Using a Timer as an Inverted Raster X-Pos Register

Hi, I'm Hermit Soft. On CSDB I've shared my strongly optimized (shortened) CIA-timer using stable raster method. Now it's here to fill the gap in Codebase64. I hope it will be useful in “everyday's demo coding” . :)

A short pre-description: The first cmp$d012, bne*-3 waits till the raster-row given in the Accu. The occurence of this testing jitters in 1…7 cycles to the real starting (cycle 1) time of the rasterline, because these two commands take 4+3 cycles, and so delays detection. A raster row is 63 cycles long, and CPU can use all these cycles to do operations if no SPRITE or BADLINE fetches the actual raster-row. Also important to prevent memory page-boundaries because some commands take 1 more cycle when crossing them. Use program-start address and scan-part addresses of 256 multiplies to be clear.

We do a second rasterrow-test (cmp $d012) after exactly 56 cycles. This 56 cycles builds up from the carefully programmed part between the first “bne” and the second “cmp $d012”. If the raster-row changed, jitter made too much delay and new row started. In this case we jump back with second “bne” to the syncronization (first cmp$d012) and repeat this until we catch a case when the jitter doesn't make delay, and we can set our timer to it.

The timer (re)starts with sta$dc0e,y which is the most strictly timed, to show 7…1 on $dc04 when using the d012 scan (B-part). It counts 9 steps like 8,8,7,6,5,4,3,2,1….and again and again. It runs in cycle-syncron with the raster-beam.

Setting the CIA1-timerA to beam in the program beginning

Accumulator can be any value, no need to set.

```
         sei           ;we don't want lost cycles by IRQ calls :)
 sync:
         cmp $d012     ;scan for begin rasterline (A=$11 after first return)
         bne *-3       ;wait if not reached rasterline #$11 yet
         ldy #8        ;the walue for cia timer fetch & for y-delay loop
2 cycles
         sty $dc04     ;CIA Timer will count from 8,8 down to 7,6,5,4,3,2,1
4 cycles
         dey           ;Y=Y-1 (8 iterations: 7,6,5,4,3,2,1,0)
2 cycles*8
         bne *-1       ;loop needed to complete the poll-delay with 39
cycles    3 cycles*7+2 cycles*1
         sty $dc05     ;no need Hi-byte for timer at all (or it will mess
up)    4 cycles
         sta $dc0e,y   ;forced restart of the timer to value 8 (set in dc04)
5 cycles
         lda #$11      ;value for d012 scan and for timerstart in dc0e
2 cycles
```

```
         cmp $d012     ;check if line ended (new line) or not (same line)
         sty $d015     ;switch off sprites, they eat cycles when fetched
         bne sync      ;if line changed after 63 cycles, resyncronize it!
             ....      ;the rest (this is also a stable-timed point, can be
used for sg.)
```

Example 1: Stabilizing 7 cycle jitter when using cmp $d012

```
scan:
         ldx #$31    ;a good value that's not badline, in border and 1=white
         cpx $d012   ;scan rasterline
         bne *-3     ;wait until rasterline will be $31
         lda $dc04   ;check timer A, here it jitters between 7...1
         eor #7      ;A=7-A so jitter will be 0...6 in A
         sta *+4     ;self-writing code, the bpl jump-address = A
         bpl *+2     ;the jump to timer (A) dependent byte
         cmp #$c9    ;if A=0, cmp#$c9; if A=1, cmp #$c9 again 2 cycles later
         cmp #$c9    ;if A=2, cmp#$c9, if A=3, CMP #$EA 2 cycles later
         bit $ea24   ;if A=4,bit$ea24; if A=5, bit $ea, if A=6, only NOP
          ...........  anything you like, let's see a short example

         stx $d020   ;x was 1 so border is white at the stable cycle
         sty $d020   ;y ended in 0 in sync routine, so border black after 4
cycles
          ...........     ;pay attention on delay before rescan rasterline,
to avoid cmp$D012 again on this same rasterline in the middle.
                           If the scan happens again in the same line, that
may crash the program, because DC04 won't be usable for jumptable.
                           (For this stx $d020, sty$d020 this is specially
not problem because of the lucky timing. They're for try the routine.)
         jmp scan    ;go to the raster again (or can go new raster)
```

Example 2: stabilizing a raster IRQ

The first part (CIA setter) is also good for raster IRQ. The important thing is to do the “pha” (3 cycles) before “lda $dc04” in the called irq.

```
irq:
         pha
         lda $dc04
         eor #7
         sta *+4
         bpl *+2
         lda #$a9
         lda #$a9
         lda $eaa5
```

```
        txa
        pha
        tya
        pha

        ; the rest processes in the IRQ routine

        asl $d019
        pla
        tay
        pla
        tax
        pla
        rti
```

That's for now, test it, use it, or whatever you want... Or you can ask questions on csdb forum for this topic:

http://noname.c64.org/csdb/forums/?roomid=11&topicid=65658