```
; Keyboard scan code interface
;
; Assembles serial data bits from keyboard into a byte, and signals an interrupt
; when all the bits have been read.
;
; A 74LS161 counts the bits as they arrive. The count increments from 0 to 11 after all
; bits have been read. The GAL reads the count and asserts IRQ as long as count=11. The
; '161 is cleared asynchronously when the data byte is read (OE is low) or during reset,
; effectively clearing the IRQ and preparing the counter for the next byte.
;
; IRQ is also wired to the 74LS161's ENP input, inhibiting further counting once the count
; reaches 11. This ensures that if a byte isn't read promptly before the next byte arrives,
; the next byte will be ignored, rather than having the count increment further from 11 and
; get all out of sync.

CHIP keyboard G22V10

kbclk=1 kbdata=2 cnt0=3 cnt1=4 cnt2=5 cnt3=6 _reset=7 _hwsel_usbkybd=8 a1=9 a0=10
    _ld_mem=11 gnd=12
clk=13 _aclrcnt=14 _irq=15 q7=16 q6=17 q5=18 q4=19 q3=20 q2=21 q1=22 q0=23 vcc=24

EQUATIONS

; the keyboard transmits 11 bits:
; 0 start
; 1-8 data LSB first
; 9 odd parity
; 10 stop
q0.oe = /_hwsel_usbkybd*a1*/a0*_ld_mem
q0 := /cnt3*/cnt2*/cnt1*cnt0*kbdata + cnt3*q0 + cnt2*q0 + cnt1*q0 + /cnt0*q0
q1.oe = /_hwsel_usbkybd*a1*/a0*_ld_mem
q1 := /cnt3*/cnt2*cnt1*/cnt0*kbdata + cnt3*q1 + cnt2*q1 + /cnt1*q1 + cnt0*q1
q2.oe = /_hwsel_usbkybd*a1*/a0*_ld_mem
q2 := /cnt3*/cnt2*cnt1*cnt0*kbdata + cnt3*q2 + cnt2*q2 + /cnt1*q2 + /cnt0*q2
q3.oe = /_hwsel_usbkybd*a1*/a0*_ld_mem
q3 := /cnt3*cnt2*/cnt1*/cnt0*kbdata + cnt3*q3 + /cnt2*q3 + cnt1*q3 + cnt0*q3
q4.oe = /_hwsel_usbkybd*a1*/a0*_ld_mem
q4 := /cnt3*cnt2*/cnt1*cnt0*kbdata + cnt3*q4 + /cnt2*q4 + cnt1*q4 + /cnt0*q4
q5.oe = /_hwsel_usbkybd*a1*/a0*_ld_mem
q5 := /cnt3*cnt2*cnt1*/cnt0*kbdata + cnt3*q5 + /cnt2*q5 + /cnt1*q5 + cnt0*q5
q6.oe = /_hwsel_usbkybd*a1*/a0*_ld_mem
q6 := /cnt3*cnt2*cnt1*cnt0*kbdata + cnt3*q6 + /cnt2*q6 + /cnt1*q6 + /cnt0*q6
q7.oe = /_hwsel_usbkybd*a1*/a0*_ld_mem
q7 := cnt3*/cnt2*/cnt1*/cnt0*kbdata + /cnt3*q7 + cnt2*q7 + cnt1*q7 + cnt0*q7

; irq is asserted when count is 11 (all the bits have been read)
/_irq = cnt3*/cnt2*cnt1*cnt0

; Count is cleared asynchronously when data is read, or during reset.
; Clear will only be asserted during the 2nd half of the system clock, in an
; imperfect attempt to avoid glitches.
; Furthermore, clear will only be asserted if IRQ is asserted.
/_aclrcnt = /_irq*/clk*/_hwsel_usbkybd*a1*/a0*_ld_mem + /_reset
```

```
; Horizontal low pixel counter
;
; Counts the number of pixels in a scan line

CHIP horizlo G22V10

; dotclk is 25.175 MHz = 39.72 ns period
; 800 count per scanline = 31.78us
; 512 pixels in visible video region

; vsize  lines  mode
;  00     240   per line
;  01     240   per frame
;  10     480   per line
;  11     480   per frame

dotclk=1 _vramcs=2 _newline=3 text=4 vsize1=5 r0=6 vram7=7 _oe_mem=8 _we_mem=9 _hwsel_pal
    =10 bpp0=11 gnd=12
bpp1=13 _we_pal=14 c3m=15 _oe_vram=16 invert=17 _ldbyte=18 c3=19 c2=20 c1=21 c0=22 v0=23
    vcc=24

EQUATIONS

; Load a new data byte every 4 pixels in 2, 4, or 8 bpp modes.
; Load a new data byte every 8 pixels in 1 bpp mode.
; Note this is registered, in order to reduce the delay from clock to valid value at
; the shifter input. As a result, the equations check for the column before the
; one where _ldbyte should be asserted.
/_ldbyte := bpp0*c1*/c0 +
            bpp1*c1*/c0 +
            /bpp1*/bpp0*c2*c1*/c0

invert = text*vram7

/_oe_vram = _vramcs + /_oe_mem

/_we_pal = /_hwsel_pal*/_we_mem

/c0 := /_newline +
       c0

/c1 := /_newline +
       c1*c0 +
       /c1*/c0

/c2 := /_newline +
       c2*c1*c0 +
       /c2*/c1 +
       /c2*/c0

/c3 := /_newline +
       c3*c2*c1*c0 +
       /c3*/c2 +
       /c3*/c1 +
       /c3*/c0

c3m.oe = _vramcs
/c3m := /_newline +
       c3*c2*c1*c0 +
       /c3*/c2 +
       /c3*/c1 +
       /c3*/c0

v0.oe  = _vramcs
/v0 = text +
      /text*/vsize1*/c2 +
```

```
      /text*vsize1*/r0
```

```
; Horizontal high pixel counter
;
; Counts the number of pixels in a scan line, and generates the hsync, hblank, and data
; load enable signals.

CHIP horizhi G22V10

; dotclk is 25.175 MHz = 39.72 ns period
; 800 count per scanline = 31.78us
; 512 pixels in visible video region

; vsize  lines  mode
;  00     240   per line
;  01     240   per frame
;  10     480   per line
;  11     480   per frame

; v0 (LSB of VRAM address) behavior:
; always 0 for text mode
; c2 for 240 line modes (address changes every 4 columns, every 2 rows)
; r0 for 480 line modes (address changes every 8 columns, every row)

dotclk=1 _vramcs=2 c0=3 c1=4 c2=5 c3=6 vsize0=7 vsize1=8 newframe=9 _vsync=10 r9=11 gnd=
    12
nc=13 _ldmode=14 _hsync=15 _hblank=16 /c9=17 c8=18 c7=19 c6=20 c5=21 c4=22 _newline=23
    vcc=24

EQUATIONS

; could try making this unregistered
/_newline := c9*c8*/c7*/c6*/c5*c4*c3*c2*c1*/c0

; If vsize0 = 1, then _ldmode is asserted only once per frame, instead of every line.
; That allows the two mode bytes per line to be used as pixel data instead.
; For vsize = X1, _ldmode is asserted at the beginning of lines 490-491
; The mode byte is loaded when c = 7. For 1bpp mode or 480 line modes, this is the first
; byte of the line. For 2/4/8bpp 240 line modes, this is the second byte of the line.
/_ldmode = _vramcs*/vsize0*/c9*/c8*/c7*/c6*/c5*/c4*/c3*c2*c1*c0 +
           _vramcs*vsize0*/_vsync*/c9*/c8*/c7*/c6*/c5*/c4*/c3*c2*c1*c0

; _hblank is theoretically active for columns 512-799, but it gets complicated:
; If vsize0 = 0 (mode per line), then _hblank is also active for columns 0-7, while the
    line mode
; bytes are loaded.
