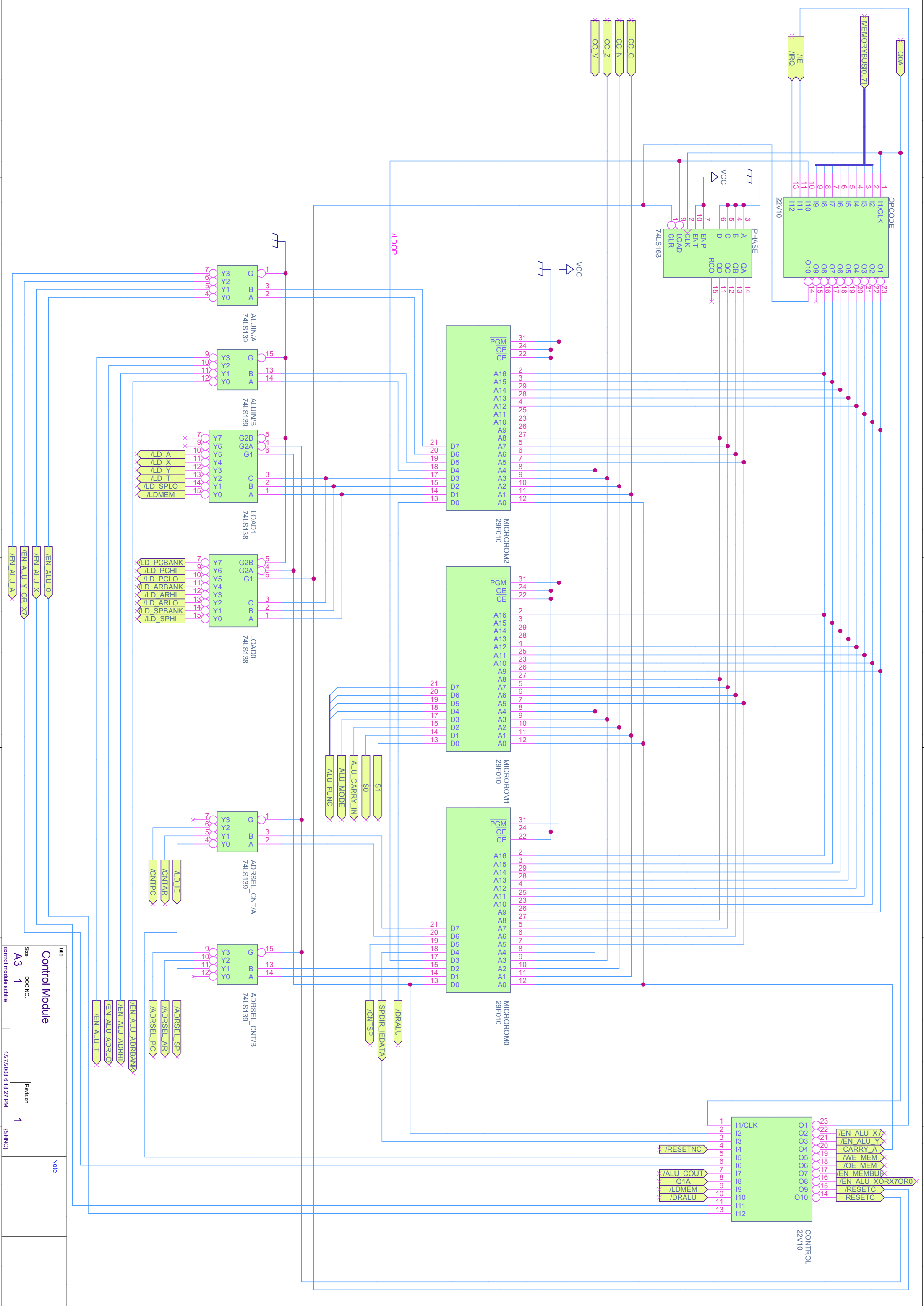


Q1 lags Q0 by 90 degrees

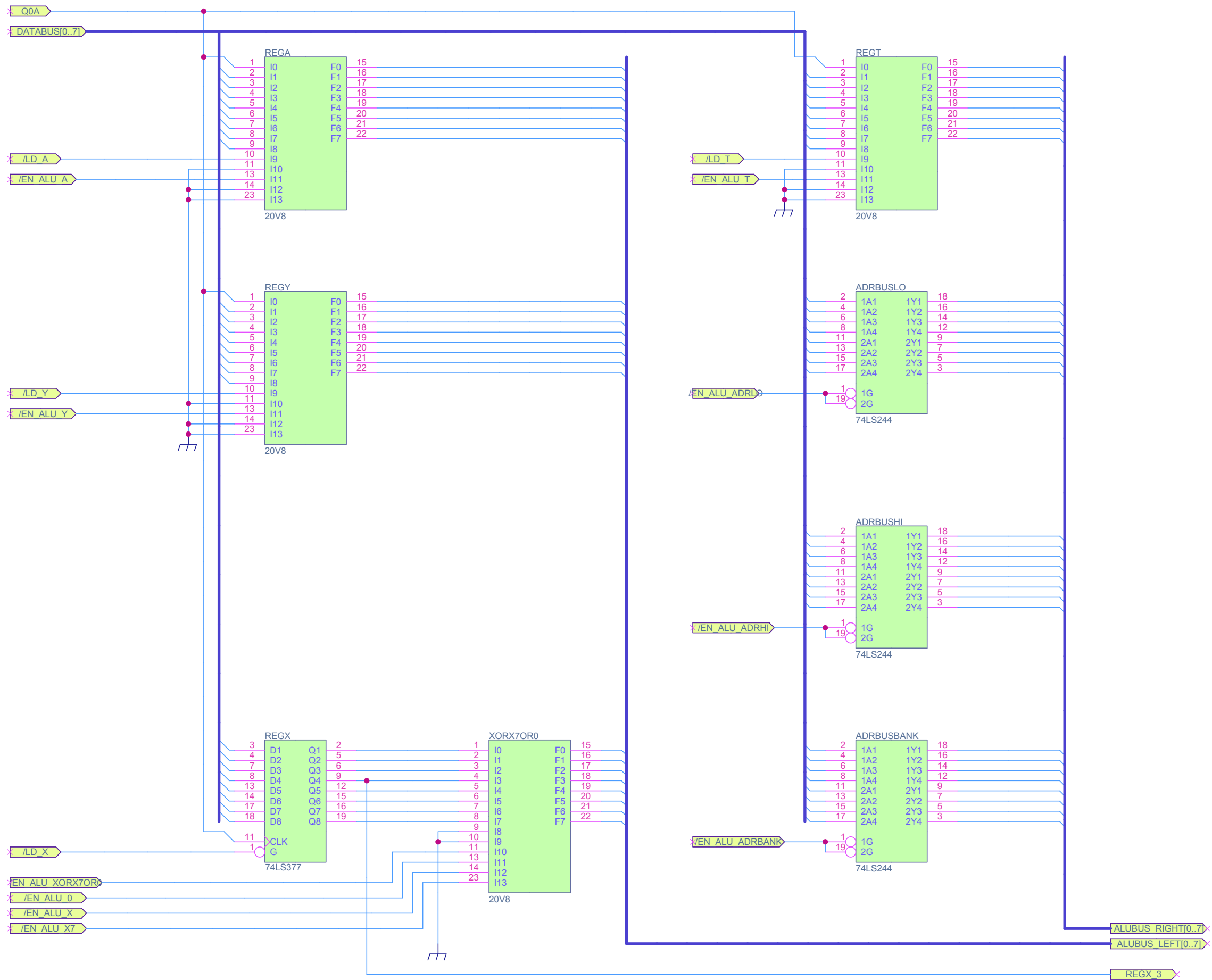
Q0

Q1

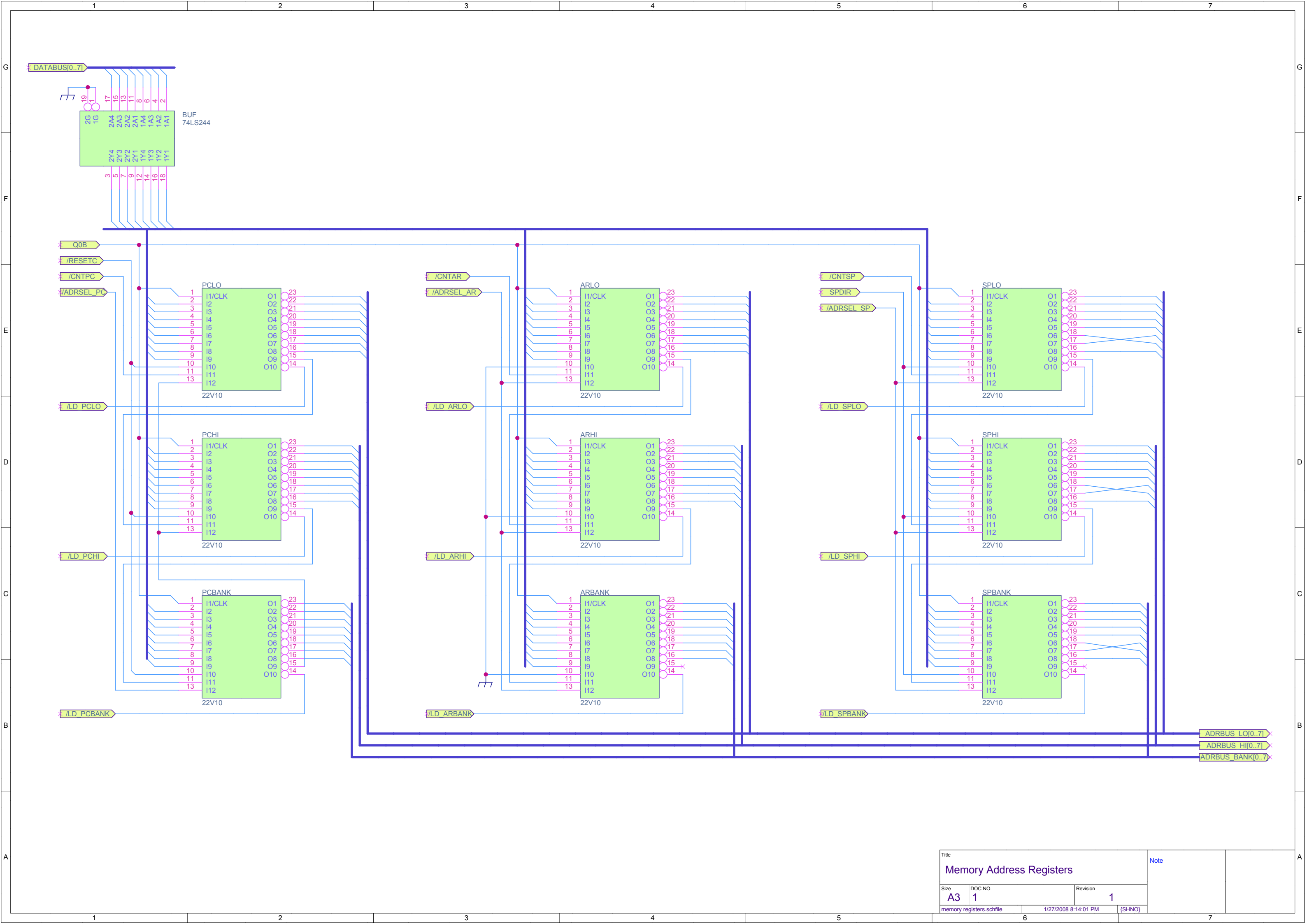
Title			Note
Clock/Reset System			
Size	DOC NO.	Revision	
