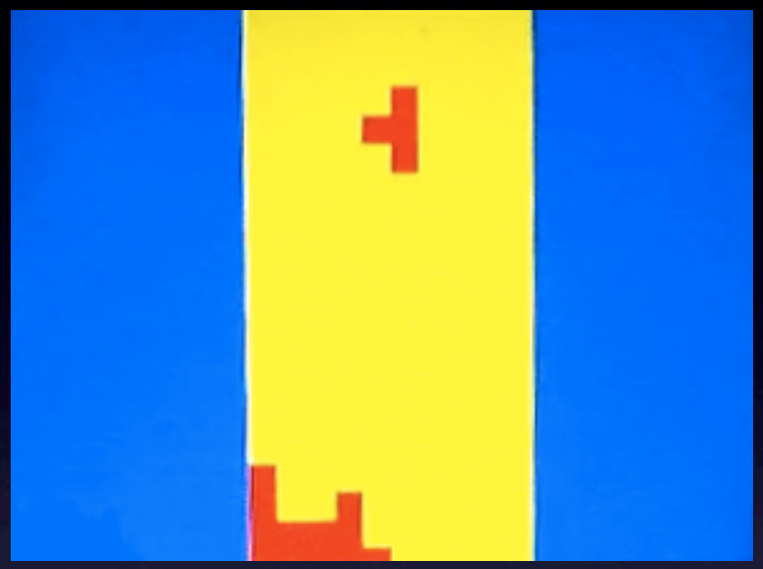
Software-only VGA generator

Mark Bowers - Fall 2011

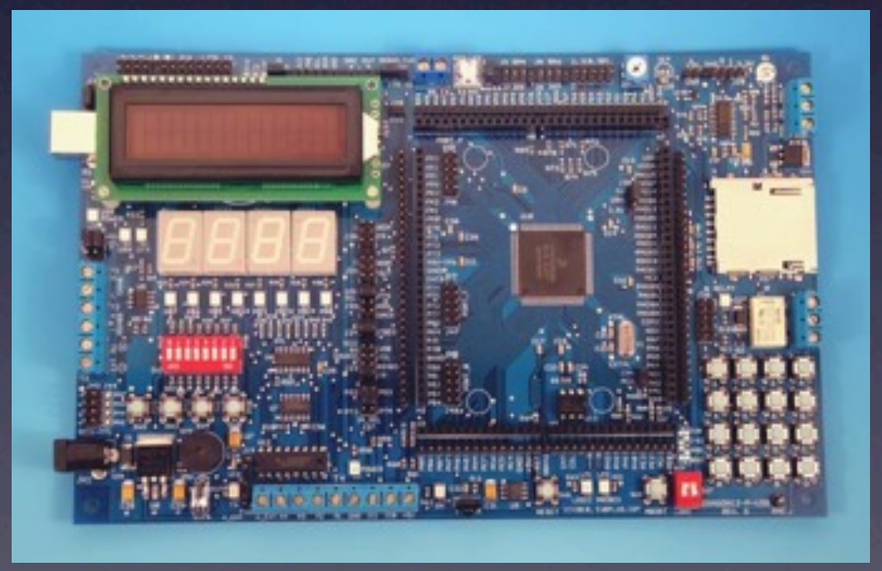
EGR280 Tetris

Winter 2009



3-axis accelerometer (analog)

button (GPIO)



Dragon12 microcontroller

(game logic)

pixel data over RS-232

VGA

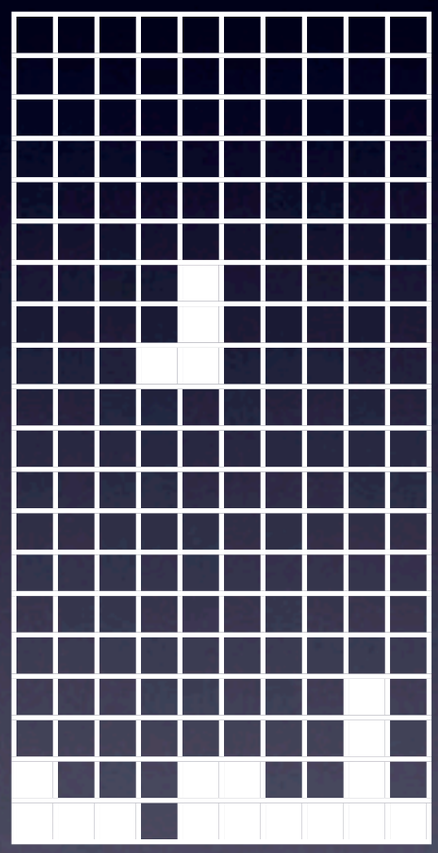
Nexys-2 FPGA board

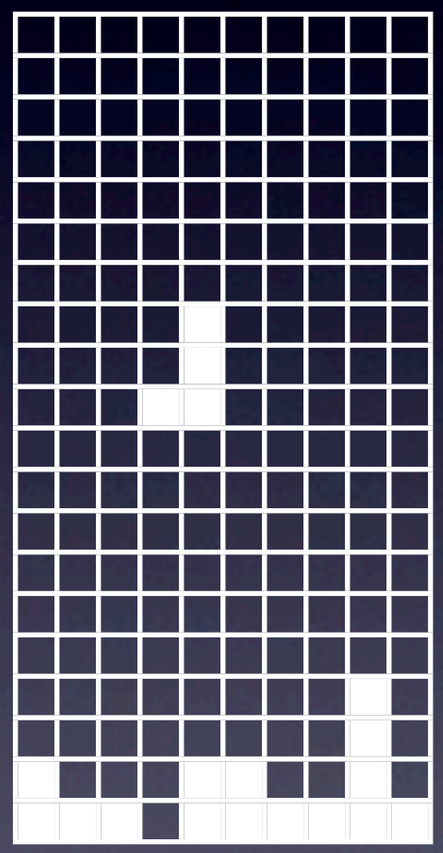
(VGA hardware)