; Note 1: This is registered. As a result, the equations check for the column before the
; one where _hblank should be asserted.
; Note 2: Whatever the current column count, the pixel data currently being input to the
    palette
; is from 8 columns ago at 1bpp, or 4 columns ago at 2/4/8bpp.
; When loading every 8 columns:
;   shifter loads when C=0..0111
;   next cycle C=0..1000, and pixel being input is C=0..0000
;   so column count is 8 ahead of actual pixel
; When loading every 4 columns:
;   shifter loads when c=0..0011
;   next cycle C=0..0100, and pixel being input is C=0..0000
;   so column count is 4 ahead of actual pixel
; Note 3: There would ordinarily be a +/- 4 column difference in when the first visible
    pixel appears
; relative to _hsync, between 1bpp and 2/4/8 bpp at the same vertical resolution. This is
     undesirable
; because it requires bpp to be used in the equation for _hblank, and it means mixed-mode
     screens
; with some 1bpp lines and some 2/4/8bpp lines wouldn't be flush with each other. The
    solution adopted
; here is to use the 1bpp _hblank timing everywhere. For 2/4/8bpp lines, this has the
```

```
    effect of
; suppressing the first 4 columns of visible pixels, and attaching them to the end of the
    line instead.
; So the logical line begins at byte 1 of the line data for 2/4/8bpp at 240 lines with no
    mode data,
; and ends at byte 0. With mode data, it begins at byte 3 and ends at byte 0. Software
    must compensate
; to create the intended display. But for 2/4/8bpp lines with vsize1 = 1 (480 lines),
    this
; wrap-around incorrectly displays the mode data as visible pixels.
; For vsize0 = 1: _hblank should be inactive for columns 8-519
; For vsize0 = 0: _hblank should be inactive for columns 16-519
; Then subtract 1 from all these to account for _hblank being registered.
; 00 0000 0111 - 7
; 00 0000 1XXX - 8-15
;
; 00 0000 1111 - 15
; 00 0001 XXXX - 16-31
; 00 001X XXXX - 32-63
; 00 01XX XXXX - 64-127
; 00 1XXX XXXX - 128-255
; 01 XXXX XXXX - 256-511
; 10 0000 00XX - 512-515
; 10 0000 010X - 516-517
; 10 0000 0110 - 518
_hblank := vsize0*/c9*/c8*/c7*/c6*/c5*/c4*/c3*c2*c1*c0 +
           vsize0*/c9*/c8*/c7*/c6*/c5*/c4*c3 +
           /c9*/c8*/c7*/c6*/c5*/c4*c3*c2*c1*c0 +
           /c9*/c8*/c7*/c6*/c5*c4 +
           /c9*/c8*/c7*/c6*c5 +
           /c9*/c8*/c7*c6 +
           /c9*/c8*c7 +
           /c9*c8 +
           c9*/c8*/c7*/c6*/c5*/c4*/c3*/c2 +
           c9*/c8*/c7*/c6*/c5*/c4*/c3*c2*/c1 +
           c9*/c8*/c7*/c6*/c5*/c4*/c3*c2*c1*/c0

; _hsync is active for 96 pixels, 600-695, 3.81us
; this is intentionally offset in order to shift the visible 512
; pixels 64 to the right, to center the display within a 640 pixel region
; 10 0101 1XXX - 600-607
; 10 011X XXXX - 608-639
; 10 100X XXXX - 640-671
; 10 1010 XXXX - 672-687
; 10 1011 0XXX - 688-695
/_hsync := c9*/c8*/c7*c6*/c5*c4*c3 +
           c9*/c8*/c7*c6*c5 +
           c9*/c8*c7*/c6*/c5 +
           c9*/c8*c7*/c6*c5*/c4 +
           c9*/c8*c7*/c6*c5*c4*/c3


; column count wraps at 799
c4.oe = _vramcs
/c4 := c9*c8*/c7*/c6*/c5*c4*c3*c2*c1*c0 +
       c4*c3*c2*c1*c0 +
       /c4*/c3 +
       /c4*/c2 +
       /c4*/c1 +
       /c4*/c0

c5.oe = _vramcs
/c5 := c9*c8*/c7*/c6*/c5*c4*c3*c2*c1*c0 +
       c5*c4*c3*c2*c1*c0 +
       /c5*/c4 +
       /c5*/c3 +
```

```
       /c5*/c2 +
       /c5*/c1 +
       /c5*/c0

c6.oe = _vramcs
/c6 := c9*c8*/c7*/c6*/c5*c4*c3*c2*c1*c0 +
       c6*c5*c4*c3*c2*c1*c0 +
       /c6*/c5 +
       /c6*/c4 +
       /c6*/c3 +
       /c6*/c2 +
       /c6*/c1 +
       /c6*/c0

c7.oe = _vramcs
/c7 := c9*c8*/c7*/c6*/c5*c4*c3*c2*c1*c0 +
       c7*c6*c5*c4*c3*c2*c1*c0 +
       /c7*/c6 +
       /c7*/c5 +
       /c7*/c4 +
       /c7*/c3 +
       /c7*/c2 +
       /c7*/c1 +
       /c7*/c0

c8.oe = _vramcs
/c8 := c9*c8*/c7*/c6*/c5*c4*c3*c2*c1*c0 +
       c8*c7*c6*c5*c4*c3*c2*c1*c0 +
       /c8*/c7 +
       /c8*/c6 +
       /c8*/c5 +
       /c8*/c4 +
       /c8*/c3 +
       /c8*/c2 +
       /c8*/c1 +
       /c8*/c0

c9.oe = _vramcs
/c9 := c9*c8*/c7*/c6*/c5*c4*c3*c2*c1*c0 +
       c9*c8*c7*c6*c5*c4*c3*c2*c1*c0 +
       /c9*/c8 +
       /c9*/c7 +
       /c9*/c6 +
       /c9*/c5 +
       /c9*/c4 +
       /c9*/c3 +
       /c9*/c2 +
       /c9*/c1 +
       /c9*/c0
```

```
; USB module interface, LCD interface, and interrupt request generator
;
; Generates control signals for USBMOD4 I/O.
; Generates enable signal for LCD.
; Provides a status register for USBMOD4 ready/transmit/receive status flags.
; ORs individual IRQ lines from all interrupt-generating devices to create IRQ for CPU.

; device memory map
; 3F00 - LCD data
; 3F01 - LCD control
; 3F10 - USB data
; 3F11 - USB status
; 3F12 - keyboard data
; 3F13 - keyboard status
; 3F20..3F2F - RTC (see rtc.s)
; 3F30 - video size
; 3F40 - video palette pixel address (write mode)
; 3F44 - video palette color value
; 3F48 - video palette pixel mask
; 3F4C - video palette pixel address (read mode, not supported)


CHIP device G22V10

q0=1 q1=2 _ld_mem=3 _hwsel_usbkybd=4 _hwsel_lcd=5 a0=6 a1=7 _usb_txf=8 _usb_rxf=9
    _usb_pen=10
    _irq_rtc=11 gnd=12
_irq_kybd=13 lcd_rs=14 lcd_run=15 _lcdin_g=16 lcd_e=17 _irq=18 d2=19 d1=20 d0=21 usb_wr=
    22
   _usb_rd=23 vcc=24

EQUATIONS

; LCD is configured write-only. Data is first stored in a dedicated register, then
    written
; to LCD on the next clock cycle. As wired, the LCD busy flag can't be read. Therefore
    the
; CPU must wait long enough before each LCD write to make sure the previous write
    operation
; has completed.