A3	1	1	
clock generator.schfile		1/27/2008 8:21:50 PM	(SHNO)



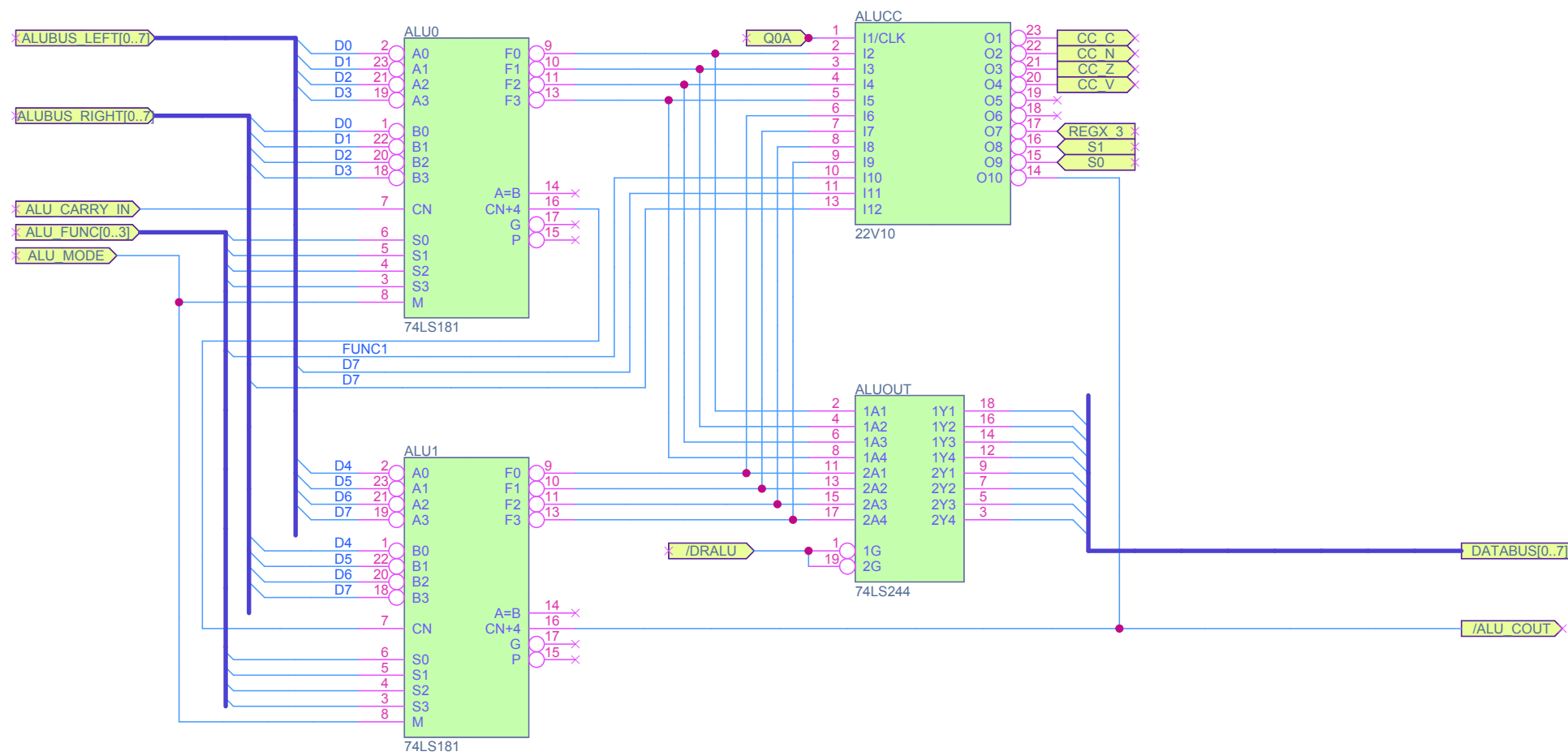
Title		Control Module	
Size	DOC NO.	Revision	
A3	1	1	
control module schlie		1/27/2008 6:18:27 PM (SHNO)	
Note			



