols,x
        sta ,x+
        cmpx #scrstt+(rows-1)*cols
        bne lp1@
        lda #space
        clrb
lp2@:
        sta b,x
        incb
        cmpb #cols
        bne lp2@
        bra save

cntl:
        cmpa #bs
        bne not_bs

is_bs:
        cmpx #scrstt
        beq exit
        lda #space
```

```
        sta ,-x
        bra save

not_bs:
        cmpa #cr
        bne not_cr

is_cr:
        tfr x,d
        andb #cols-1
        negb
        addb #cols
        lda #space
loop@:
        sta ,x+
        decb
        bne loop@
        bra may_scroll

not_cr:
        puls a,b,x,pc
```