;
; LCD Operation Times
; clear,  home: 1.52ms
; other writes: 41us

/_lcdin_g = /_hwsel_lcd*/_ld_mem
lcd_run := /_lcdin_g
lcd_rs := /_lcdin_g*/a0 + _lcdin_g*lcd_rs
lcd_e = lcd_run*/q0 + lcd_run*q1

; _irq should only change on a clock boundary
/_irq := /_usb_rxf + /_irq_kybd + /_irq_rtc

; USB and Keyboard share a 16 byte device space. Map:
; XX00 USB data
; XX01 USB status
; XX10 Keyboard data
; XX11 Keyboard status

; USB read and write are both only active during the last quarter of the clock cycle.
usb_wr = /_hwsel_usbkybd*/a0*/a1*/_ld_mem*/q0*/q1
/_usb_rd = /_hwsel_usbkybd*/a0*/a1*_ld_mem*/q0*/q1

; 3 bit status regsiter for Keyboard/USB
d2.oe = /_hwsel_usbkybd*a0*_ld_mem
d2 = _usb_pen*/a1
```

```
d1.oe = /_hwsel_usbkybd*a0*_ld_mem
d1 = _usb_txf*/a1
d0.oe = /_hwsel_usbkybd*a0*_ld_mem
d0 = _usb_rxf*/a1 + _irq_kybd*a1
```

```
; Address decoder
;
; Decodes the 24-bit address, and generates enable signals for ROM, RAM, and hardware
; devices. The particular decode behavior implemented here determines the machine's memory
; map.
;
; Total address space is 16MB:
; $000000 - $003EFF is ROM (16K - 256 = 15.75K)
; $003F00 - $003FFF is hardware devices (16 device maps of 16 bytes each = 256 bytes)
; $004000 - $00FFFF is RAM (48K) for interrupt vector, ISR wrapper routine, and scratch
; $010000 - $07FFFF is RAM (512K - 64K = 448K)
; $080000 - $0FFFFF is a mirror of RAM (512K) allowing full contiguous access
; $100000 - $17FFFF is a mirror of ROM (512K) allowing full contiguous access
; $180000 - $1FFFFF is Aux RAM (not installed)
; $200000 - $207FFF is VRAM (32K)
; $208000 - $20FFFF is a mirror of VRAM
; $210000 - $FFFFFF is unused

CHIP decode G22V10

a8=1 a9=2 a10=3 a11=4 a12=5 a13=6 a14=7 a15=8 a16=9 a17=10 a18=11 gnd=12
a19=13 a20=14 a21=15 a22=16 a23=17 _vram_cs=18 _en_membuf=19 _ram1_cs=20 _ram0_cs=21
    _rom_cs=22 _hw_cs=23 vcc=24

EQUATIONS

; VRAM means [A23..A15] = 0010 0000 0
; Don't assert _vram_cs unless this is an actual memory access. This should really be
; the case for all the memory selects here, but it seems to cause problems when applied
; to the other memory.
/_vram_cs = /_en_membuf*/a23*/a22*a21*/a20*/a19*/a18*/a17*/a16*/a15

; RAM1 means [A23..A19] = 0001 1
/_ram1_cs = /a23*/a22*/a21*a20*a19

; RAM0 means [A23..A19] = 0000 0 and [A18..A14] != 000 00 or
 ;           [A23..19] = 0000 1 (banks 8-15)
/_ram0_cs = /a23*/a22*/a21*/a20*/a19*a18 +
            /a23*/a22*/a21*/a20*/a19*a17 +
            /a23*/a22*/a21*/a20*/a19*a16 +
            /a23*/a22*/a21*/a20*/a19*a15 +
            /a23*/a22*/a21*/a20*/a19*a14 +
            /a23*/a22*/a21*/a20*a19

; ROM means [A23..A14] = 0000 0000 00 and [A13..A8] != 11 1111 or
;           [A23..A17] = 0001 0 (banks 16-23)
/_rom_cs = /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a13 +
           /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a12 +
           /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a11 +
           /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a10 +
           /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a9 +
           /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a8 +
           /a23*/a22*/a21*a20*/a19

; hardware device means [A23..A8] = 0000 0000 0011 1111
/_hw_cs = /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*a13*a12*a11*a10*a9*a8
```

```
; Control system logic
;
; Stores previous carry for address computations, and interrupt enable flag.
; Determines which register the shared enable bit for Y and X7 should refer to.
; Generates timed control signals for the memory system and hardware devices.

CHIP control G22V10

q0=1 load3=2 spdir_iedata=3 resetnc=4 _ld_ie=5 _en_alu_yorx7=6 _alu_cout=7 q1=8 _ldmem=9
     _dralu=10 _en_alu_x=11 gnd=12
_en_alu_0=13 resetc=14 _resetc=15 _en_alu_xorx7or0=16 _en_membuf=17 _oe_mem=18 _we_mem=19
     _carry_a=20 _en_alu_y=21 _en_alu_x7=22 _ie=23 vcc=24

EQUATIONS

; _reset should only change on a clock boundary
_resetc := /resetnc
resetc := resetnc

_ie := /_resetc + _resetc*/_ld_ie*spdir_iedata + _resetc*_ld_ie*_ie

; Note: SPLO is not part of /LOAD3, so this can't use A for adding a relative offset to ↙
    the stack

_carry_a := load3*_carry_a + /load3*_alu_cout

; Y and X7 share a control bit: use X7 with address register destinations, otherwise use ↙
    Y.
; Note SPLO is not included here. _carry_a will not be loaded when loading SPLO.
_en_alu_x7 = _en_alu_yorx7 + load3
_en_alu_y = _en_alu_yorx7 + /load3

; X, X7, and 0 are all emitted by a single GAL with a shared enable flag.
_en_alu_xorx7or0 = _en_alu_x*_en_alu_yorx7*_en_alu_0 + _en_alu_x*load3*_en_alu_0

; Wait until after the first quarter clock cycle to assert _en_membuf, which will prevent
; the address decoder from selecting VRAM before then. This is quite possibly a bad idea,
; and will reduce the max clock speed of the machine.
/_en_membuf = /q0*_dralu + q1*_dralu + /q0*/_ldmem + q1*/_ldmem

_oe_mem = /_ldmem

; write-enable can only be active during the last quarter of the clock cycle
/_we_mem = /_ldmem*/q0*/q1
```

```
; AY-3-8910 programmable sound generator interface
;
; Generates bus singals for data transfer between CPU and PSG over multiple clock cycles.
;
; AUDIO_DATA     = 0XXX
; AUDIO_REGISTER = 1XXX
;
; bdir bc1  function
;  0    0   inactive
;  0    1   read from PSG
;  1    0   write to PSG
;  1    1   latch address
;
; bdir bc1  s1   s0   state
;  0    0    0    0   idle
;  0    0    0    1   setup data
;  1    0    0    0   data pulse 1
;  1    0    0    1   data pulse 2
;  1    0    1    0   data pulse 3
;  1    0    1    1   data pulse 4
;  1    1    0    0   latch address

CHIP audio G22V10

clk0=1 nc=2 nc=3 nc=4 nc=5 nc=6 nc=7 nc=8 nc=9 _audio_cs=10 a3=11 gnd=12
nc=13 bc1=14 bdir=15 nc=16 nc=17 nc=18 nc=19 nc=20 nc=21 s1=22 s0=23 vcc=24

EQUATIONS

bdir := /_audio_cs*a3 + /bdir*/bc1*/s1*s0 + bdir*/bc1*/s1*/s0 + bdir*/bc1*/s1*s0 + bdir*/↙
    bc1*s1*/s0

bc1 := /_audio_cs*a3

s0 := /_audio_cs*/a3 + bdir*/bc1*/s0

s1 := bdir*/bc1*/s1*s0 + bdir*/bc1*s1*/s0
```

```
; AR (address) register
;
; Holds one byte of the address register. Can be loaded or count up.