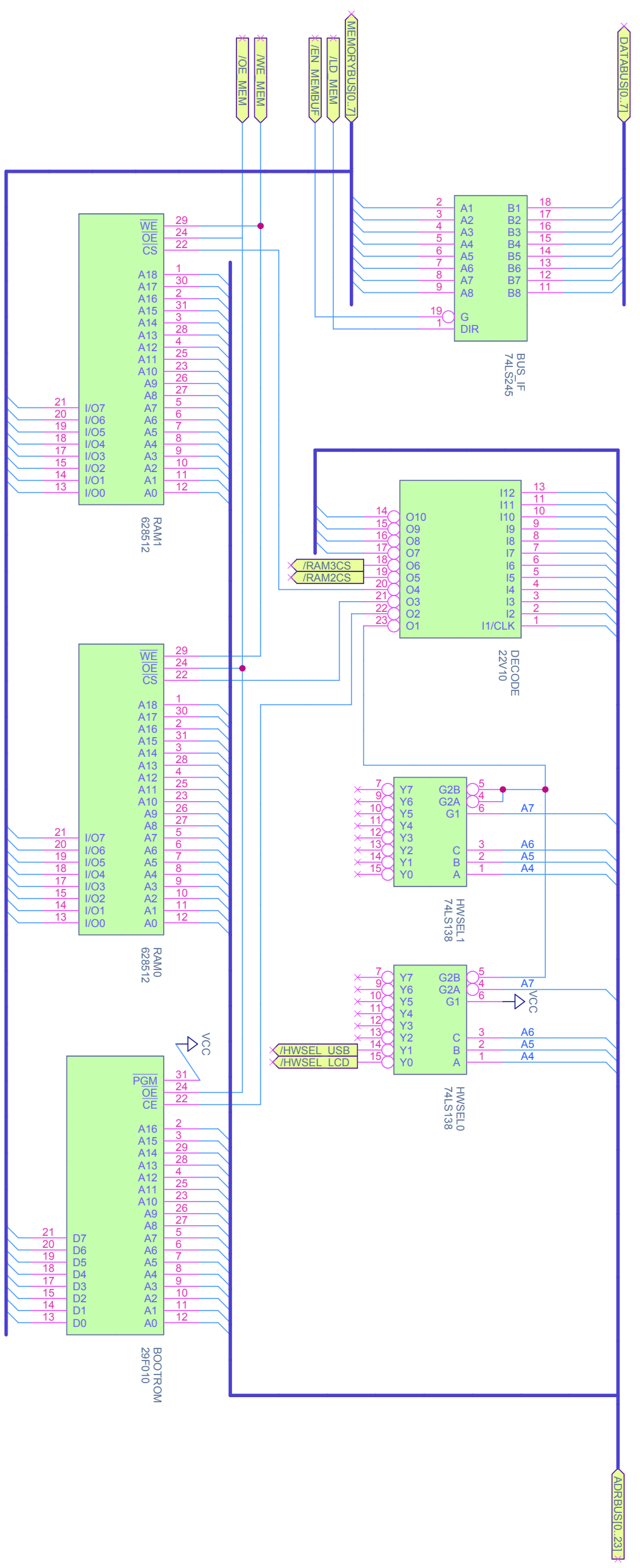
Title		Data Registers		Note	
Size	DOC NO.	Revision	1		
A3	1	1/27/2008 6:29:11 PM	(SHNO)		
data registers.schfile					