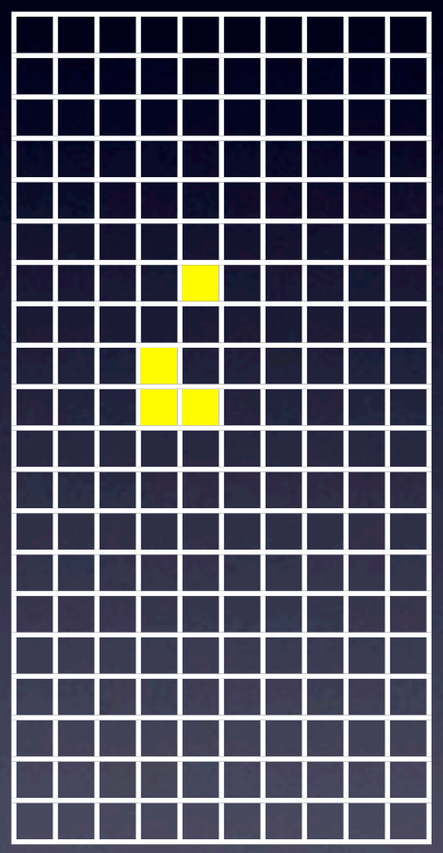
Video over RS-232

HCS12

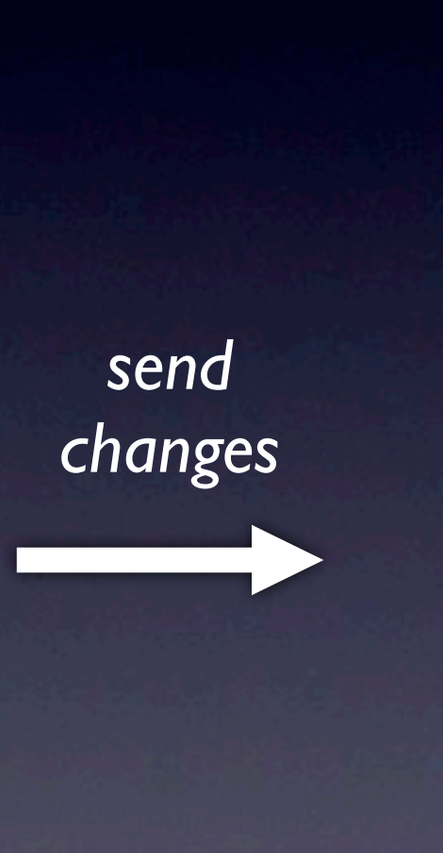
FPGA







send changes

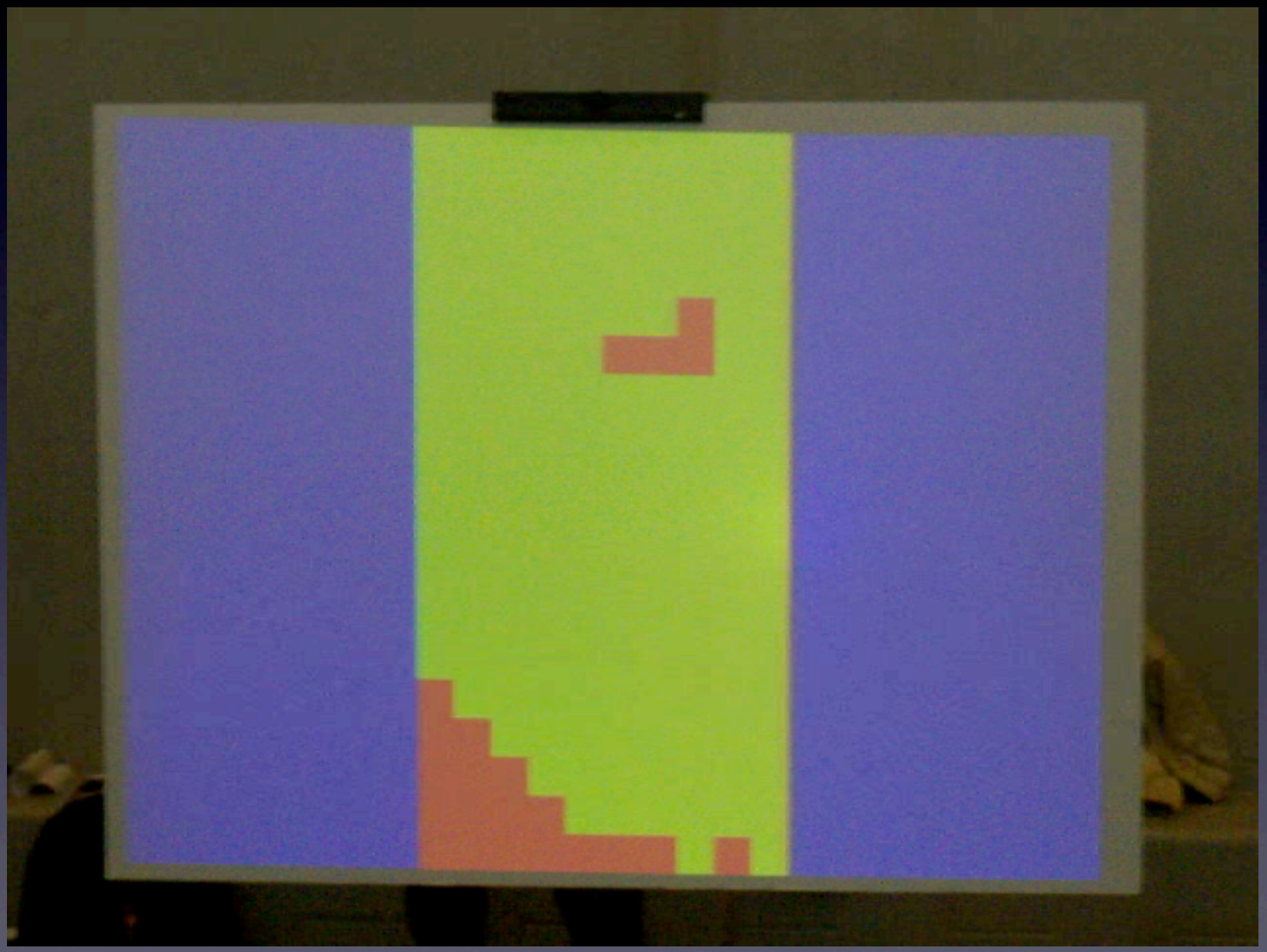


previous

next

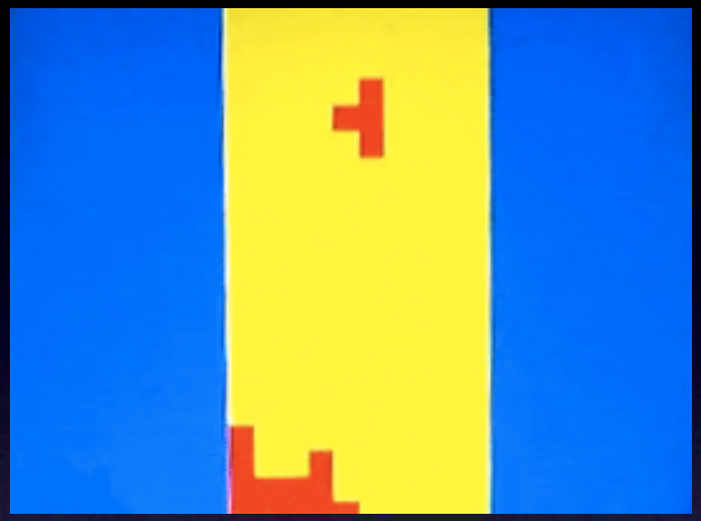
delta

EGR280 Tetris



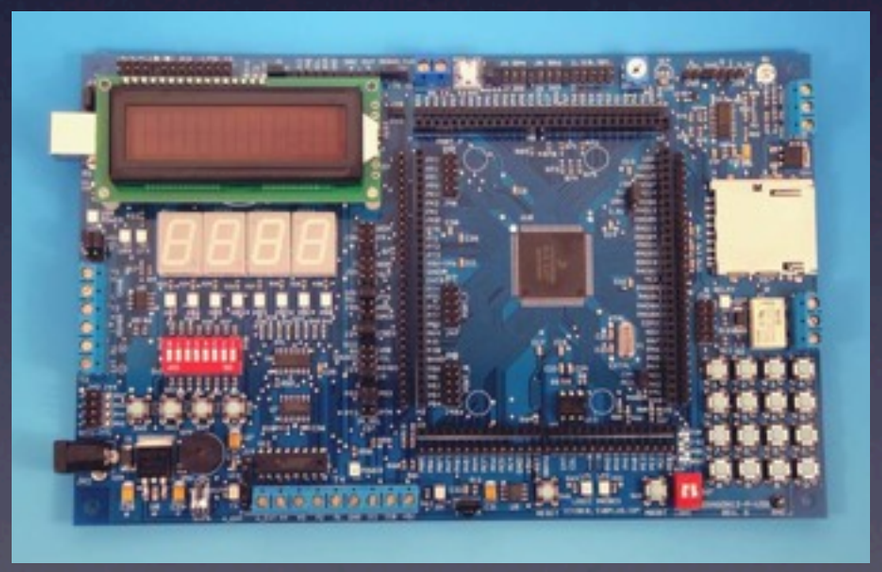
EGR280 Tetris

Winter 2009



3-axis accelerometer (analog)

button (GPIO)