CHIP ar G22V10

; Note q outputs are not in order, because:
;  q6 has 15 product terms, and must be on pin 18 or 19
;  q5 has 13 product terms, and must be on pin 17-20
;  q4 and q7 have 11 product terms, and must be on pin 16-21
;  q3 has 9 product terms, and must be on pin 15-22

clk0=1 d0=2 d1=3 d2=4 d3=5 d4=6 d5=7 d6=8 d7=9 nc=10 _cnt_in=11 gnd=12
_oe=13 _ld=14 _cnt_out=15 q7=16 q5=17 q6=18 q4=19 q3=20 q2=21 q1=22 q0=23 vcc=24

EQUATIONS

q0.oe = /_oe
q1.oe = /_oe
q2.oe = /_oe
q3.oe = /_oe
q4.oe = /_oe
q5.oe = /_oe
q6.oe = /_oe
q7.oe = /_oe

/_cnt_out = /_cnt_in*q0*q1*q2*q3*q4*q5*q6*q7

q0 := _cnt_in*_ld*q0 + /_ld*d0 +
       _ld*/_cnt_in*/q0*/clk0

q1 := _cnt_in*_ld*q1 + /_ld*d1 +
       _ld*/_cnt_in*q0*/q1 +
       _ld*/_cnt_in*q1*/q0

q2 := _cnt_in*_ld*q2 + /_ld*d2 +
       _ld*/_cnt_in*q0*q1*/q2 +
       _ld*/_cnt_in*q2*/q1 +
       _ld*/_cnt_in*q2*/q0

q3 := _cnt_in*_ld*q3 + /_ld*d3 +
       _ld*/_cnt_in*q0*q1*q2*/q3 +
       _ld*/_cnt_in*q3*/q2 +
       _ld*/_cnt_in*q3*/q1 +
       _ld*/_cnt_in*q3*/q0

q4 := _cnt_in*_ld*q4 + /_ld*d4 +
       _ld*/_cnt_in*q0*q1*q2*q3*/q4 +
       _ld*/_cnt_in*q4*/q3 +
       _ld*/_cnt_in*q4*/q2 +
       _ld*/_cnt_in*q4*/q1 +
       _ld*/_cnt_in*q4*/q0

q5 := _cnt_in*_ld*q5 + /_ld*d5 +
       _ld*/_cnt_in*q0*q1*q2*q3*q4*/q5 +
       _ld*/_cnt_in*q5*/q4 +
       _ld*/_cnt_in*q5*/q3 +
       _ld*/_cnt_in*q5*/q2 +
       _ld*/_cnt_in*q5*/q1 +
       _ld*/_cnt_in*q5*/q0

q6 := _cnt_in*_ld*q6 + /_ld*d6 +
       _ld*/_cnt_in*q0*q1*q2*q3*q4*q5*/q6 +
       _ld*/_cnt_in*q6*/q5 +
       _ld*/_cnt_in*q6*/q4 +
       _ld*/_cnt_in*q6*/q3 +
```

```
        _ld*/_cnt_in*q6*/q2 +
        _ld*/_cnt_in*q6*/q1 +
        _ld*/_cnt_in*q6*/q0

q7 := _cnt_in*_ld*q7 + /_ld*d7 +
        _ld*/_cnt_in*q0*q1*q2*q3*q4*q5*q6*/q7 +
        _ld*/_cnt_in*q7*/q6 +
        _ld*/_cnt_in*q7*/q5 +
        _ld*/_cnt_in*q7*/q4 +
        _ld*/_cnt_in*q7*/q3 +
        _ld*/_cnt_in*q7*/q2 +
        _ld*/_cnt_in*q7*/q1 +
        _ld*/_cnt_in*q7*/q0
```

```
; ALU condition codes
;
; Computes, stores, and shifts the condition codes from the ALU.
; Carry (C) active low
; Negative (N) active high
; Zero/Equal (z) active low
; Overflow (V) active low
;
; f[7..0] is the ALU result
; func1 is bit 1 of the ALU function that performed. 1 means it was a subtraction.
; a7 and b7 are the MSB's of the left and right ALU operands.
; regx3 is bit 3 from the X register
;
; s1 s0 operation
;  0  0 no change
;  0  1 load N and Z
;  1  0 shift right
;  1  1 load all


CHIP alucc G22V10

q0=1 f0=2 f1=3 f2=4 f3=5 f4=6 f5=7 f6=8 f7=9 func1=10 a7=11 gnd=12
b7=13 _cin=14 s0=15 s1=16 regx3=17 _ld_a=18 _en_alu_t=19 _cc_v=20 _cc_z=21 cc_n=22 _cc_c=↙
    23 vcc=24

EQUATIONS

/_cc_c := /s1*/_cc_c +
         s1*/s0*/cc_n +
         s1*s0*/_cin

  cc_n := /s1*/s0*cc_n +
         s1*/s0*_cc_z +
         s0*f7

/_cc_z := /s1*/s0*/_cc_z +
          s1*/s0*/_cc_v +
          s0*/f0*/f1*/f2*/f3*/f4*/f5*/f6*/f7

; load V only when op is A <- ? * T
/_cc_v := /s1*/_cc_v +               ; hold
          s1*/s0*/regx3 +            ; shift right

          _ld_a*s1*s0*/_cc_v +       ; ignore load
          _en_alu_t*s1*s0*/_cc_v +   ; ignore load

          /_ld_a*/_en_alu_t*s1*s0*/a7*/b7*f7*/func1 + ; load
          /_ld_a*/_en_alu_t*s1*s0*a7*b7*/f7*/func1  + ; load
          /_ld_a*/_en_alu_t*s1*s0*/a7*b7*f7*func1   + ; load
          /_ld_a*/_en_alu_t*s1*s0*a7*/b7*/f7*func1    ; load
```

```
; A buffer that passes through a byte unmodified, or copies
; input bit 7 to all the output bits, or outputs zero.
;
; Used for X register, X7 pseudo-register, and Zero pseudo-register.
;
; Uses 22V8 Complex mode.

CHIP xorx7or0 G20V8

d0=1 d1=2 d2=3 d3=4 d4=5 d5=6 d6=7 d7=8 nc=9 nc=10 _en_xorx7or0=11 gnd=12
_en_0=13 _en_x=14 q0=15 q1=16 q2=17 q3=18 q4=19 q5=20 q6=21 q7=22 _en_x7=23 vcc=24

EQUATIONS

q0.oe = /_en_xorx7or0
q0 = /_en_x*d0 + /_en_x7*d7

q1.oe = /_en_xorx7or0
q1 = /_en_x*d1 + /_en_x7*d7

q2.oe = /_en_xorx7or0
q2 = /_en_x*d2 + /_en_x7*d7

q3.oe = /_en_xorx7or0
q3 = /_en_x*d3 + /_en_x7*d7

q4.oe = /_en_xorx7or0
q4 = /_en_x*d4 + /_en_x7*d7

q5.oe = /_en_xorx7or0
q5= /_en_x*d5 + /_en_x7*d7

q6.oe = /_en_xorx7or0
q6 = /_en_x*d6 + /_en_x7*d7

q7.oe = /_en_xorx7or0
q7 = /_en_x*d7 + /_en_x7*d7
```

```
; Video mode register
;
; Loads and stores a byte of video mode setttings from VRAM.