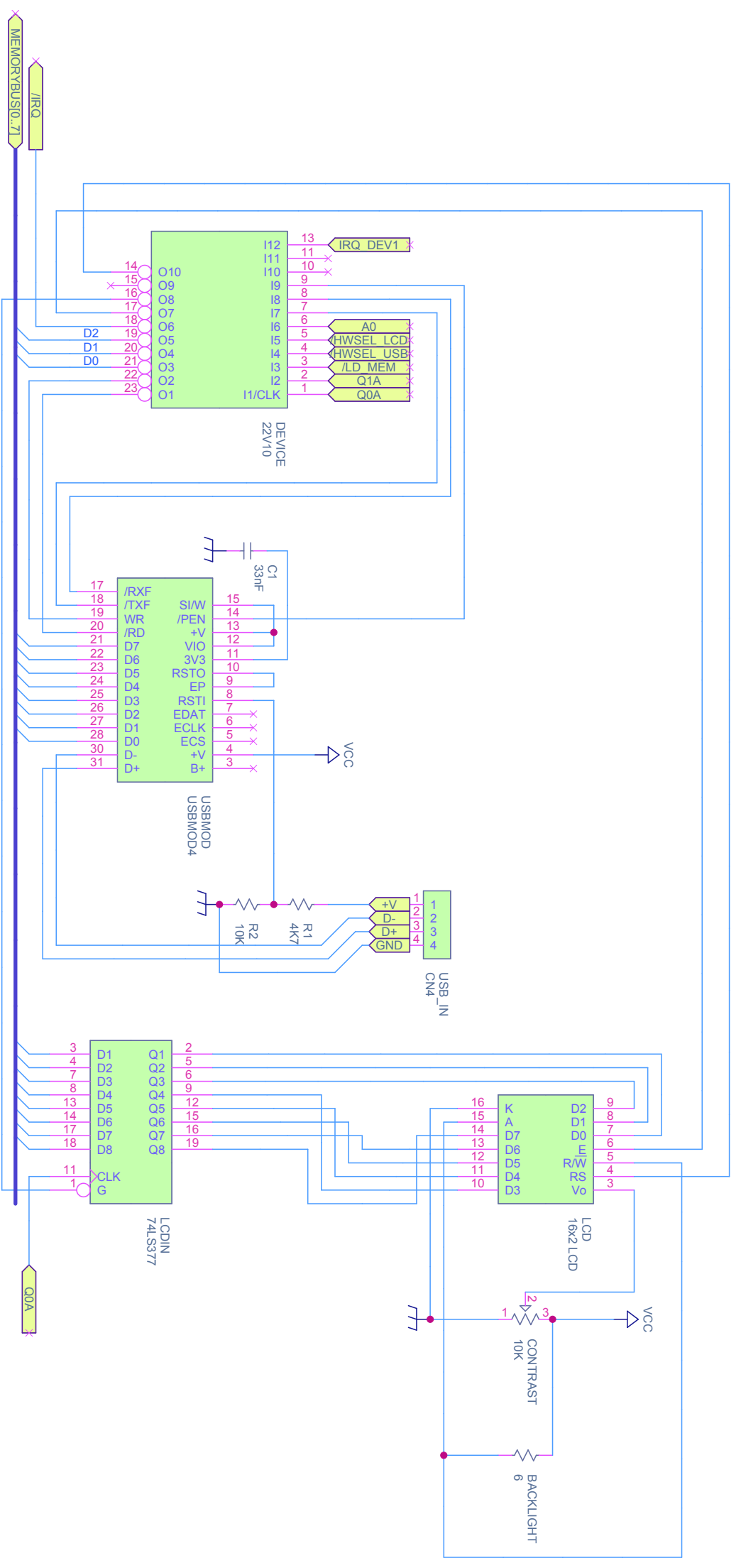
Title		Memory Address Registers		Note	
Size	DOC NO.	Revision	1		
memory registers.schfile		1/27/2008 8:14:01 PM		(SHNO)	



Title		ALU		Note	
Size	DOC NO.	Revision	1		
A3	1	1/27/2008 6:07:31 PM	(SHNO)		



Title		Revision		Date	
Memory System		1		1/28/2008 5:11:31 PM	
Doc No.	Rev	Author	Check	Date	
A3	1	memory_scfille			
Note					