Dragon12 microcontroller

(game logic)

pixel data over RS-232

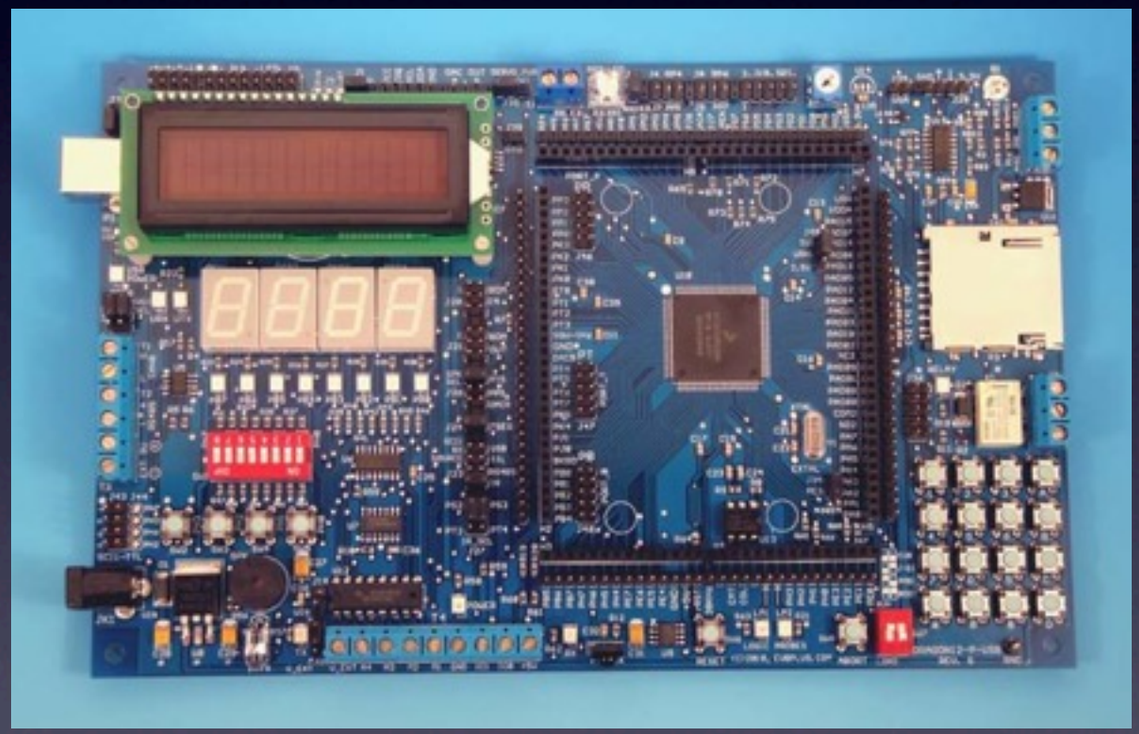
VGA

Nexys-2 FPGA board

(video controller)

Can we do everything on the microcontroller?

user input



display

<u>NO</u> external hardware!

VGA signals

sync pulses

0 - 5 volts

vertical sync

one frame per 16.7ms (60Hz)
(480 lines per frame)

horizontal sync

one line per 32μs (31.25kHz)
(640 pixels per line)