CHIP vidmode G22V10

; dotclk is 25.175 MHz = 39.72 ns period

dotclk=1 d0=2 d1=3 d2=4 d3=5 d4=6 d5=7 d6=8 d7=9 _ldmode=10 nc=11 gnd=12
nc=13 oe_pal=14 _oe_charrom=15 rmask=16 text=17 bpp1=18 bpp0=19 pal3=20 pal2=21 pal1=22
    pal0=23 vcc=24

EQUATIONS

pal0.oe = oe_pal
pal0 := /_ldmode*d0 +
        _ldmode*pal0

pal1.oe = oe_pal
pal1 := /_ldmode*d1 +
        _ldmode*pal1

pal2.oe = oe_pal
pal2 := /_ldmode*d2 +
        _ldmode*pal2

pal3.oe = oe_pal
pal3 := /_ldmode*d3 +
        _ldmode*pal3

bpp0 := /_ldmode*d4 +
        _ldmode*bpp0

bpp1 := /_ldmode*d5 +
        _ldmode*bpp1

text := /_ldmode*d6 +
        _ldmode*text

rmask:= /_ldmode*d7 +
        _ldmode*rmask

_oe_charrom = /text

oe_pal = /bpp0 + /bpp1
```

```
; Low Vertical pixel counter
;
; Counts the low bits of the number of lines in a frame, and outputs masked versions of ↙
    the bits.

CHIP vertlo G22V10

; dotclk is 25.175 MHz = 39.72 ns period
; 525 lines per frame for 640x480 video, 480 lines in visible video region, negative     ↙
    vsync
;
; vsize  lines  mode
;  00     240   per line
;  01     240   per frame
;  10     480   per line
;  11     480   per frame

dotclk=1 _vramcs=2 rmask=3 text=4 newframe=5 _newline=6 _hwsel_video=7 nc=8 _we_mem=9 d6=↙
    10 d7=11 gnd=12
nc=13 vsize1=14 vsize0=15 _we_vram=16 r3=17 r2=18 r1=19 r0=20 r3m=21 r2m=22 r1m=23 vcc=24

EQUATIONS

/_we_vram = /_vramcs*/_we_mem

; These registers are clocked with a different clock than the CPU driving the bus!
; Hopefully this will still work OK?
vsize0 := _we_mem*vsize0 +
          _hwsel_video*vsize0 +
          /_hwsel_video*/_we_mem*d6

vsize1 := _we_mem*vsize1 +
          _hwsel_video*vsize1 +
          /_hwsel_video*/_we_mem*d7

r1m.oe = _vramcs
/r1m := rmask +
        text +
        newframe*/_newline +
        _newline*/r1 +
        /_newline*r1*r0 +
        /_newline*/r1*/r0

r2m.oe = _vramcs
/r2m := text +
        newframe*/_newline +
        _newline*/r2 +
        /_newline*r2*r1*r0 +
        /_newline*/r2*/r1 +
        /_newline*/r2*/r0

r3m.oe = _vramcs
/r3m := text +
        newframe*/_newline +
        _newline*/r3 +
        /_newline*r3*r2*r1*r0 +
        /_newline*/r3*/r2 +
        /_newline*/r3*/r1 +
        /_newline*/r3*/r0

/r0 := newframe*/_newline +
       _newline*/r0 +
       /_newline*r0

/r1 := newframe*/_newline +
       _newline*/r1 +
```

```
       /_newline*r1*r0 +
       /_newline*/r1*/r0

/r2 := newframe*/_newline +
       _newline*/r2 +
       /_newline*r2*r1*r0 +
       /_newline*/r2*/r1 +
       /_newline*/r2*/r0

/r3 := newframe*/_newline +
       _newline*/r3 +
       /_newline*r3*r2*r1*r0 +
       /_newline*/r3*/r2 +
       /_newline*/r3*/r1 +
       /_newline*/r3*/r0
```

```
; High vertical pixel counter
;
; Counts the high bits of the number of lines in a frame, and generates the vsync, vblank
    , and blank
; signals.

CHIP verthi G22V10

; dotclk is 25.175 MHz = 39.72 ns period
; 525 lines per frame for 640x480 video, 480 lines in visible video region, negative
    vsync
;
; vsize  lines  mode
;  00     240   per line
;  01     240   per frame
;  10     480   per line
;  11     480   per frame

dotclk=1 _cpuaccess=2 r0=3 r1=4 r2=5 r3=6 _newline=7 _hblank=8 vsize0=9 _hwsel_video=10
    _oe_mem=11 gnd=12
nc=13 _blank=14 _vsync=15 _vblank=16 r9=17 r8=18 r7=19 r6=20 r5=21 r4=22 newframe=23 vcc=
    24

EQUATIONS

; 524 = 10 0000 1100
newframe = r9*/r8*/r7*/r6*/r5*/r4*r3*r2*/r1*/r0

; _blank = _hblank*_vblank*_cpuaccess ; try this to blank the display during CPU access
_blank = _hblank*_vblank*_cpuaccess

; at 640x480, _vsync is low for lines 490 and 491, high oherwise
; 01 1110 101x - 490-491 (low)
_vsync := r9 + /r8 + /r7 + /r6 + /r5 + r4 + /r3 + r2 + /r1

; _vblank is low for lines 480+
; 480 = 01 1110 0000
_vblank.oe = /_hwsel_video*/_oe_mem
/_vblank := /r9*r8*r7*r6*r5 +              ; 480-511
            r9                            ; 512-1023

r4.oe = _cpuaccess
/r4 := newframe*/_newline +
       _newline*/r4 +
       /_newline*r4*r3*r2*r1*r0 +
       /_newline*/r4*/r3 +
       /_newline*/r4*/r2 +
       /_newline*/r4*/r1 +
       /_newline*/r4*/r0

r5.oe = _cpuaccess
/r5 := newframe*/_newline +
       _newline*/r5 +
       /_newline*r5*r4*r3*r2*r1*r0 +
       /_newline*/r5*/r4 +
       /_newline*/r5*/r3 +
       /_newline*/r5*/r2 +
       /_newline*/r5*/r1 +
       /_newline*/r5*/r0

r6.oe = _cpuaccess
/r6 := newframe*/_newline +
       _newline*/r6 +
       /_newline*r6*r5*r4*r3*r2*r1*r0 +
       /_newline*/r6*/r5 +
       /_newline*/r6*/r4 +
```

```
       /_newline*/r6*/r3 +
       /_newline*/r6*/r2 +
       /_newline*/r6*/r1 +
       /_newline*/r6*/r0

r7.oe = _cpuaccess
/r7 := newframe*/_newline +
       _newline*/r7 +
       /_newline*r7*r6*r5*r4*r3*r2*r1*r0 +
       /_newline*/r7*/r6 +
       /_newline*/r7*/r5 +
       /_newline*/r7*/r4 +
       /_newline*/r7*/r3 +
       /_newline*/r7*/r2 +
       /_newline*/r7*/r1 +
       /_newline*/r7*/r0

r8.oe = _cpuaccess
/r8 := newframe*/_newline +
       _newline*/r8 +
       /_newline*r8*r7*r6*r5*r4*r3*r2*r1*r0 +
       /_newline*/r8*/r7 +
       /_newline*/r8*/r6 +
       /_newline*/r8*/r5 +
       /_newline*/r8*/r4 +
       /_newline*/r8*/r3 +
       /_newline*/r8*/r2 +
       /_newline*/r8*/r1 +
       /_newline*/r8*/r0

/r9 := newframe*/_newline +
       _newline*/r9 +
       /_newline*r9*r8*r7*r6*r5*r4*r3*r2*r1*r0 +
       /_newline*/r9*/r8 +
       /_newline*/r9*/r7 +
       /_newline*/r9*/r6 +
       /_newline*/r9*/r5 +
       /_newline*/r9*/r4 +
       /_newline*/r9*/r3 +
       /_newline*/r9*/r2 +
       /_newline*/r9*/r1 +
       /_newline*/r9*/r0
```

```
; SP (stack pointer) register
;
; Holds one byte of the stack pointer. Can be loaded or count up/down.