Title		Hardware Devices	
Size	DOCC NO.	Revision	
A3	1	1	
Date		1/27/2008 6:08:53 PM	
Author		[SHNO]	
Note			

```
1 ; opcode register
2 ;
3 ; Stores current opcode. Clears opcode at the start of a new instruction, when an
4 ; interrupt is pending. Loads opcode when _reset is low.
5 ;
6 ; _ie is interrupt enable
7
8
9 CHIP opcode G22V10
10
11 clk0=1 d0=2 d1=3 d2=4 d3=5 d4=6 d5=7 d6=8 d7=9 _ldop=10 _ie=11 gnd=12
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
13
14 EQUATIONS
15
16 q0 := /_reset*d0 + _reset*/_ldop*_irq*d0 + _reset*/_ldop*_ie*d0 + _reset*_ldop*q0
17 q1 := /_reset*d1 + _reset*/_ldop*_irq*d1 + _reset*/_ldop*_ie*d1 + _reset*_ldop*q1
18 q2 := /_reset*d2 + _reset*/_ldop*_irq*d2 + _reset*/_ldop*_ie*d2 + _reset*_ldop*q2
19 q3 := /_reset*d3 + _reset*/_ldop*_irq*d3 + _reset*/_ldop*_ie*d3 + _reset*_ldop*q3
20 q4 := /_reset*d4 + _reset*/_ldop*_irq*d4 + _reset*/_ldop*_ie*d4 + _reset*_ldop*q4
21 q5 := /_reset*d5 + _reset*/_ldop*_irq*d5 + _reset*/_ldop*_ie*d5 + _reset*_ldop*q5
22 q6 := /_reset*d6 + _reset*/_ldop*_irq*d6 + _reset*/_ldop*_ie*d6 + _reset*_ldop*q6
23 q7 := /_reset*d7 + _reset*/_ldop*_irq*d7 + _reset*/_ldop*_ie*d7 + _reset*_ldop*q7
24
25
26
```

```
1 ; Control system logic
2 ;
3 ; Stores previous carry for address computations, and interrupt enable flag.
4 ; Determines which register the shared enable bit for Y and X7 should refer to.
5 ; Generates timed control signals for the memory system and hardware devices.
6
7 CHIP control G22V10
8
9 q0=1 load3=2 spdir_iedata=3 _resetnc=4 _ld_ie=5 _en_alu_yorx7=6 _alu_cout=7 q1=8 _ldmem=9
10 _dralu=10 _en_alu_x=11 gnd=12
11 _en_alu_0=13 _resetc=14 _resetc=15 _en_alu_xorx7or0=16 _en_membuf=17 _oe_mem=18 _we_mem=19
12 _carry_a=20 _en_alu_y=21 _en_alu_x7=22 _ie=23 vcc=24
13
14 EQUATIONS
15
16 ; _reset should only change on a clock boundary
17 _resetc := _resetnc
18 resetc := /_resetnc
19
20 _ie := /_resetc + _resetc*_ld_ie*spdir_iedata
21
22 _carry_a := _en_alu_yorx7*_carry_a + load3*_carry_a + /_en_alu_yorx7*/load3*_alu_cout
23
24 ; Y and X7 share a control bit: use X7 with address register destinations, otherwise use Y.
25 ; Note SPL0 is not included here. _carry_a will not be loaded when loading SPL0.
26 _en_alu_x7 = _en_alu_yorx7 + load3
27 _en_alu_y = _en_alu_yorx7 + /load3
28
29 ; X, X7, and 0 are all emitted by a single GAL with a shared enable flag.
30 _en_alu_xorx7or0 = _en_alu_x*_en_alu_yorx7*_en_alu_0 + _en_alu_x*load3*_en_alu_0
31
32 /_en_membuf = _dralu + /_ldmem
33
34 _oe_mem = /_ldmem
35
36 ; write-enable can only be active during the last quarter of the clock cycle
37 /_we_mem = /_ldmem*/q0*/q1
```

```
1 ; Generic 8-bit data register with load enable and output enable
2 ;
3 ; Used for the A, Y, and T registers
4 ;
5 ; Uses 22V8 registered mode. Pin 13 is hard-wired as an output enable in this mode.
6
7 CHIP reg G20V8
8
9 clk0=1 d0=2 d1=3 d2=4 d3=5 d4=6 d5=7 d6=8 d7=9 _ld=10 nc=11 gnd=12
10 _oe=13 nc=14 q0=15 q1=16 q2=17 q3=18 q4=19 q5=20 q6=21 q7=22 nc=23 vcc=24
11
12 EQUATIONS
13
14 q0 := /_ld*d0 + _ld*q0
15
16 q1 := /_ld*d1 + _ld*q1
17
18 q2 := /_ld*d2 + _ld*q2
19
20 q3 := /_ld*d3 + _ld*q3
21
22 q4 := /_ld*d4 + _ld*q4
23
24 q5 := /_ld*d5 + _ld*q5
25
26 q6 := /_ld*d6 + _ld*q6
27
28 q7 := /_ld*d7 + _ld*q7
29
30
31
```

```
1 ; A buffer that passes through a byte unmodified, or copies
2 ; input bit 7 to all the output bits, or outputs zero.
3 ;
4 ; Used for X register, X7 pseudo-register, and Zero pseudo-register.
5 ;
6 ; Uses 22V8 Complex mode.
7
8 CHIP xorx7or0 G20V8
9
10 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
11 _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
12
13 EQUATIONS
14
15 q0.oe = /_en_xorx7or0
16 q0 = /_en_x*d0 + /_en_x7*d7
17
18 q1.oe = /_en_xorx7or0
19 q1 = /_en_x*d1 + /_en_x7*d7
20
21 q2.oe = /_en_xorx7or0
22 q2 = /_en_x*d2 + /_en_x7*d7
23
24 q3.oe = /_en_xorx7or0
25 q3 = /_en_x*d3 + /_en_x7*d7
26
27 q4.oe = /_en_xorx7or0
28 q4 = /_en_x*d4 + /_en_x7*d7
29
30 q5.oe = /_en_xorx7or0
31 q5 = /_en_x*d5 + /_en_x7*d7
32
33 q6.oe = /_en_xorx7or0
34 q6 = /_en_x*d6 + /_en_x7*d7
35
36 q7.oe = /_en_xorx7or0
37 q7 = /_en_x*d7 + /_en_x7*d7
38
39
40
```

```

1 ; PC (program counter) register, LO and HI bytes (bits 0 to 15).
2 ;
3 ; Holds one byte of the program counter. Can be reset, loaded, or count up.
4
5 CHIP pclohi G22V10
6
7 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
8 _oe=13 _ld=14 _cnt_out=15 q7=16 q6=17 q5=18 q4=19 q3=20 q2=21 q1=22 q0=23 vcc=24
9
10 EQUATIONS
11
12 ; Output is also enabled when reset is active.
13 ; _adrsel_pc and _reset are combined externally to create _oe.
14 q0.oe = /_oe
15 q1.oe = /_oe
16 q2.oe = /_oe
17 q3.oe = /_oe
18 q4.oe = /_oe
19 q5.oe = /_oe
20 q6.oe = /_oe
21 q7.oe = /_oe
22
23 /_cnt_out = /_cnt_in*q0*q1*q2*q3*q4*q5*q6*q7
24
25 /q0 := /_reset + _reset*_cnt_in*_ld*/q0 + _reset*/_ld*/d0 +
26 _reset*_ld*/_cnt_in*q0
27
28 /q1 := /_reset + _reset*_cnt_in*_ld*/q1 + _reset*/_ld*/d1 +
29 _reset*_ld*/_cnt_in*q0*q1 +
30 _reset*_ld*/_cnt_in*/q0*/q1
31
32 /q2 := /_reset + _reset*_cnt_in*_ld*/q2 + _reset*/_ld*/d2 +
33 _reset*_ld*/_cnt_in*q0*q1*q2 +
34 _reset*_ld*/_cnt_in*/q2*/q1 +
35 _reset*_ld*/_cnt_in*/q2*/q0
36
37 /q3 := /_reset + _reset*_cnt_in*_ld*/q3 + _reset*/_ld*/d3 +
38 _reset*_ld*/_cnt_in*q0*q1*q2*q3 +
39 _reset*_ld*/_cnt_in*/q3*/q2 +
40 _reset*_ld*/_cnt_in*/q3*/q1 +
41 _reset*_ld*/_cnt_in*/q3*/q0
42
43 /q4 := /_reset + _reset*_cnt_in*_ld*/q4 + _reset*/_ld*/d4 +
44 _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4 +
45 _reset*_ld*/_cnt_in*/q4*/q3 +
46 _reset*_ld*/_cnt_in*/q4*/q2 +
47 _reset*_ld*/_cnt_in*/q4*/q1 +
48 _reset*_ld*/_cnt_in*/q4*/q0
49
50 /q5 := /_reset + _reset*_cnt_in*_ld*/q5 + _reset*/_ld*/d5 +
51 _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5 +
52 _reset*_ld*/_cnt_in*/q5*/q4 +
53 _reset*_ld*/_cnt_in*/q5*/q3 +
54 _reset*_ld*/_cnt_in*/q5*/q2 +
55 _reset*_ld*/_cnt_in*/q5*/q1 +
56 _reset*_ld*/_cnt_in*/q5*/q0
57
58 /q6 := /_reset + _reset*_cnt_in*_ld*/q6 + _reset*/_ld*/d6 +
59 _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5*q6 +
60 _reset*_ld*/_cnt_in*/q6*/q5 +
61 _reset*_ld*/_cnt_in*/q6*/q4 +
62 _reset*_ld*/_cnt_in*/q6*/q3 +
63 _reset*_ld*/_cnt_in*/q6*/q2 +
64 _reset*_ld*/_cnt_in*/q6*/q1 +
65 _reset*_ld*/_cnt_in*/q6*/q0
66
67 /q7 := /_reset + _reset*_cnt_in*_ld*/q7 + _reset*/_ld*/d7 +
68 _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5*q6*q7 +
69 _reset*_ld*/_cnt_in*/q7*/q6 +
70 _reset*_ld*/_cnt_in*/q7*/q5 +
71 _reset*_ld*/_cnt_in*/q7*/q4 +