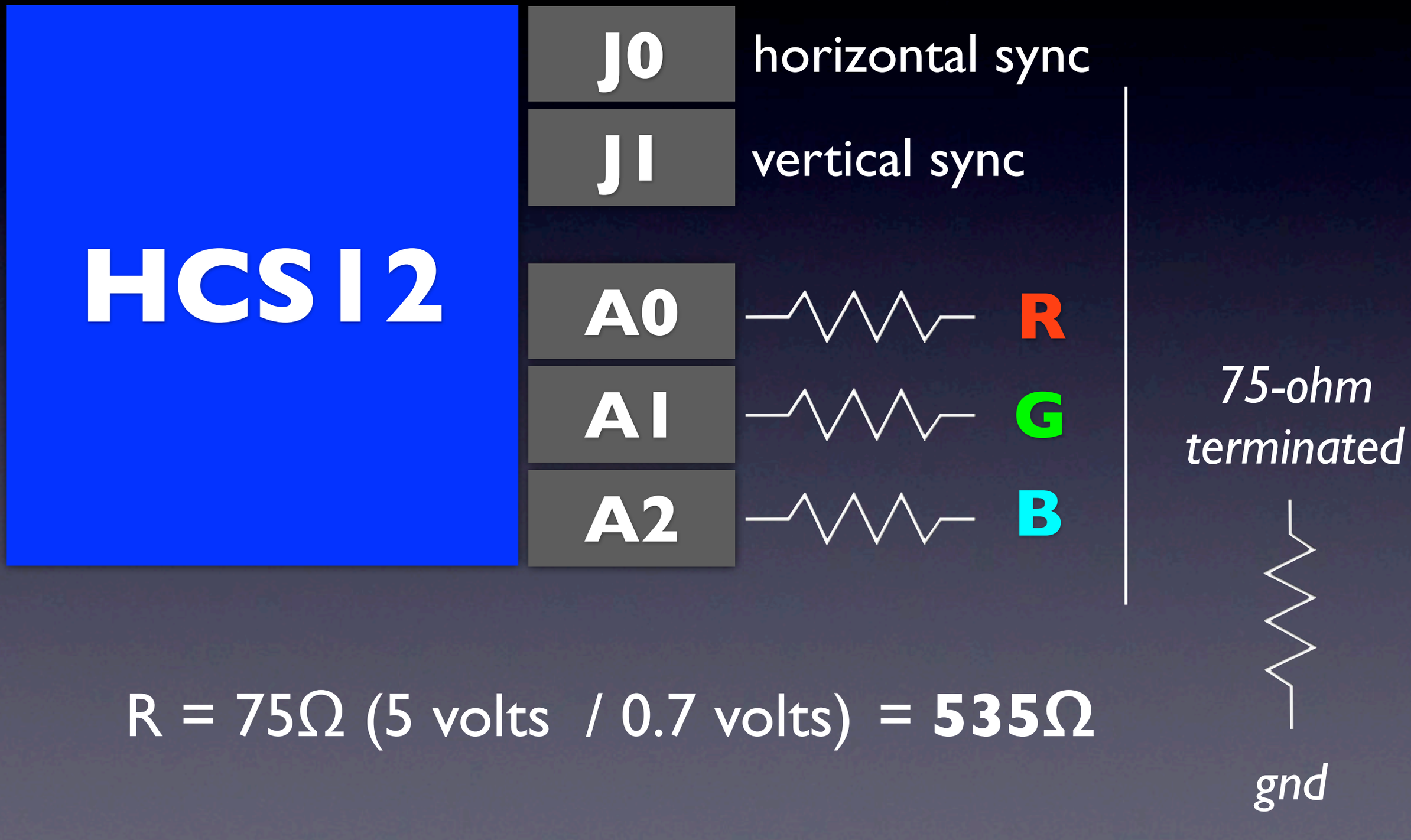
color

red
green
blue

} analog
0 - 0.7 volts

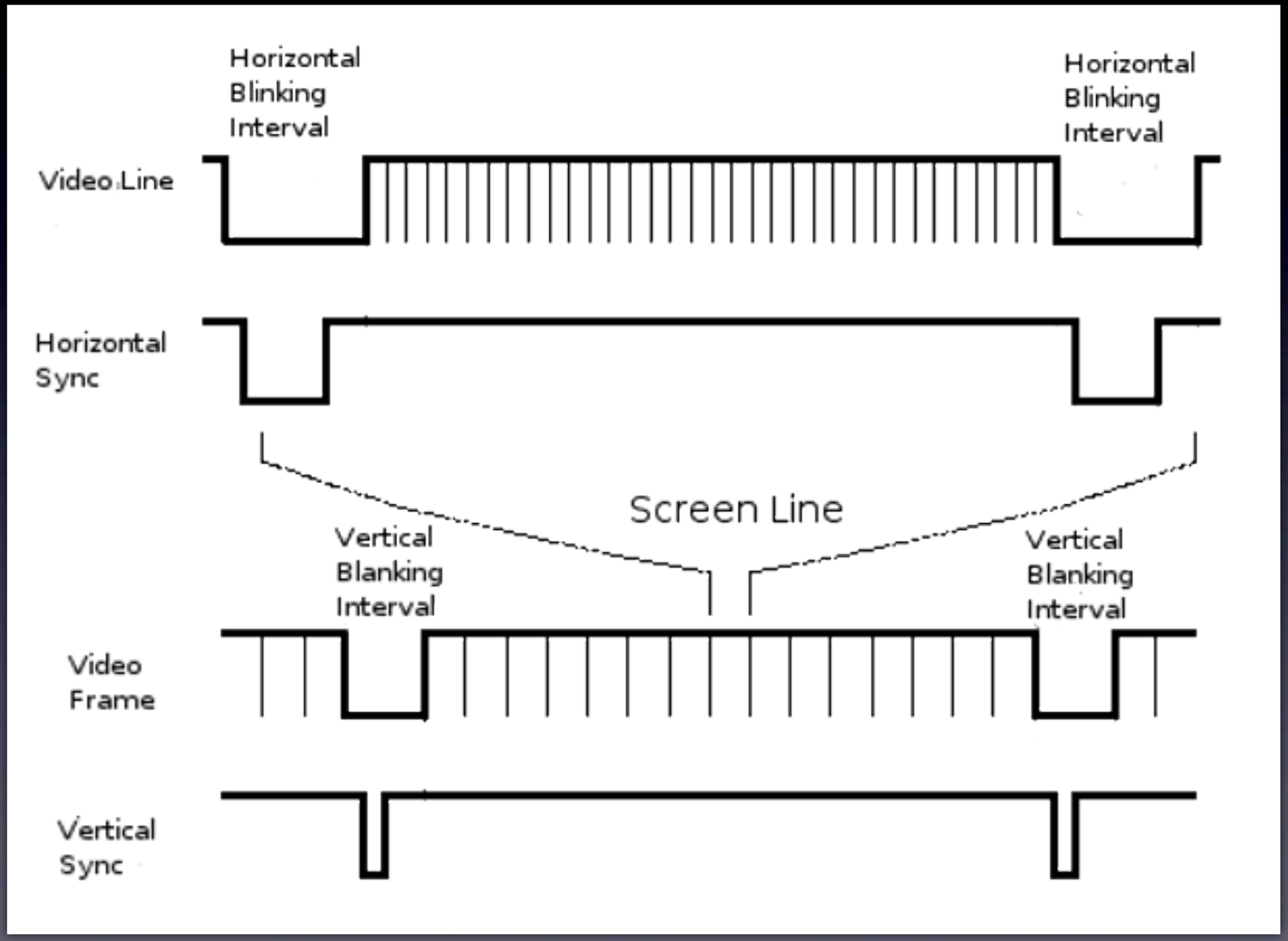
one pixel per 40ns
(25MHz)

Schematic



R = 75Ω (5 volts / 0.7 volts) = **535Ω**

VGA timing


Horizontal
Blinking
Interval
Horizontal
Blinking
Interval
Video Line
Horizontal
Sync
Screen Line
Vertical
Blanking
Interval
Vertical
Blanking
Interval
Video
Frame
Vertical
Sync

HCS12 PLL

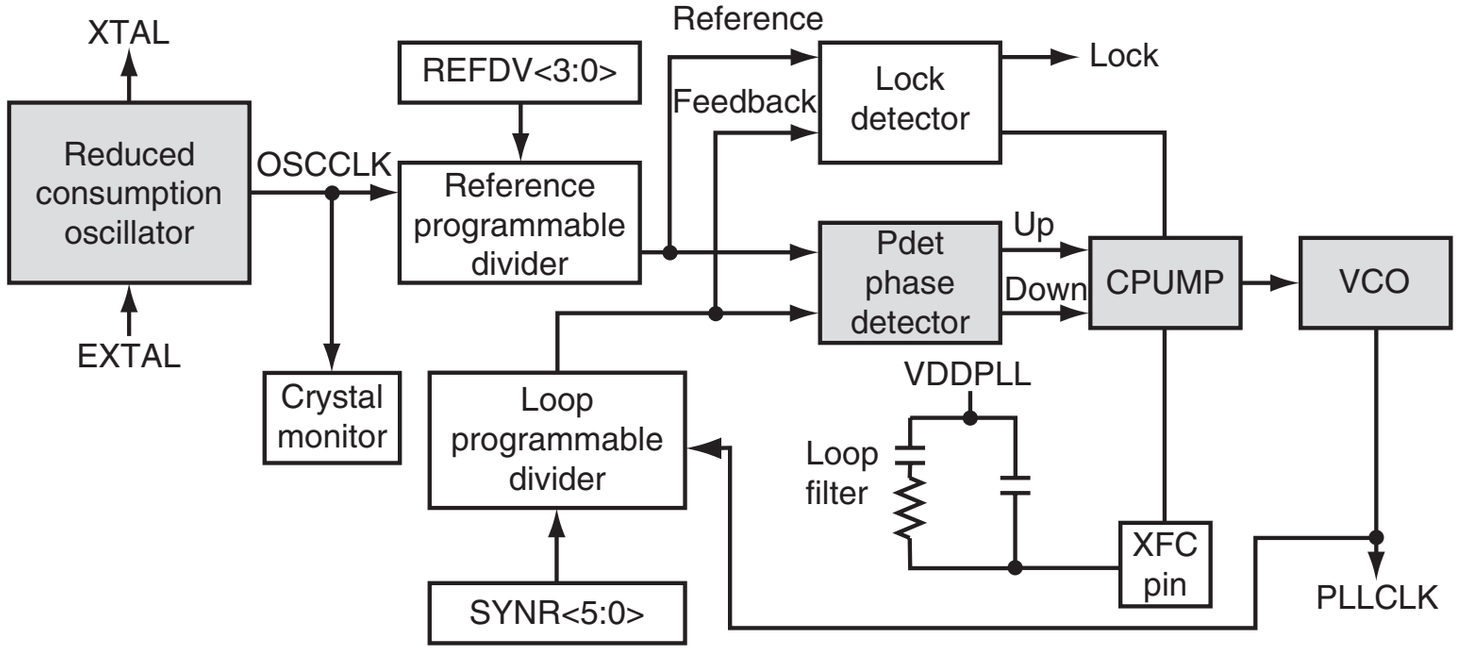


Figure 6.10 ■ PLL functional diagram

PLLCLK = OSCCLK * 2 * (SYNR+1) / (REFDV+1)