CHIP sp G22V10

; Note q outputs are not in order, because:
;  q6 has 15 product terms, and must be on pin 18 or 19
;  q5 has 13 product terms, and must be on pin 17-20
;  q4 and q7 have 11 product terms, and must be on pin 16-21
;  q3 has 9 product terms, and must be on pin 15-22

clk0=1 d0=2 d1=3 d2=4 d3=5 d4=6 d5=7 d6=8 d7=9 dir=10 _cnt_in=11 gnd=12
_oe=13 _ld=14 _cnt_out=15 q7=16 q5=17 q6=18 q4=19 q3=20 q2=21 q1=22 q0=23 vcc=24

EQUATIONS

q0.oe = /_oe
q1.oe = /_oe
q2.oe = /_oe
q3.oe = /_oe
q4.oe = /_oe
q5.oe = /_oe
q6.oe = /_oe
q7.oe = /_oe

/_cnt_out = _ld*/_cnt_in*dir*q0*q1*q2*q3*q4*q5*q6*q7 +
            _ld*/_cnt_in*/dir*/q0*/q1*/q2*/q3*/q4*/q5*/q6*/q7

/q0 := _cnt_in*_ld*/q0 + /_ld*/d0 +
       _ld*/_cnt_in*q0

/q1 := _cnt_in*_ld*/q1 + /_ld*/d1 +
       _ld*/_cnt_in*dir*q0*q1 +
       _ld*/_cnt_in*dir*/q0*/q1 +
       _ld*/_cnt_in*/dir*q0*/q1 +
       _ld*/_cnt_in*/dir*/q0*q1

/q2 := _cnt_in*_ld*/q2 + /_ld*/d2 +
       _ld*/_cnt_in*dir*q0*q1*q2 +
       _ld*/_cnt_in*dir*/q2*/q1 +
       _ld*/_cnt_in*dir*/q2*/q0 +
       _ld*/_cnt_in*/dir*q2*/q1*/q0 +
       _ld*/_cnt_in*/dir*/q2*q1 +
       _ld*/_cnt_in*/dir*/q2*q0

/q3 := _cnt_in*_ld*/q3 + /_ld*/d3 +
       _ld*/_cnt_in*dir*q0*q1*q2*q3 +
       _ld*/_cnt_in*dir*/q3*/q2 +
       _ld*/_cnt_in*dir*/q3*/q1 +
       _ld*/_cnt_in*dir*/q3*/q0 +
       _ld*/_cnt_in*/dir*q3*/q2*/q1*/q0 +
       _ld*/_cnt_in*/dir*/q3*q2 +
       _ld*/_cnt_in*/dir*/q3*q1 +
       _ld*/_cnt_in*/dir*/q3*q0

/q4 := _cnt_in*_ld*/q4 + /_ld*/d4 +
       _ld*/_cnt_in*dir*q0*q1*q2*q3*q4 +
       _ld*/_cnt_in*dir*/q4*/q3 +
       _ld*/_cnt_in*dir*/q4*/q2 +
       _ld*/_cnt_in*dir*/q4*/q1 +
       _ld*/_cnt_in*dir*/q4*/q0 +
       _ld*/_cnt_in*/dir*q4*/q3*/q2*/q1*/q0 +
       _ld*/_cnt_in*/dir*/q4*q3 +
       _ld*/_cnt_in*/dir*/q4*q2 +
       _ld*/_cnt_in*/dir*/q4*q1 +
```

```
       _ld*/_cnt_in*/dir*/q4*q0

/q5 := _cnt_in*_ld*/q5 + /_ld*/d5 +
       _ld*/_cnt_in*dir*q0*q1*q2*q3*q4*q5 +
       _ld*/_cnt_in*dir*/q5*/q4 +
       _ld*/_cnt_in*dir*/q5*/q3 +
       _ld*/_cnt_in*dir*/q5*/q2 +
       _ld*/_cnt_in*dir*/q5*/q1 +
       _ld*/_cnt_in*dir*/q5*/q0 +
       _ld*/_cnt_in*/dir*q5*/q4*/q3*/q2*/q1*/q0 +
       _ld*/_cnt_in*/dir*/q5*q4 +
       _ld*/_cnt_in*/dir*/q5*q3 +
       _ld*/_cnt_in*/dir*/q5*q2 +
       _ld*/_cnt_in*/dir*/q5*q1 +
       _ld*/_cnt_in*/dir*/q5*q0

/q6 := _cnt_in*_ld*/q6 + /_ld*/d6 +
       _ld*/_cnt_in*dir*q0*q1*q2*q3*q4*q5*q6 +
       _ld*/_cnt_in*dir*/q6*/q5 +
       _ld*/_cnt_in*dir*/q6*/q4 +
       _ld*/_cnt_in*dir*/q6*/q3 +
       _ld*/_cnt_in*dir*/q6*/q2 +
       _ld*/_cnt_in*dir*/q6*/q1 +
       _ld*/_cnt_in*dir*/q6*/q0 +
       _ld*/_cnt_in*/dir*q6*/q5*/q4*/q3*/q2*/q1*/q0 +
       _ld*/_cnt_in*/dir*/q6*q5 +
       _ld*/_cnt_in*/dir*/q6*q4 +
       _ld*/_cnt_in*/dir*/q6*q3 +
       _ld*/_cnt_in*/dir*/q6*q2 +
       _ld*/_cnt_in*/dir*/q6*q1 +
       _ld*/_cnt_in*/dir*/q6*q0

q7 := _cnt_in*_ld*q7 + /_ld*d7 +
      _ld*/_cnt_in*dir*/q7*q6*q5*q4*q3*q2*q1*q0 +
      _ld*/_cnt_in*dir*q7*_cnt_out +
      _ld*/_cnt_in*/dir*/q7*/q6*/q5*/q4*/q3*/q2*/q1*/q0 +
      _ld*/_cnt_in*/dir*q7*q6 +
      _ld*/_cnt_in*/dir*q7*q5 +
      _ld*/_cnt_in*/dir*q7*q4 +
      _ld*/_cnt_in*/dir*q7*q3 +
      _ld*/_cnt_in*/dir*q7*q2 +
      _ld*/_cnt_in*/dir*q7*q1 +
      _ld*/_cnt_in*/dir*q7*q0
```

```
; Video data bit shifter
;
; Loads and shifts a byte from VRAM, for input to the palette.

CHIP shifter G22V10

; dotclk is 25.175 MHz = 39.72 ns period

dotclk=1 d0=2 d1=3 d2=4 d3=5 d4=6 d5=7 d6=8 d7=9 c0=10 _ldbyte=11 gnd=12
bpp0=13 bpp1=14 q0=15 q7=16 q6=17 q5=18 q4=19 q3=20 q2=21 q1=22 invert=23 vcc=24

; bpp   definition
; 00    512@1bpp, shift 1 bit every column, don't output q4-q7
; 01    512@2bpp, shift 2 bits every column, don't output q4-q7
; 10    256@4bpp, shift 4 bits every 2 columns, don't output q4-q7
; 11    128@8bpp, don't shift, output q4-q7

EQUATIONS

q0 := /_ldbyte*/invert*/bpp1*/bpp0*d7 + ; Load and reverse when _ldbyte=0, invert=0, hres
    =00.
      /_ldbyte*invert*/bpp1*/bpp0*/d7 + ; Load, invert, and reverse when _ldbyte=0,
    invert=1, hres=00.
      /_ldbyte*/bpp1*bpp0*d6 +          ; Load and swap bit pairs when _ldbyte=0, hres=01
    .
      /_ldbyte*bpp1*/bpp0*d4 +          ; Load and swap nibbles when _ldbyte=0, hres=10.
      /_ldbyte*bpp1*bpp0*d0 +           ; Load straight through when _ldbytye=0, hres=11.
      _ldbyte*/bpp1*/bpp0*q1 +      ; If not loading, always shift by 1 when hres=00.