```

```
72     _reset*_ld*/_cnt_in*/q7*/q3 +
73     _reset*_ld*/_cnt_in*/q7*/q2 +
74     _reset*_ld*/_cnt_in*/q7*/q1 +
75     _reset*_ld*/_cnt_in*/q7*/q0
76
```

```

1 ; PC (program counter) register, BANK byte (bits 16 to 23).
2 ;
3 ; Holds one byte of the program counter. Can be reset, loaded, or count up.
4
5 CHIP pcbank G22V10
6
7 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
8 _adrsel_pc=13 _ld=14 _oe_out=15 q7=16 q6=17 q5=18 q4=19 q3=20 q2=21 q1=22 q0=23 vcc=24
9
10 EQUATIONS
11
12 ; Output is also enabled when reset is active.
13 ; _adrsel_pc and _reset are combined here to create _oe.
14 _oe_out = _adrsel_pc*_reset
15
16 q0.oe = /_oe_out
17 q1.oe = /_oe_out
18 q2.oe = /_oe_out
19 q3.oe = /_oe_out
20 q4.oe = /_oe_out
21 q5.oe = /_oe_out
22 q6.oe = /_oe_out
23 q7.oe = /_oe_out
24
25 /q0 := /_reset + _reset*_cnt_in*_ld*/q0 + _reset*/_ld*/d0 +
26     _reset*_ld*/_cnt_in*q0
27
28 /q1 := /_reset + _reset*_cnt_in*_ld*/q1 + _reset*/_ld*/d1 +
29     _reset*_ld*/_cnt_in*q0*q1 +
30     _reset*_ld*/_cnt_in*/q0*/q1
31
32 /q2 := /_reset + _reset*_cnt_in*_ld*/q2 + _reset*/_ld*/d2 +
33     _reset*_ld*/_cnt_in*q0*q1*q2 +
34     _reset*_ld*/_cnt_in*/q2*/q1 +
35     _reset*_ld*/_cnt_in*/q2*/q0
36
37 /q3 := /_reset + _reset*_cnt_in*_ld*/q3 + _reset*/_ld*/d3 +
38     _reset*_ld*/_cnt_in*q0*q1*q2*q3 +
39     _reset*_ld*/_cnt_in*/q3*/q2 +
40     _reset*_ld*/_cnt_in*/q3*/q1 +
41     _reset*_ld*/_cnt_in*/q3*/q0
42
43 /q4 := /_reset + _reset*_cnt_in*_ld*/q4 + _reset*/_ld*/d4 +
44     _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4 +
45     _reset*_ld*/_cnt_in*/q4*/q3 +
46     _reset*_ld*/_cnt_in*/q4*/q2 +
47     _reset*_ld*/_cnt_in*/q4*/q1 +
48     _reset*_ld*/_cnt_in*/q4*/q0
49
50 /q5 := /_reset + _reset*_cnt_in*_ld*/q5 + _reset*/_ld*/d5 +
51     _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5 +
52     _reset*_ld*/_cnt_in*/q5*/q4 +
53     _reset*_ld*/_cnt_in*/q5*/q3 +
54     _reset*_ld*/_cnt_in*/q5*/q2 +
55     _reset*_ld*/_cnt_in*/q5*/q1 +
56     _reset*_ld*/_cnt_in*/q5*/q0
57
58 /q6 := /_reset + _reset*_cnt_in*_ld*/q6 + _reset*/_ld*/d6 +
59     _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5*q6 +
60     _reset*_ld*/_cnt_in*/q6*/q5 +
61     _reset*_ld*/_cnt_in*/q6*/q4 +
62     _reset*_ld*/_cnt_in*/q6*/q3 +
63     _reset*_ld*/_cnt_in*/q6*/q2 +
64     _reset*_ld*/_cnt_in*/q6*/q1 +
65     _reset*_ld*/_cnt_in*/q6*/q0
66
67 /q7 := /_reset + _reset*_cnt_in*_ld*/q7 + _reset*/_ld*/d7 +
68     _reset*_ld*/_cnt_in*q0*q1*q2*q3*q4*q5*q6*q7 +
69     _reset*_ld*/_cnt_in*/q7*/q6 +
70     _reset*_ld*/_cnt_in*/q7*/q5 +
71     _reset*_ld*/_cnt_in*/q7*/q4 +

```

```
72     _reset*_ld*/_cnt_in*/q7*/q3 +
73     _reset*_ld*/_cnt_in*/q7*/q2 +
74     _reset*_ld*/_cnt_in*/q7*/q1 +
75     _reset*_ld*/_cnt_in*/q7*/q0
76
```

```

1 ; AR (address) register
2 ;
3 ; Holds one byte of the address register. Can be loaded or count up.
4
5 CHIP ar G22V10
6
7 ; Note q outputs are not in order, because:
8 ; q6 has 15 product terms, and must be on pin 18 or 19
9 ; q5 has 13 product terms, and must be on pin 17-20
10 ; q4 and q7 have 11 product terms, and must be on pin 16-21
11 ; q3 has 9 product terms, and must be on pin 15-22
12
13 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
14 _oe=13 _ld=14 _cnt_out=15 q7=16 q6=17 q5=18 q4=19 q3=20 q2=21 q1=22 q0=23 vcc=24
15
16 EQUATIONS
17
18 q0.oe = /_oe
19 q1.oe = /_oe
20 q2.oe = /_oe
21 q3.oe = /_oe
22 q4.oe = /_oe
23 q5.oe = /_oe
24 q6.oe = /_oe
25 q7.oe = /_oe
26
27 /_cnt_out = /_cnt_in*q0*q1*q2*q3*q4*q5*q6*q7
28
29 /q0 := _cnt_in*_ld*/q0 + /_ld*/d0 +
30 _ld*/_cnt_in*q0
31
32 /q1 := _cnt_in*_ld*/q1 + /_ld*/d1 +
33 _ld*/_cnt_in*q0*q1 +
34 _ld*/_cnt_in*/q0*/q1
35
36 /q2 := _cnt_in*_ld*/q2 + /_ld*/d2 +
37 _ld*/_cnt_in*q0*q1*q2 +
38 _ld*/_cnt_in*/q2*/q1 +
39 _ld*/_cnt_in*/q2*/q0
40
41 /q3 := _cnt_in*_ld*/q3 + /_ld*/d3 +
42 _ld*/_cnt_in*q0*q1*q2*q3 +
43 _ld*/_cnt_in*/q3*/q2 +
44 _ld*/_cnt_in*/q3*/q1 +
45 _ld*/_cnt_in*/q3*/q0
46
47 /q4 := _cnt_in*_ld*/q4 + /_ld*/d4 +
48 _ld*/_cnt_in*q0*q1*q2*q3*q4 +
49 _ld*/_cnt_in*/q4*/q3 +
50 _ld*/_cnt_in*/q4*/q2 +
51 _ld*/_cnt_in*/q4*/q1 +
52 _ld*/_cnt_in*/q4*/q0
53
54 /q5 := _cnt_in*_ld*/q5 + /_ld*/d5 +
55 _ld*/_cnt_in*q0*q1*q2*q3*q4*q5 +
56 _ld*/_cnt_in*/q5*/q4 +
57 _ld*/_cnt_in*/q5*/q3 +
58 _ld*/_cnt_in*/q5*/q2 +
59 _ld*/_cnt_in*/q5*/q1 +
60 _ld*/_cnt_in*/q5*/q0
61
62 /q6 := _cnt_in*_ld*/q6 + /_ld*/d6 +
63 _ld*/_cnt_in*q0*q1*q2*q3*q4*q5*q6 +
64 _ld*/_cnt_in*/q6*/q5 +
65 _ld*/_cnt_in*/q6*/q4 +
66 _ld*/_cnt_in*/q6*/q3 +
67 _ld*/_cnt_in*/q6*/q2 +
68 _ld*/_cnt_in*/q6*/q1 +
69 _ld*/_cnt_in*/q6*/q0
70
71 /q7 := _cnt_in*_ld*/q7 + /_ld*/d7 +