50MHz = 16MHz * 2 * (24+1) / (15+1)

HCS12 clock

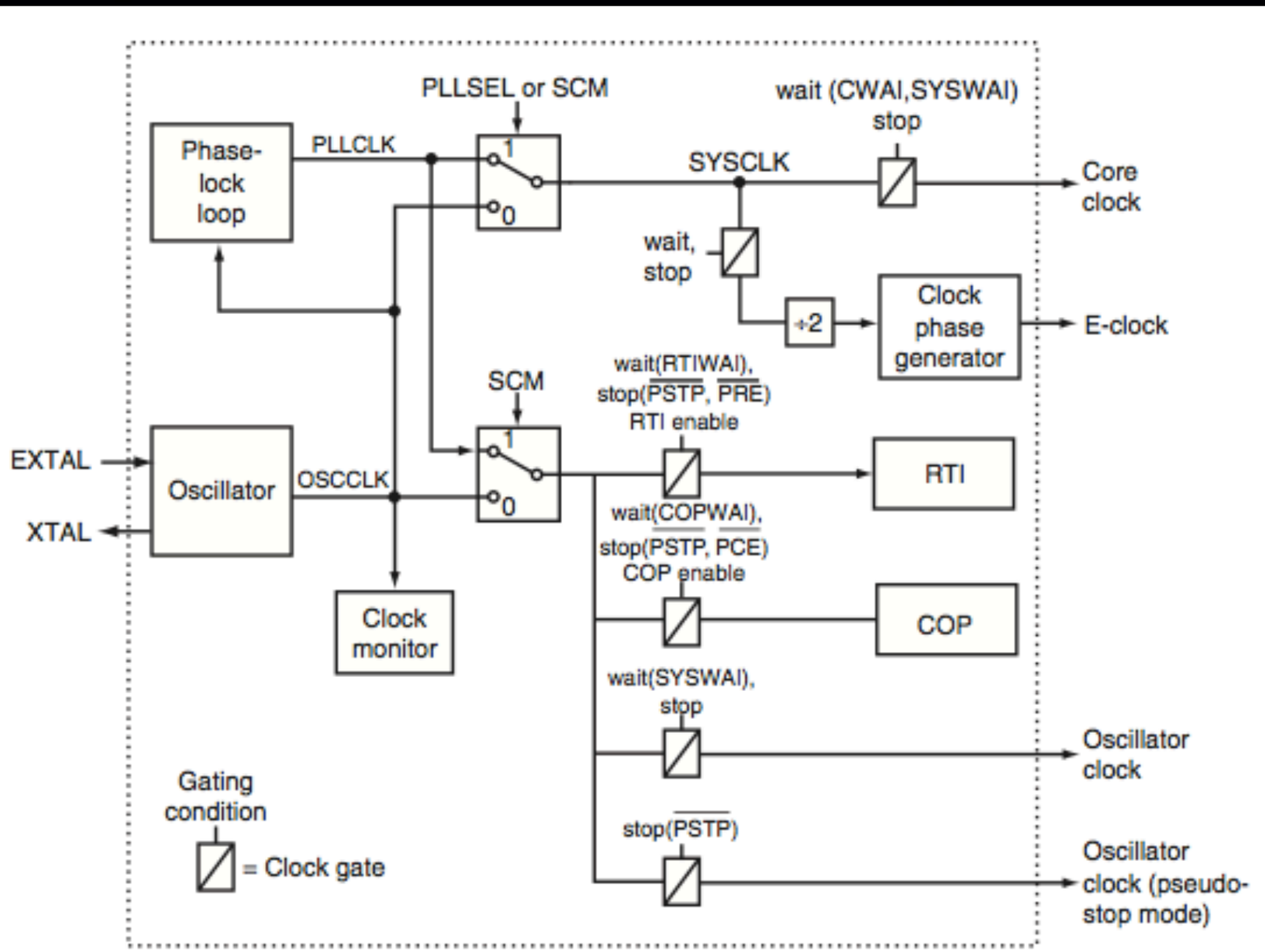


Figure 6.15 ■ HCS12 clock generation circuit

Clock setup routine

```
PLL_setup:
            bclr CLKSEL,%10000000       ; switch clock to OSCCLK
            bset PLLCTL,%01000000       ; turn on PLL
            movb #$18, SYNR             ; set clock multiplier
            movb #$0F, REFDV            ; set clock divider
PLL_lock:   brclr CRGFLG,$08, PLL_lock  ; wait until locked
            bset CLKSEL,%10000000       ; switch clock to PLLCLK
            bclr PEAR,$10               ; put ECLK on PE4 (pin 39)
            rts
```

GPIO speed

ECLK

MOVB $M_1 \Rightarrow M_2$

Source Form[(1)]	Address Mode	Object Code	Access Detail HCS12	Access Detail M68HC12
MOVB #*opr8*, *opr16a*	IMM–EXT	18 0B ii hh ll	OPwP	OPwP
MOVB #*opr8i*, *oprx0_xysp*	IMM–IDX	18 08 xb ii	OPwO	OPwO
MOVB *opr16a*, *opr16a*	EXT–EXT	18 0C hh ll hh ll	OrPwPO	OrPwPO
MOVB *opr16a*, *oprx0_xysp*	EXT–IDX	18 09 xb hh ll	OPrPw	OPrPw
MOVB *oprx0_xysp*, *opr16a*	IDX–EXT	18 0D xb hh ll	OrPwP	OrPwP
MOVB *oprx0_xysp*, *oprx0_xysp*	IDX–IDX	18 0A xb xb	OrPwO	OrPwO