      _ldbyte*/bpp1*bpp0*q2 +       ; If not loading, always shift by 2 when hres=01.
      _ldbyte*bpp1*/bpp0*c0*q4 +    ; If not loading, shift by 4 when hres=10 and c0=1.
      _ldbyte*bpp1*/bpp0*/c0*q0 +       ; If not loading, maintain present value when
    hres=10 and c0=0.
      _ldbyte*bpp1*bpp0*q0          ; If not loading, maintain present value when hres=11
    .


q1 := /_ldbyte*/invert*/bpp1*/bpp0*d6 +
      /_ldbyte*invert*/bpp1*/bpp0*/d6 +
      /_ldbyte*/bpp1*bpp0*d7 +
      /_ldbyte*bpp1*/bpp0*d5 +
      /_ldbyte*bpp1*bpp0*d1 +
      _ldbyte*/bpp1*/bpp0*q2 +
      _ldbyte*/bpp1*bpp0*q3 +
      _ldbyte*bpp1*/bpp0*c0*q5 +
      _ldbyte*bpp1*/bpp0*/c0*q1 +
      _ldbyte*bpp1*bpp0*q1

q2 := /_ldbyte*/invert*/bpp1*/bpp0*d5 +
      /_ldbyte*invert*/bpp1*/bpp0*/d5 +
      /_ldbyte*/bpp1*bpp0*d4 +
      /_ldbyte*bpp1*/bpp0*d6 +
      /_ldbyte*bpp1*bpp0*d2 +
      _ldbyte*/bpp1*/bpp0*q3 +
      _ldbyte*/bpp1*bpp0*q4 +
      _ldbyte*bpp1*/bpp0*c0*q6 +
      _ldbyte*bpp1*/bpp0*/c0*q2 +
      _ldbyte*bpp1*bpp0*q2


q3 := /_ldbyte*/invert*/bpp1*/bpp0*d4 +
      /_ldbyte*invert*/bpp1*/bpp0*/d4 +
      /_ldbyte*/bpp1*bpp0*d5 +
      /_ldbyte*bpp1*/bpp0*d7 +
      /_ldbyte*bpp1*bpp0*d3 +
      _ldbyte*/bpp1*/bpp0*q4 +
      _ldbyte*/bpp1*bpp0*q5 +
```

```
      _ldbyte*bpp1*/bpp0*c0*q7 +
      _ldbyte*bpp1*/bpp0*/c0*q3 +
      _ldbyte*bpp1*bpp0*q3

q4.oe = bpp0*bpp1
q4 := /_ldbyte*/invert*/bpp1*/bpp0*d3 +
      /_ldbyte*invert*/bpp1*/bpp0*/d3 +
      /_ldbyte*/bpp1*bpp0*d2 +
      /_ldbyte*bpp1*/bpp0*d0 +
      /_ldbyte*bpp1*bpp0*d4 +
      _ldbyte*/bpp1*/bpp0*q5 +
      _ldbyte*/bpp1*bpp0*q6 +
      _ldbyte*bpp1*/bpp0*/c0*q4 +
      _ldbyte*bpp1*bpp0*q4

q5.oe = bpp0*bpp1
q5 := /_ldbyte*/invert*/bpp1*/bpp0*d2 +
      /_ldbyte*invert*/bpp1*/bpp0*/d2 +
      /_ldbyte*/bpp1*bpp0*d3 +
      /_ldbyte*bpp1*/bpp0*d1 +
      /_ldbyte*bpp1*bpp0*d5 +
      _ldbyte*/bpp1*/bpp0*q6 +
      _ldbyte*/bpp1*bpp0*q7 +
      _ldbyte*bpp1*/bpp0*/c0*q5 +
      _ldbyte*bpp1*bpp0*q5

q6.oe = bpp0*bpp1
q6 := /_ldbyte*/invert*/bpp1*/bpp0*d1 +
      /_ldbyte*invert*/bpp1*/bpp0*/d1 +
      /_ldbyte*/bpp1*bpp0*d0 +
      /_ldbyte*bpp1*/bpp0*d2 +
      /_ldbyte*bpp1*bpp0*d6 +
      _ldbyte*/bpp1*/bpp0*q7+
      _ldbyte*bpp1*/bpp0*/c0*q6 +
      _ldbyte*bpp1*bpp0*q6

q7.oe = bpp0*bpp1
q7 := /_ldbyte*/invert*/bpp1*/bpp0*d0 +
      /_ldbyte*invert*/bpp1*/bpp0*/d0 +
      /_ldbyte*/bpp1*bpp0*d1 +
      /_ldbyte*bpp1*/bpp0*d3 +
      /_ldbyte*bpp1*bpp0*d7 +
      _ldbyte*bpp1*/bpp0*/c0*q7 +
      _ldbyte*bpp1*bpp0*q7
```

```
; Generic 8-bit data register with load enable and output enable
;
; Used for the A, Y, and T registers
;
; Uses 22V8 registered mode. Pin 13 is hard-wired as an output enable in this mode.

CHIP reg G20V8

clk0=1 d0=2 d1=3 d2=4 d3=5 d4=6 d5=7 d6=8 d7=9 _ld=10 nc=11 gnd=12
_oe=13 nc=14 q0=15 q1=16 q2=17 q3=18 q4=19 q5=20 q6=21 q7=22 nc=23 vcc=24

EQUATIONS

q0 := /_ld*d0 + _ld*q0

q1 := /_ld*d1 + _ld*q1

q2 := /_ld*d2 + _ld*q2

q3 := /_ld*d3 + _ld*q3

q4 := /_ld*d4 + _ld*q4

q5 := /_ld*d5 + _ld*q5

q6 := /_ld*d6 + _ld*q6

q7 := /_ld*d7 + _ld*q7
```

```
; PC (program counter) register, LO and HI bytes (bits 0 to 15).
;
; Holds one byte of the program counter. Can be reset, loaded, or count up.

CHIP pclohi G22V10

clk0=1 d0=2 d1=3 d2=4 d3=5 d4=6 d5=7 d6=8 d7=9 _reset=10 _cnt_in=11 gnd=12
_oe=13 _ld=14 _cnt_out=15 q7=16 q5=17 q6=18 q4=19 q3=20 q2=21 q1=22 q0=23 vcc=24

EQUATIONS

; Output is also enabled when reset is active.
; _adrsel_pc and _reset are combined externally to create _oe.