```

```
72      _ld*/_cnt_in*q0*q1*q2*q3*q4*q5*q6*q7 +
73      _ld*/_cnt_in*/q7*/q6 +
74      _ld*/_cnt_in*/q7*/q5 +
75      _ld*/_cnt_in*/q7*/q4 +
76      _ld*/_cnt_in*/q7*/q3 +
77      _ld*/_cnt_in*/q7*/q2 +
78      _ld*/_cnt_in*/q7*/q1 +
79      _ld*/_cnt_in*/q7*/q0
80
```

```

1 ; SP (stack pointer) register
2 ;
3 ; Holds one byte of the stack pointer. Can be loaded or count up/down.
4
5 CHIP sp G22V10
6
7 ; Note q outputs are not in order, because:
8 ; q6 has 15 product terms, and must be on pin 18 or 19
9 ; q5 has 13 product terms, and must be on pin 17-20
10 ; q4 and q7 have 11 product terms, and must be on pin 16-21
11 ; q3 has 9 product terms, and must be on pin 15-22
12
13 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
14 _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
15
16 EQUATIONS
17
18 q0.oe = /_oe
19 q1.oe = /_oe
20 q2.oe = /_oe
21 q3.oe = /_oe
22 q4.oe = /_oe
23 q5.oe = /_oe
24 q6.oe = /_oe
25 q7.oe = /_oe
26
27 /_cnt_out = _ld*/_cnt_in*dir*q0*q1*q2*q3*q4*q5*q6*q7 +
28             _ld*/_cnt_in*/dir*/q0*/q1*/q2*/q3*/q4*/q5*/q6*/q7
29
30 /q0 := _cnt_in*_ld*/q0 + /_ld*/d0 +
31         _ld*/_cnt_in*q0
32
33 /q1 := _cnt_in*_ld*/q1 + /_ld*/d1 +
34         _ld*/_cnt_in*dir*q0*q1 +
35         _ld*/_cnt_in*dir*/q0*/q1 +
36         _ld*/_cnt_in*/dir*q0*/q1 +
37         _ld*/_cnt_in*/dir*/q0*q1
38
39 /q2 := _cnt_in*_ld*/q2 + /_ld*/d2 +
40         _ld*/_cnt_in*dir*q0*q1*q2 +
41         _ld*/_cnt_in*dir*/q2*/q1 +
42         _ld*/_cnt_in*dir*/q2*/q0 +
43         _ld*/_cnt_in*/dir*q2*/q1*/q0 +
44         _ld*/_cnt_in*/dir*/q2*q1 +
45         _ld*/_cnt_in*/dir*/q2*q0
46
47 /q3 := _cnt_in*_ld*/q3 + /_ld*/d3 +
48         _ld*/_cnt_in*dir*q0*q1*q2*q3 +
49         _ld*/_cnt_in*dir*/q3*/q2 +
50         _ld*/_cnt_in*dir*/q3*/q1 +
51         _ld*/_cnt_in*dir*/q3*/q0 +
52         _ld*/_cnt_in*/dir*q3*/q2*/q1*/q0 +
53         _ld*/_cnt_in*/dir*/q3*q2 +
54         _ld*/_cnt_in*/dir*/q3*q1 +
55         _ld*/_cnt_in*/dir*/q3*q0
56
57 /q4 := _cnt_in*_ld*/q4 + /_ld*/d4 +
58         _ld*/_cnt_in*dir*q0*q1*q2*q3*q4 +
59         _ld*/_cnt_in*dir*/q4*/q3 +
60         _ld*/_cnt_in*dir*/q4*/q2 +
61         _ld*/_cnt_in*dir*/q4*/q1 +
62         _ld*/_cnt_in*dir*/q4*/q0 +
63         _ld*/_cnt_in*/dir*q4*/q3*/q2*/q1*/q0 +
64         _ld*/_cnt_in*/dir*/q4*q3 +
65         _ld*/_cnt_in*/dir*/q4*q2 +
66         _ld*/_cnt_in*/dir*/q4*q1 +
67         _ld*/_cnt_in*/dir*/q4*q0
68
69 /q5 := _cnt_in*_ld*/q5 + /_ld*/d5 +
70         _ld*/_cnt_in*dir*q0*q1*q2*q3*q4*q5 +
71         _ld*/_cnt_in*dir*/q5*/q4 +

```

```

72      _ld*/_cnt_in*dir*/q5*/q3 +
73      _ld*/_cnt_in*dir*/q5*/q2 +
74      _ld*/_cnt_in*dir*/q5*/q1 +
75      _ld*/_cnt_in*dir*/q5*/q0 +
76      _ld*/_cnt_in*/dir*/q5*/q4*/q3*/q2*/q1*/q0 +
77      _ld*/_cnt_in*/dir*/q5*/q4 +
78      _ld*/_cnt_in*/dir*/q5*/q3 +
79      _ld*/_cnt_in*/dir*/q5*/q2 +
80      _ld*/_cnt_in*/dir*/q5*/q1 +
81      _ld*/_cnt_in*/dir*/q5*/q0
82
83 /q6 := _cnt_in*_ld*/q6 + /_ld*/d6 +
84      _ld*/_cnt_in*dir*/q0*q1*q2*q3*q4*q5*q6 +
85      _ld*/_cnt_in*dir*/q6*/q5 +
86      _ld*/_cnt_in*dir*/q6*/q4 +
87      _ld*/_cnt_in*dir*/q6*/q3 +
88      _ld*/_cnt_in*dir*/q6*/q2 +
89      _ld*/_cnt_in*dir*/q6*/q1 +
90      _ld*/_cnt_in*dir*/q6*/q0 +
91      _ld*/_cnt_in*/dir*/q6*/q5*/q4*/q3*/q2*/q1*/q0 +
92      _ld*/_cnt_in*/dir*/q6*/q5 +
93      _ld*/_cnt_in*/dir*/q6*/q4 +
94      _ld*/_cnt_in*/dir*/q6*/q3 +
95      _ld*/_cnt_in*/dir*/q6*/q2 +
96      _ld*/_cnt_in*/dir*/q6*/q1 +
97      _ld*/_cnt_in*/dir*/q6*/q0
98
99 q7 := _cnt_in*_ld*q7 + /_ld*d7 +
100      _ld*/_cnt_in*dir*/q7*q6*q5*q4*q3*q2*q1*q0 +
101      _ld*/_cnt_in*dir*/q7*/_cnt_out +
102      _ld*/_cnt_in*/dir*/q7*/q6*/q5*/q4*/q3*/q2*/q1*/q0 +
103      _ld*/_cnt_in*/dir*/q7*q6 +
104      _ld*/_cnt_in*/dir*/q7*q5 +
105      _ld*/_cnt_in*/dir*/q7*q4 +
106      _ld*/_cnt_in*/dir*/q7*q3 +
107      _ld*/_cnt_in*/dir*/q7*q2 +
108      _ld*/_cnt_in*/dir*/q7*q1 +
109      _ld*/_cnt_in*/dir*/q7*q0
110