Minimum five ECLK cycles to write a byte

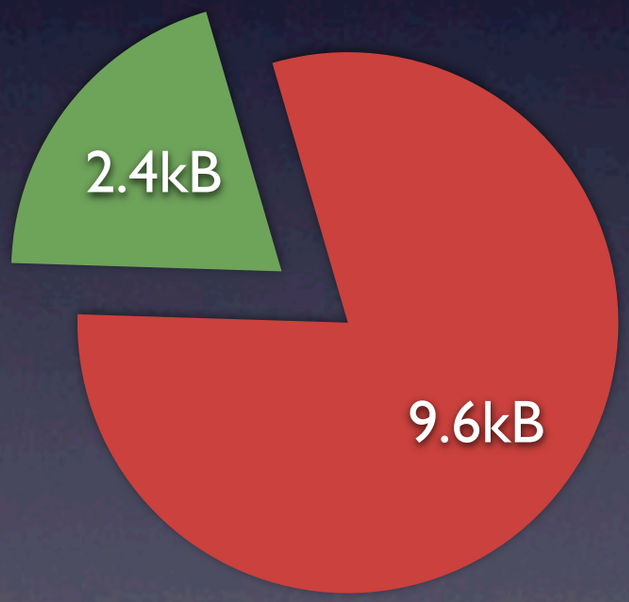
Memory constraints

12kB internal RAM
on MC9S12DG256

640 x 480 = 307 kB

160 x 120 = 19.2 kB

80 x 60 = 4.8 kB



double buffered
80x60 array

modify ⇒ buffer 1

buffer 2 ⇒ display

Timer module

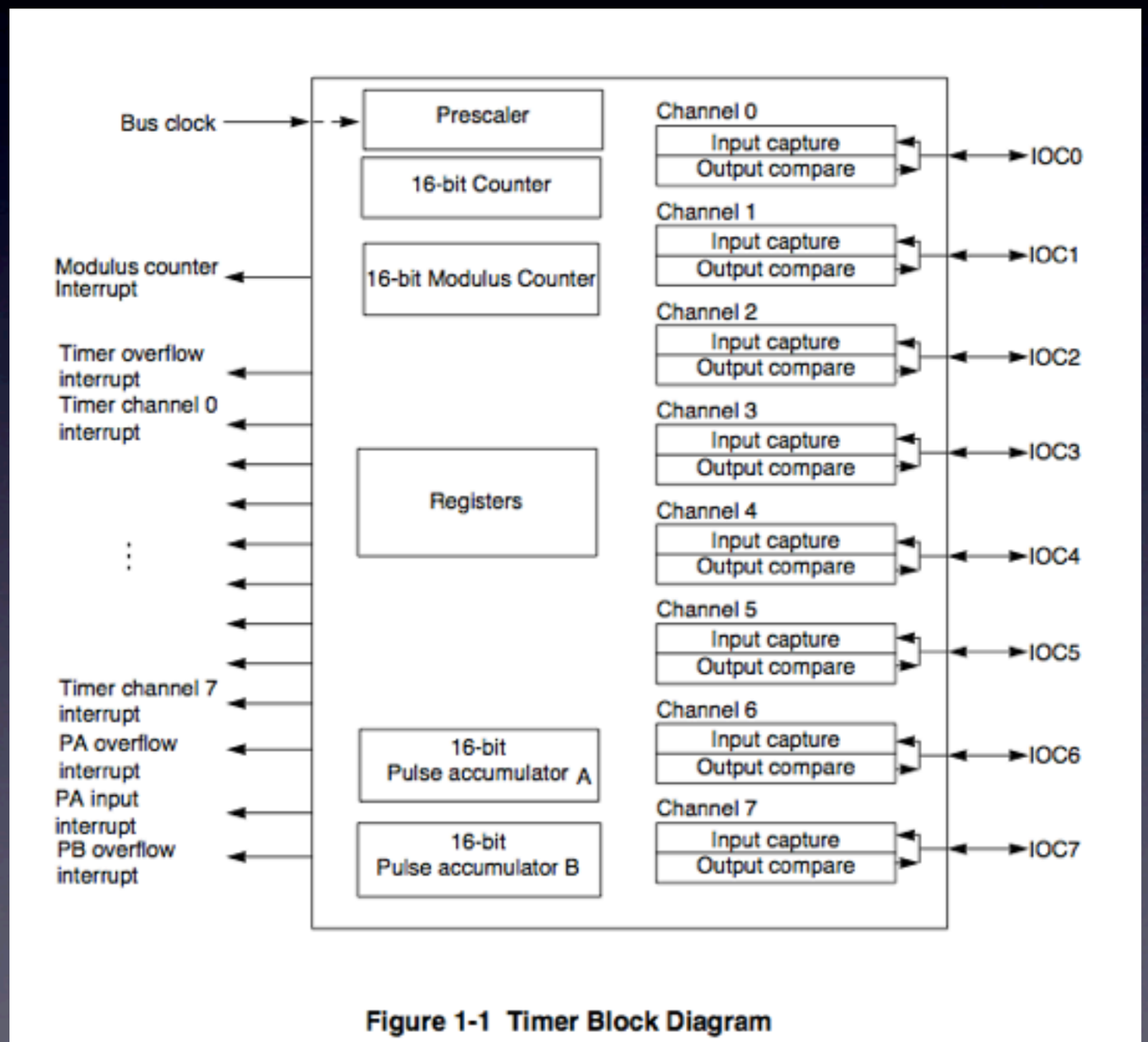


Figure 1-1 Timer Block Diagram

Modulus counter

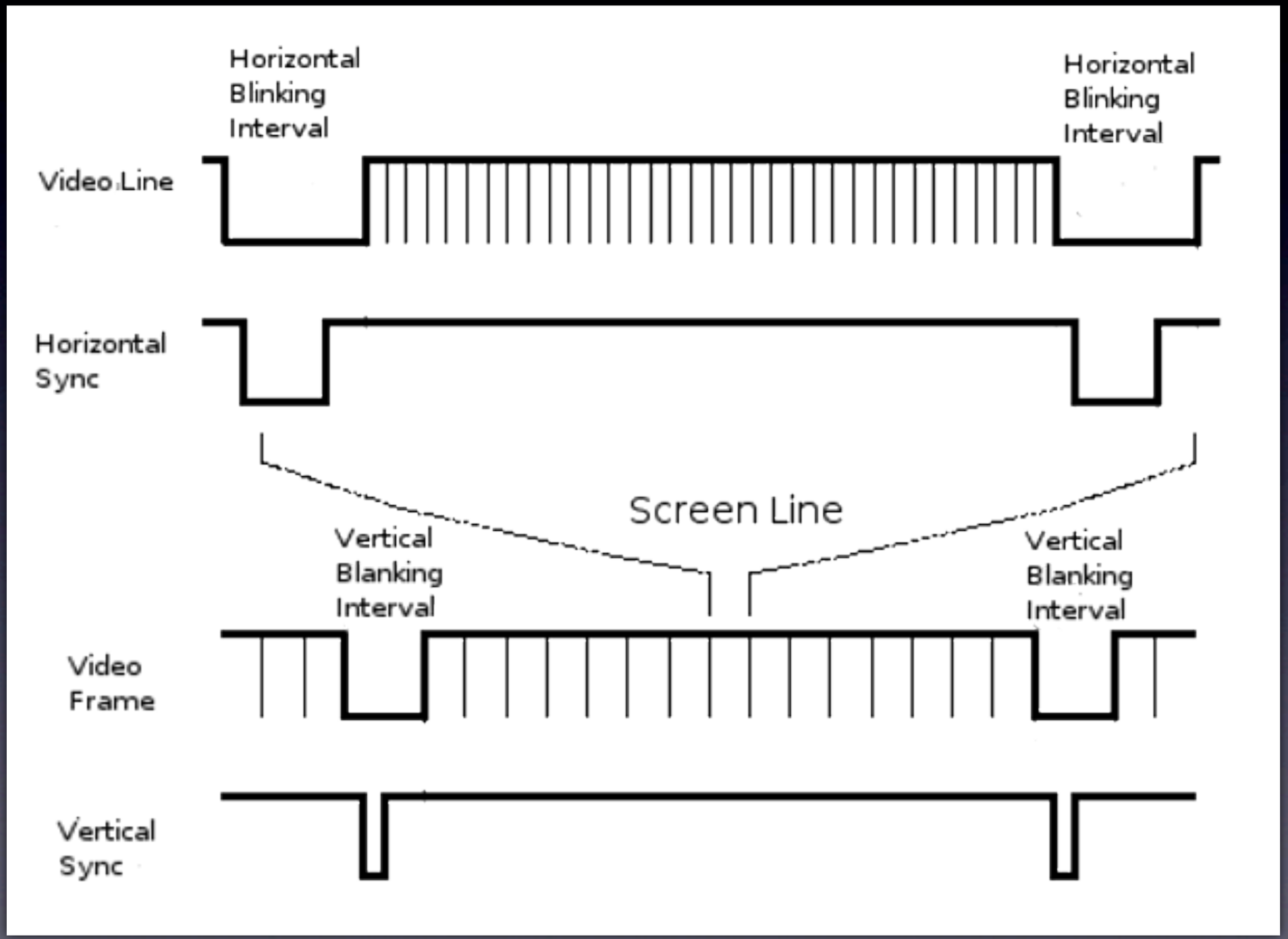
- 16-bit down-counter
- After reaching zero:
  1. Fire interrupt
  2. Set counter again
- Good for generating a very accurate periodic interrupt