q0.oe = /_oe
q1.oe = /_oe
q2.oe = /_oe
q3.oe = /_oe
q4.oe = /_oe
q5.oe = /_oe
q6.oe = /_oe
q7.oe = /_oe

/_cnt_out = /_cnt_in*q0*q1*q2*q3*q4*q5*q6*q7

q0 := _reset*_cnt_in*_ld*q0 + _reset*/_ld*d0 +
      _reset*_ld*/_cnt_in*/q0*/clk0

/q1 := /_reset + _reset*_cnt_in*_ld*/q1 + _reset*/_ld*/d1 +
      _reset*_ld*/_cnt_in*q0*q1 +
      _reset*_ld*/_cnt_in*/q0*/q1

/q2 := /_reset + _reset*_cnt_in*_ld*/q2 + _reset*/_ld*/d2 +
      _reset*_ld*/_cnt_in*q0*q1*q2 +
      _reset*_ld*/_cnt_in*/q2*/q1 +
      _reset*_ld*/_cnt_in*/q2*/q0

/q3 := /_reset + _reset*_cnt_in*_ld*/q3 + _reset*/_ld*/d3 +
      _reset*_ld*/_cnt_in*q0*q1*q2*q3 +
      _reset*_ld*/_cnt_in*/q3*/q2 +
      _reset*_ld*/_cnt_in*/q3*/q1 +
      _reset*_ld*/_cnt_in*/q3*/q0

/q4 := /_reset + _reset*_cnt_in*_ld*/q4 + _reset*/_ld*/d4 +
      _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4 +
      _reset*_ld*/_cnt_in*/q4*/q3 +
      _reset*_ld*/_cnt_in*/q4*/q2 +
      _reset*_ld*/_cnt_in*/q4*/q1 +
      _reset*_ld*/_cnt_in*/q4*/q0

/q5 := /_reset + _reset*_cnt_in*_ld*/q5 + _reset*/_ld*/d5 +
      _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5 +
      _reset*_ld*/_cnt_in*/q5*/q4 +
      _reset*_ld*/_cnt_in*/q5*/q3 +
      _reset*_ld*/_cnt_in*/q5*/q2 +
      _reset*_ld*/_cnt_in*/q5*/q1 +
      _reset*_ld*/_cnt_in*/q5*/q0

/q6 := /_reset + _reset*_cnt_in*_ld*/q6 + _reset*/_ld*/d6 +
      _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5*q6 +
      _reset*_ld*/_cnt_in*/q6*/q5 +
      _reset*_ld*/_cnt_in*/q6*/q4 +
      _reset*_ld*/_cnt_in*/q6*/q3 +
      _reset*_ld*/_cnt_in*/q6*/q2 +
      _reset*_ld*/_cnt_in*/q6*/q1 +
      _reset*_ld*/_cnt_in*/q6*/q0
```

```
/q7 := /_reset + _reset*_cnt_in*_ld*/q7 + _reset*/_ld*/d7 +
       _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5*q6*q7 +
       _reset*_ld*/_cnt_in*/q7*/q6 +
       _reset*_ld*/_cnt_in*/q7*/q5 +
       _reset*_ld*/_cnt_in*/q7*/q4 +
       _reset*_ld*/_cnt_in*/q7*/q3 +
       _reset*_ld*/_cnt_in*/q7*/q2 +
       _reset*_ld*/_cnt_in*/q7*/q1 +
       _reset*_ld*/_cnt_in*/q7*/q0
```

```
; PC (program counter) register, BANK byte (bits 16 to 23).
;
; Holds one byte of the program counter. Can be reset, loaded, or count up.

CHIP pcbank G22V10

clk0=1 d0=2 d1=3 d2=4 d3=5 d4=6 d5=7 d6=8 d7=9 _reset=10 _cnt_in=11 gnd=12
_adrsel_pc=13 _ld=14 _oe_out=15 q7=16 q5=17 q6=18 q4=19 q3=20 q2=21 q1=22 q0=23 vcc=24

EQUATIONS

; Output is also enabled when reset is active.
; _adrsel_pc and _reset are combined here to create _oe.
_oe_out = _adrsel_pc*_reset

q0.oe = /_oe_out
q1.oe = /_oe_out
q2.oe = /_oe_out
q3.oe = /_oe_out
q4.oe = /_oe_out
q5.oe = /_oe_out
q6.oe = /_oe_out
q7.oe = /_oe_out

/q0 := /_reset + _reset*_cnt_in*_ld*/q0 + _reset*/_ld*/d0 +
       _reset*_ld*/_cnt_in*q0

/q1 := /_reset + _reset*_cnt_in*_ld*/q1 + _reset*/_ld*/d1 +
       _reset*_ld*/_cnt_in*q0*q1 +
       _reset*_ld*/_cnt_in*/q0*/q1

/q2 := /_reset + _reset*_cnt_in*_ld*/q2 + _reset*/_ld*/d2 +
       _reset*_ld*/_cnt_in*q0*q1*q2 +
       _reset*_ld*/_cnt_in*/q2*/q1 +
       _reset*_ld*/_cnt_in*/q2*/q0

/q3 := /_reset + _reset*_cnt_in*_ld*/q3 + _reset*/_ld*/d3 +
       _reset*_ld*/_cnt_in*q0*q1*q2*q3 +
       _reset*_ld*/_cnt_in*/q3*/q2 +
       _reset*_ld*/_cnt_in*/q3*/q1 +
       _reset*_ld*/_cnt_in*/q3*/q0

/q4 := /_reset + _reset*_cnt_in*_ld*/q4 + _reset*/_ld*/d4 +
       _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4 +
       _reset*_ld*/_cnt_in*/q4*/q3 +
       _reset*_ld*/_cnt_in*/q4*/q2 +
       _reset*_ld*/_cnt_in*/q4*/q1 +
       _reset*_ld*/_cnt_in*/q4*/q0

/q5 := /_reset + _reset*_cnt_in*_ld*/q5 + _reset*/_ld*/d5 +
       _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5 +
       _reset*_ld*/_cnt_in*/q5*/q4 +
       _reset*_ld*/_cnt_in*/q5*/q3 +
       _reset*_ld*/_cnt_in*/q5*/q2 +
       _reset*_ld*/_cnt_in*/q5*/q1 +
       _reset*_ld*/_cnt_in*/q5*/q0

/q6 := /_reset + _reset*_cnt_in*_ld*/q6 + _reset*/_ld*/d6 +
       _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5*q6 +
       _reset*_ld*/_cnt_in*/q6*/q5 +
       _reset*_ld*/_cnt_in*/q6*/q4 +
       _reset*_ld*/_cnt_in*/q6*/q3 +
       _reset*_ld*/_cnt_in*/q6*/q2 +
       _reset*_ld*/_cnt_in*/q6*/q1 +
       _reset*_ld*/_cnt_in*/q6*/q0
```

```
/q7 := /_reset + _reset*_cnt_in*_ld*/q7 + _reset*/_ld*/d7 +
       _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5*q6*q7 +
       _reset*_ld*/_cnt_in*/q7*/q6 +
       _reset*_ld*/_cnt_in*/q7*/q5 +
       _reset*_ld*/_cnt_in*/q7*/q4 +
       _reset*_ld*/_cnt_in*/q7*/q3 +
       _reset*_ld*/_cnt_in*/q7*/q2 +
       _reset*_ld*/_cnt_in*/q7*/q1 +
       _reset*_ld*/_cnt_in*/q7*/q0
```

```
; opcode register
;
; Stores current opcode. Clears opcode at the start of a new instruction, when an
; interrupt is pending. Loads opcode when _reset is low.
;
; _ie is interrupt enable


CHIP opcode G22V10

clk0=1 d0=2 d1=3 d2=4 d3=5 d4=6 d5=7 d6=8 d7=9 _ldop=10 _ie=11 gnd=12
_irq=13 _reset=14 nc=15 q7=16 q6=17 q5=18 q4=19 q3=20 q2=21 q1=22 q0=23 vcc=24

EQUATIONS

q0 := /_reset*d0 + _reset*/_ldop*_irq*d0 + _reset*/_ldop*_ie*d0 + _reset*_ldop*q0
q1 := /_reset*d1 + _reset*/_ldop*_irq*d1 + _reset*/_ldop*_ie*d1 + _reset*_ldop*q1
q2 := /_reset*d2 + _reset*/_ldop*_irq*d2 + _reset*/_ldop*_ie*d2 + _reset*_ldop*q2
q3 := /_reset*d3 + _reset*/_ldop*_irq*d3 + _reset*/_ldop*_ie*d3 + _reset*_ldop*q3
q4 := /_reset*d4 + _reset*/_ldop*_irq*d4 + _reset*/_ldop*_ie*d4 + _reset*_ldop*q4
q5 := /_reset*d5 + _reset*/_ldop*_irq*d5 + _reset*/_ldop*_ie*d5 + _reset*_ldop*q5
q6 := /_reset*d6 + _reset*/_ldop*_irq*d6 + _reset*/_ldop*_ie*d6 + _reset*_ldop*q6
q7 := /_reset*d7 + _reset*/_ldop*_irq*d7 + _reset*/_ldop*_ie*d7 + _reset*_ldop*q7
```