```

```

1 ; ALU condition codes
2 ;
3 ; Computes, stores, and shifts the condition codes from the ALU.
4 ; Carry (C) active low
5 ; Negative (N) active high
6 ; Zero/Equal (z) active low
7 ; Overflow (V) active low
8 ;
9 ; f[7..0] is the ALU result
10 ; func1 is bit 1 of the ALU function that performed. 1 means it was a subtraction.
11 ; a7 and b7 are the MSB's of the left and right ALU operands.
12 ; regx3 is bit 3 from the X register
13 ;
14 ; s1 s0 operation
15 ; 0 0 no change
16 ; 0 1 shift left
17 ; 1 0 shift right
18 ; 1 1 load
19 ;
20 ; Shift left isn't really needed. It might be useful to replace that operation with
21 ; a "set" operation that directly sets VZNC from the lowest 4 bits of the ALU result?
22 ; This might enable a reduction in number of microcode instructions for some opcodes.
23
24 CHIP alucc G22V10
25
26 q0=1 f0=2 f1=3 f2=4 f3=5 f4=6 f5=7 f6=8 f7=9 func1=10 a7=11 gnd=12
27 b7=13 _cin=14 s0=15 s1=16 regx3=17 nc=18 nc=19 _cc_v=20 _cc_z=21 cc_n=22 _cc_c=23 vcc=24
28
29 EQUATIONS
30
31 /_cc_c := /s1*/s0*/_cc_c +
32         /s1*s0 +
33         s1*/s0*/cc_n +
34         s1*s0*/_cin
35
36 cc_n := /s1*/s0*cc_n +
37         /s1*s0*_cc_c +
38         s1*/s0*_cc_z +
39         s1*s0*f7
40
41 /_cc_z := /s1*/s0*/_cc_z +
42         /s1*s0*/cc_n +
43         s1*/s0*/_cc_v +
44         s1*s0*/f0*/f1*/f2*/f3*/f4*/f5*/f6*/f7
45
46 /_cc_v := /s1*/s0*/_cc_v +
47         /s1*s0*/_cc_z +
48         s1*/s0*/regx3 +
49         s1*s0*/a7*/b7*f7*/func1 +
50         s1*s0*a7*b7*/f7*/func1 +
51         s1*s0*/a7*b7*f7*func1 +
52         s1*s0*a7*/b7*/f7*func1

```

```

1 ; Address decoder
2 ;
3 ; Decodes the 24-bit address, and generates enable signals for ROM, RAM, and hardware
4 ; devices. The particular decode behavior implemented here determines the machine's memory
5 ; map.
6 ;
7 ; total address space is 16MB
8 ; $000000 - $003EFF is ROM (16K - 256 = 15.75K)
9 ; $003F00 - $003FFF is hardware devices (16 device maps of 16 bytes each)
10 ; $004000 - $07FFFF is RAM bank 0 (512K - 16K = 496K)
11 ; $080000 - $0FFFFFF is RAM bank 1 (512K)
12 ; $100000 - $17FFFF is RAM bank 2 (512K, not installed)
13 ; $180000 - $1FFFFFF is RAM bank 3 (512K, not installed)
14 ; theoretically there could be RAM banks 4 to 31 occupying the remaining address space.
15
16 CHIP decode G22V10
17
18 a8=1 a9=2 a10=3 a11=4 a12=5 a13=6 a14=7 a15=8 a16=9 a17=10 a18=11 gnd=12
19 a19=13 a20=14 a21=15 a22=16 a23=17 _ram3_cs=18 _ram2_cs=19 _ram1_cs=20 _ram0_cs=21
20     _rom_cs=22 _hw_cs=23 vcc=24
21
22 EQUATIONS
23
24 ; RAM3 means [A23..A19] = 00011
25 /_ram3_cs = /a23*/a22*/a21*a20*a19
26
27 ; RAM2 means [A23..A19] = 00010
28 /_ram2_cs = /a23*/a22*/a21*a20*/a19
29
30 ; RAM1 means [A23..A19] = 00001
31 /_ram1_cs = /a23*/a22*/a21*/a20*a19
32
33 ; RAM0 means [A23..A19] = 00000 and [A18..A14] != 00000
34 /_ram0_cs = /a23*/a22*/a21*/a20*/a19*a18 +
35             /a23*/a22*/a21*/a20*/a19*a17 +
36             /a23*/a22*/a21*/a20*/a19*a16 +
37             /a23*/a22*/a21*/a20*/a19*a15 +
38             /a23*/a22*/a21*/a20*/a19*a14
39
40 ; ROM means [A23..A14] = 0000000000 and [A13..A8] != 111111
41 /_rom_cs = /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a13 +
42           /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a12 +
43           /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a11 +
44           /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a10 +
45           /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a9 +
46           /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*/a8
47
48 ; hardware device means [A23..A8] = 0000000000111111
49 /_hw_cs = /a23*/a22*/a21*/a20*/a19*/a18*/a17*/a16*/a15*/a14*a13*a12*a11*a10*a9*a8
50
51

```

```
1 ; USB module interface, LCD interface, and interrupt request generator
2 ;
3 ; Generates control signals for USBMOD4 I/O.
4 ; Generates enable signal for LCD.
5 ; Provides a status register for USBMOD4 ready/transmit/receive status flags.
6 ; ORs individual IRQ lines from all interrupt-generating devices to create IRQ for CPU.
7
8 CHIP device G22V10
9
10 q0=1 q1=2 _ld_mem=3 _hwsel_usb=4 _hwsel_lcd=5 a0=6 _usb_txf=7 _usb_rxf=8 _usb_pen=9 nc=10
11   nc=11 gnd=12
12 nc=13 lcd_rs=14 lcd_run=15 _lcdin_g=16 lcd_e=17 _irq=18 d2=19 d1=20 d0=21 usb_wr=22
13   _usb_rd=23 vcc=24
14
15 EQUATIONS
16
17 ; LCD is configured write-only. Data is first stored in a dedicated register, then written
18 ; to LCD on the next clock cycle. As wired, the LCD busy flag can't be read. Therefore the
19 ; CPU must wait long enough before each LCD write to make sure the previous write operation
20 ; has completed.
21 ;
22 ; LCD Operation Times
23 ; clear, home: 1.52ms
24 ; other writes: 41us
25
26 /_lcdin_g = /_hwsel_lcd*/_ld_mem
27 lcd_run := /_lcdin_g
28 lcd_rs := /_lcdin_g*/a0 + _lcdin_g*lcd_rs
29 lcd_e = lcd_run*/q0 + lcd_run*q1
30
31 ; USB read and write are both only active during the last quarter of the clock cycle.
32 usb_wr = /_hwsel_usb*/a0*/_ld_mem*/q0*/q1
33 /_usb_rd = /_hwsel_usb*/a0*_ld_mem*/q0*/q1
34
35 ; _irq should only change on a clock boundary
36 _irq := _usb_rxf
37
38 ; 3 or 1 bit status register for USB or LCD
39 d2.oe = /_hwsel_usb*a0*_ld_mem
40 d2 = _usb_pen
41 d1.oe = /_hwsel_usb*a0*_ld_mem
42 d1 = _usb_txf
43 d0.oe = /_hwsel_usb*a0*_ld_mem
44 d0 = _usb_rxf
45
```