Video setup routine

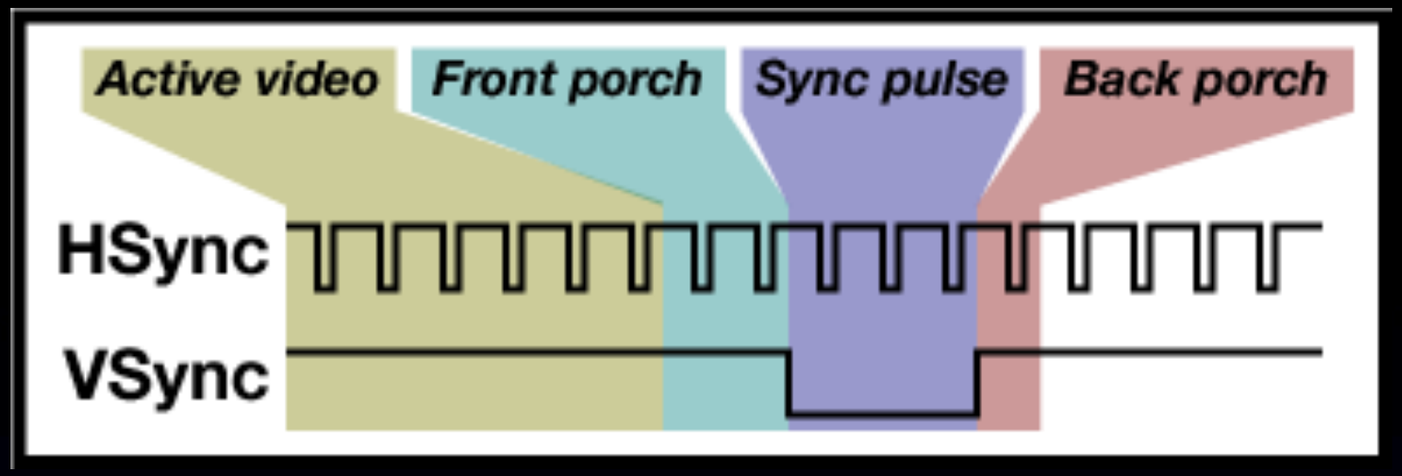
```
; setup modulus down-counter interrupt
; to fire at 31.25kHz (horizontal line rate)
; MCCTL = 11000100 (no prescale)
; MCCNT = 25MHz/31.25kHz = 800

video_enable:
        movb #$CA,HPRIO         ; elevate MDC to highest priority
        movb #%11000100,MCCTL   ; enable MDC, no prescaler
        movw #800,MCCNT         ; set count value
        bset PTJ,%00000011      ; set PJ0 and PJ1 high
        clr  PORTA              ; clear PORTA
        bset DDRJ,%00000011     ; set PJ0 and PJ1 output
        movb #$FF,DDRA          ; set PORTA output
        cli                     ; enable interrupts
        rts
```

VGA timing


Horizontal
Blinking
Interval
Horizontal
Blinking
Interval
Video Line
Horizontal
Sync
Screen Line
Vertical
Blanking
Interval
Vertical
Blanking
Interval
Video
Frame
Vertical
Sync

Video ISR



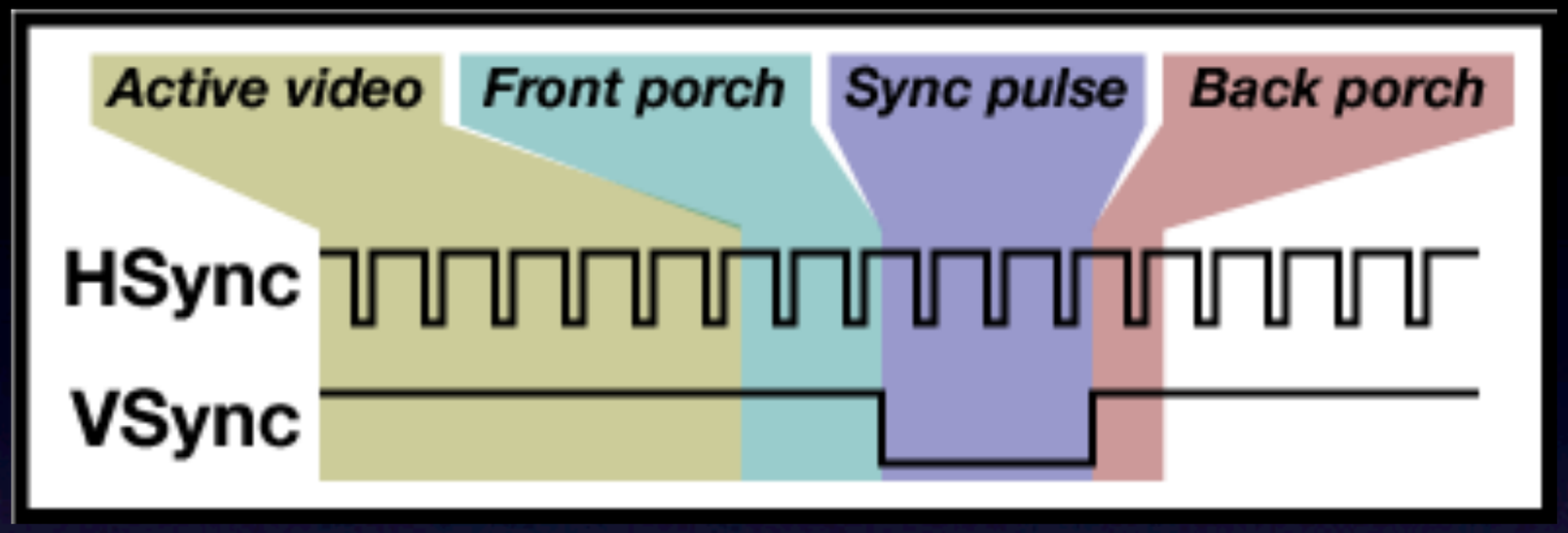
enter
region?
active
inactive
VSYNC start?
no
VSYNC end?
no
yes
yes
VSYNC set 0
VSYNC set 1
pulse HSYNC
exit interrupt
enable interrupt
done
next line
not done
next
display line
repeat?
disable interrupt
pulse HSYNC

Pixel routine

```
; writes a pixel color to PORTA
; (8 cycles)

video_1:   macro
           inx              ; increment X    (1 cycle)
           nop              ; waste time     (1 cycle)
           nop              ; waste time     (1 cycle)
           movb X, PORTA  ; write to pins  (5 cycles)
           endm
```

