Horizontal Screen Positioning (HSP) aka Wanker aka DMA-Delay

- Fix Me!: demo world records and world firsts mentions VSP+IK - there VSP is “Variable Screen Positioning” = DMA Delay = horizontal positioning. Is “HSP” common? Didn't find that on the net.

This code is for PAL machines. To make it work for NTSC you have to do some minor timing modifications for each scan-line.

The code is in ACME assembler format.

```
; This demonstrates HSP (Horizontal Screen Positioning) using a stable
raster.
; If you have any questions contact the author Martin Piper through
CodeBase64
!to "RasterTest.prg", cbm
!cpu 6510
!ct pet

ZPProcessorPortDDR          = $00
ProcessorPortDDRDefault     = %101111
ZPProcessorPort         = $01
ProcessorPortAllRAMWithIO       = %100101
CIA1InterruptControl            = $dc0d
CIA1TimerAControl           = $dc0e
CIA1TimerBControl           = $dc0f
CIA2InterruptControl            = $dd0d
CIA2TimerAControl           = $dd0e
CIA2TimerBControl           = $dd0f
VIC2InteruptControl         = $d01a
VIC2InteruptStatus          = $d019
VIC2BorderColour            = $d020
VIC2ScreenColour            = $d021
VIC2ScreenControlV          = $d011
VIC2SpriteEnable            = $d015
VIC2Raster              = $d012
CIA1KeyboardColumnJoystickA     = $dc00
KERNALNMIServiceRoutineLo       = $fffa
KERNALNMIServiceRoutineHi       = $fffb
KERNALIRQServiceRoutineLo       = $fffe
KERNALIRQServiceRoutineHi       = $ffff


!macro MACROAckRasterIRQ_A {
    lda #1
    sta VIC2InteruptStatus                ; Ack Raster interupt
}
```

```
!macro MACROAckAllIRQs_A {
    ; Ack any interrupts that might have happened from the CIAs
    lda CIA1InterruptControl
    lda CIA2InterruptControl
    ; Ack any interrupts that have happened from the VIC2
    lda #$ff
    sta VIC2InteruptStatus
}


*= $0801
!byte $0b,$08,$01,$00,$9e       ; Line 1 SYS2061
!convtab pet
!tx "2061"                        ; Address for sys start in text
!byte $00,$00,$00

!zn
.theLine = 45
    ; Stop interrupts, clear decimal mode and backup the previous stack
entries
    sei
    cld
    ; Grab everything on the stack
    ldx #$ff
    txs
    ; Init the processor port
    ldx #ProcessorPortDDRDefault
    stx ZPProcessorPortDDR
    ; Set the user requested ROM state
    ldy #ProcessorPortAllRAMWithIO
    sty ZPProcessorPort
    ; Clear all CIA to known state, interrupts off.
    lda #$7f
    sta CIA1InterruptControl
    sta CIA2InterruptControl
    lda #0
    sta VIC2InteruptControl
    sta CIA1TimerAControl
    sta CIA1TimerBControl
    sta CIA2TimerAControl
    sta CIA2TimerBControl
    +MACROAckAllIRQs_A

    ; Setup kernal and user mode IRQ vectors to point to a blank routine
    lda #<.initIRQ
    sta KERNALIRQServiceRoutineLo
    lda #>.initIRQ
    sta KERNALIRQServiceRoutineHi

    lda #<.initNMI
```

```
    sta KERNALNMIServiceRoutineLo
    lda #>.initNMI
    sta KERNALNMIServiceRoutineHi

    ; Turn off various bits in the VIC2 and SID chips
    ; Screen, sprites and volume are disabled.
    lda #0
    sta VIC2ScreenColour
    sta VIC2BorderColour
    sta VIC2ScreenControlV
    sta VIC2SpriteEnable

    ; Setup raster IRQ
    lda #<IrqTopOfScreen
    sta KERNALIRQServiceRoutineLo
    lda #>IrqTopOfScreen
    sta KERNALIRQServiceRoutineHi
    lda #1
    sta VIC2InteruptControl
    lda #.theLine
    sta VIC2Raster
    lda #$1b
    sta VIC2ScreenControlV

    lda #2
    sta VIC2BorderColour

    +MACROAckAllIRQs_A

    cli
.mainLine
    ; Wait for the top IRQ to be triggered
    lda .topIRQDone
    beq .mainLine

    ; If fire is pressed then don't update the counter
    lda #%10000
    bit CIA1KeyboardColumnJoystickA
    beq .mainLine

    lda #0
    sta .topIRQDone
    ldx .xPosOffset
    inx
    cpx #40
    bne .o1
    ldx #0
.o1
    stx .xPosOffset
    jmp .mainLine
```

```
.topIRQDone !by 0
.xPosOffset !by 0

; Remove all possibility that the timings will change due to previous code
!align 255,0
IrqTopOfScreen
    ; Line 45
    pha
    txa
    pha
    tya
    pha

    inc .topIRQDone

    lda #<.irq2
    ldx #>.irq2

    sta KERNALIRQServiceRoutineLo
    stx KERNALIRQServiceRoutineHi
    inc VIC2Raster
    +MACROAckRasterIRQ_A

    ; Begin the raster stabilisation code
    tsx
    cli
    ; These nops never really finish due to the raster IRQ triggering again
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
.irq2
    ; Line 46
    txs

    ; Delay for a while
    ldx #8
.l1
    dex
    bne .l1
```

```
    bit $ea
    nop

    ; Final cycle wobble check.
    lda #.theLine+1
    cmp VIC2Raster
    beq .start
.start

    ; The raster is now stable

    ; Line 47
    lda #$11
    sta VIC2ScreenControlV

    ; Waste some time
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    bit $ea

    ; Still in line 47
    ; Calculate a variable offset to delay by branching over nops
    lda #39
    sec
    sbc .xPosOffset
    ; divide by 2 to get the number of nops to skip
    lsr
    sta .sm1+1
    ; Force branch always
    clv

    ; Line 48

    ; Introduce a 1 cycle extra delay depending on the least significant bit
of the x offset
    bcc .sm1
.sm1
    bvc *
    ; The above branches somewhere into these nops depending on the x offset
position
    nop
    nop
```

```
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop

    ; Show the raster position is stable and varies as a function of x
offset
    ; If these border colour changes are removed then they need to be
replaced with the same
    ; cycle count of nops
    inc VIC2BorderColour
    dec VIC2BorderColour

    ; Do the HSP by tweaking the VIC2ScreenControlV register at the correct
time
    lda #$1b
    dec VIC2ScreenControlV
    inc VIC2ScreenControlV
    sta VIC2ScreenControlV


    ; Restart the IRQ chain
    lda #<IrqTopOfScreen
    ldx #>IrqTopOfScreen
    ldy #.theLine
    sta KERNALIRQServiceRoutineLo
    stx KERNALIRQServiceRoutineHi
    sty VIC2Raster
    +MACROAckRasterIRQ_A

    ; Exit the IRQ
    pla
    tay
    pla
    tax
    pla
```

```
.initIRQ
.initNMI
    rti
```