Usable time



Total frame time = **16.67ms**

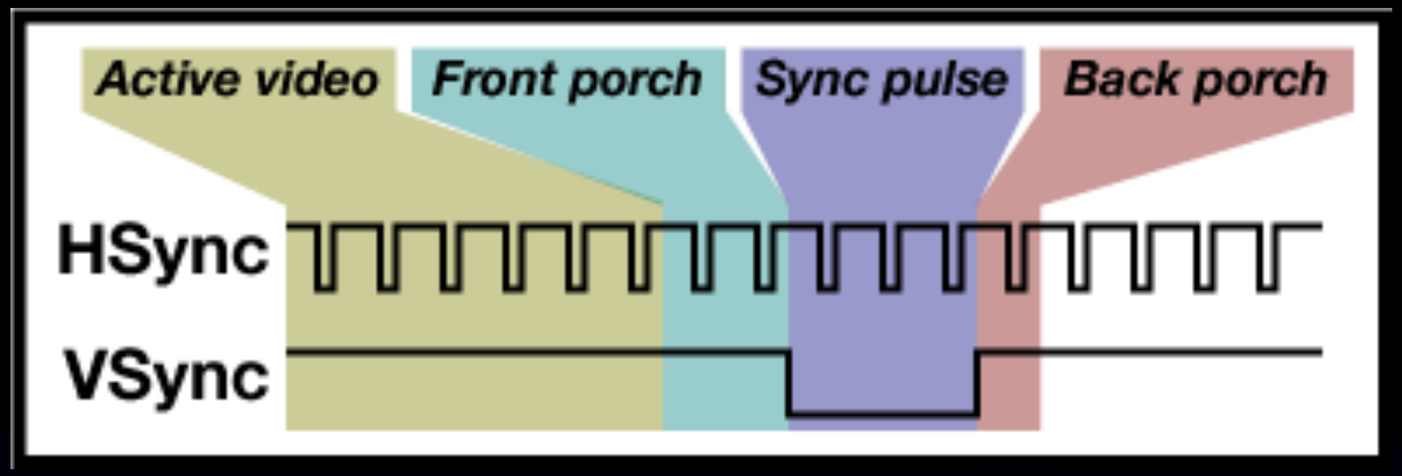
Inactive time = 0.928ms + 0.064ms + 0.320ms = **1.312ms**

Inactive HS pulses = (29 + 2 + 10) * 32µs = **0.209ms**

Usable time = **1.103ms**

1.103 / 16.67 = **6.6%**

Video ISR



enter
region?
active
inactive
VSYNC start?
no
VSYNC end?
no
yes
yes
VSYNC set 0
VSYNC set 1
pulse HSYNC
exit interrupt
enable interrupt
done
next line
not done
next
display line
repeat?
disable interrupt
pulse HSYNC

Branch balancing

```
tst vid_blanks          ; check if blank count is zero   (3 cycles)
beq start_active        ; go begin active video section

 ; common                                  (1 cycle) <- NOT BRANCHED
ldaa vid_blanks         ; load A          (3 cycles)
cmpa #31                ; VSYNC start?    (1 cycle)
beq vs_low

; (A/=31)               ;                 (1 cycle) <- NOT BRANCHED
cmpa #29                ; VSYNC end?      (1 cycle)
beq vs_high

; else                                    (1 cycle) <- NOT BRANCHED
        bclr PTJ,%00000001      ; HS=0            (4 cycles)
        nop                     ; burn time       (1 cycle)
        bra hs_pulse
        ; TOTAL FROM START=(3)+(1+3+1)+(1+1)+(1+4+1)=17

         ; (A=31)
 vs_low: ; bring VSYNC low                        (3 cycles) <- BRANCHED
        movb #%00000000,PTJ     ; VS=0, HS=0      (4 cycles)
        nop                     ; burn time       (1 cycle)
        nop                     ; burn time       (1 cycle)
        bra hs_pulse
        ; TOTAL FROM START=(3)+(1+3+1)+(3+4+1+1)=17

         ; (A=29)
vs_high: ; bring VSYNC high                       (3 cycles) <- BRANCHED
        movb #%00000010,PTJ      ; VS=1, HS=0     (4 cycles)
        bra hs_pulse
        ; TOTAL FROM START=(3)+(1+3+1)+(1+1)+(3+4)=17
```

Results

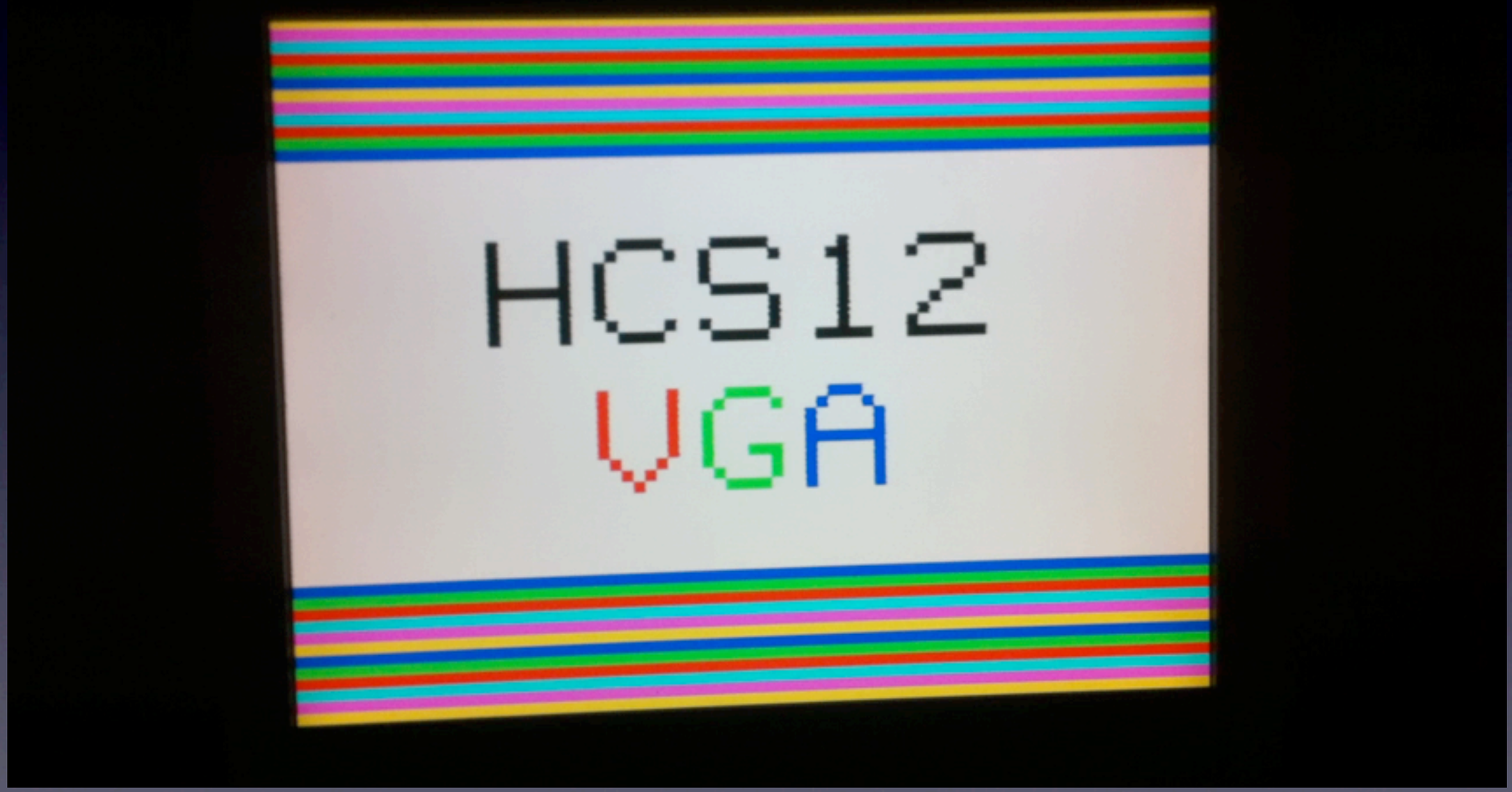

Envision

HCS12

VGA

HCS12

VGA


HCS12
VGA


HCS12