Memory Systems 2

96 SHEETS • 5 x 5 QUAD
10⅛ x 7⅞ • 53-110

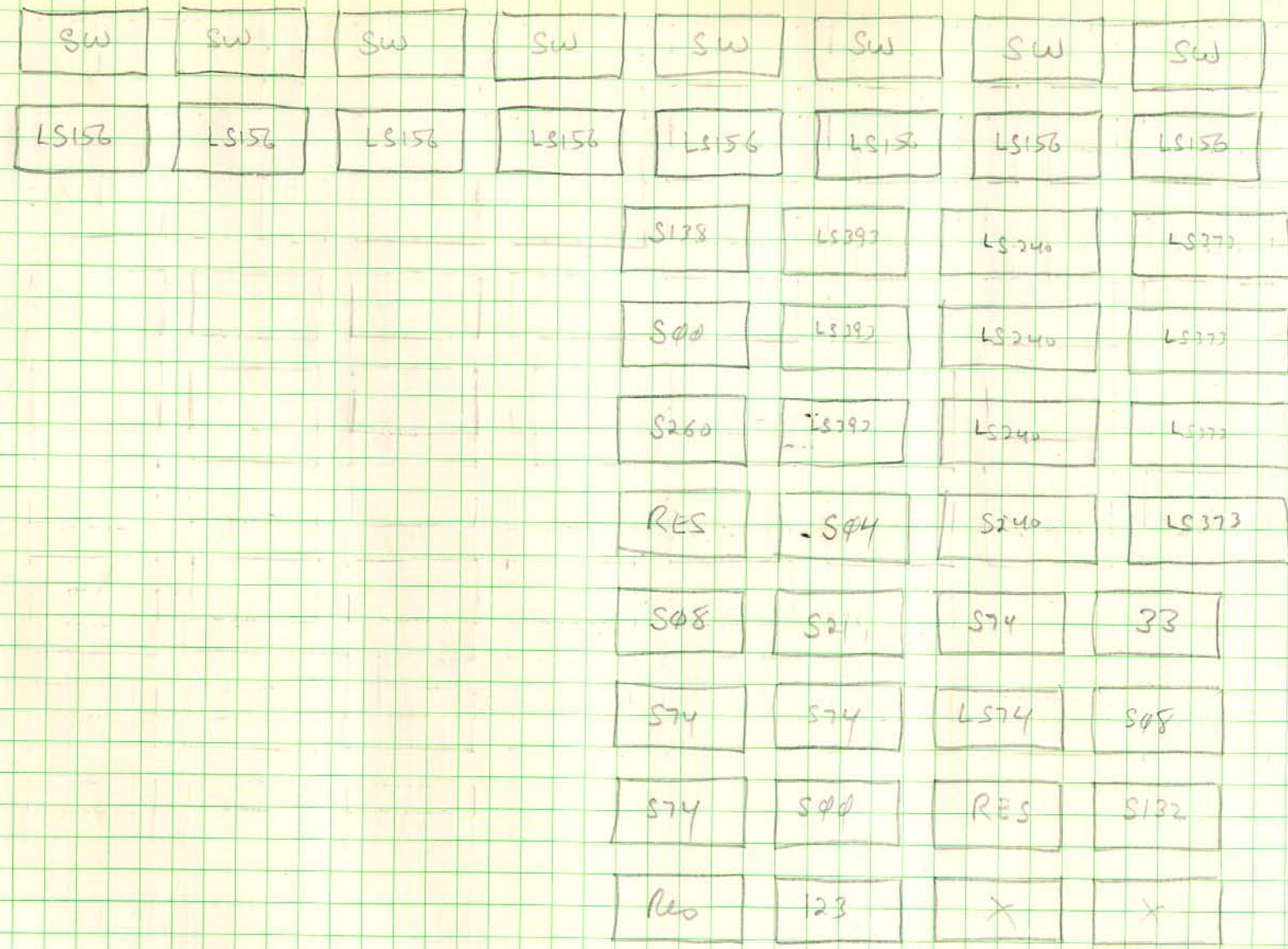
NATIONAL

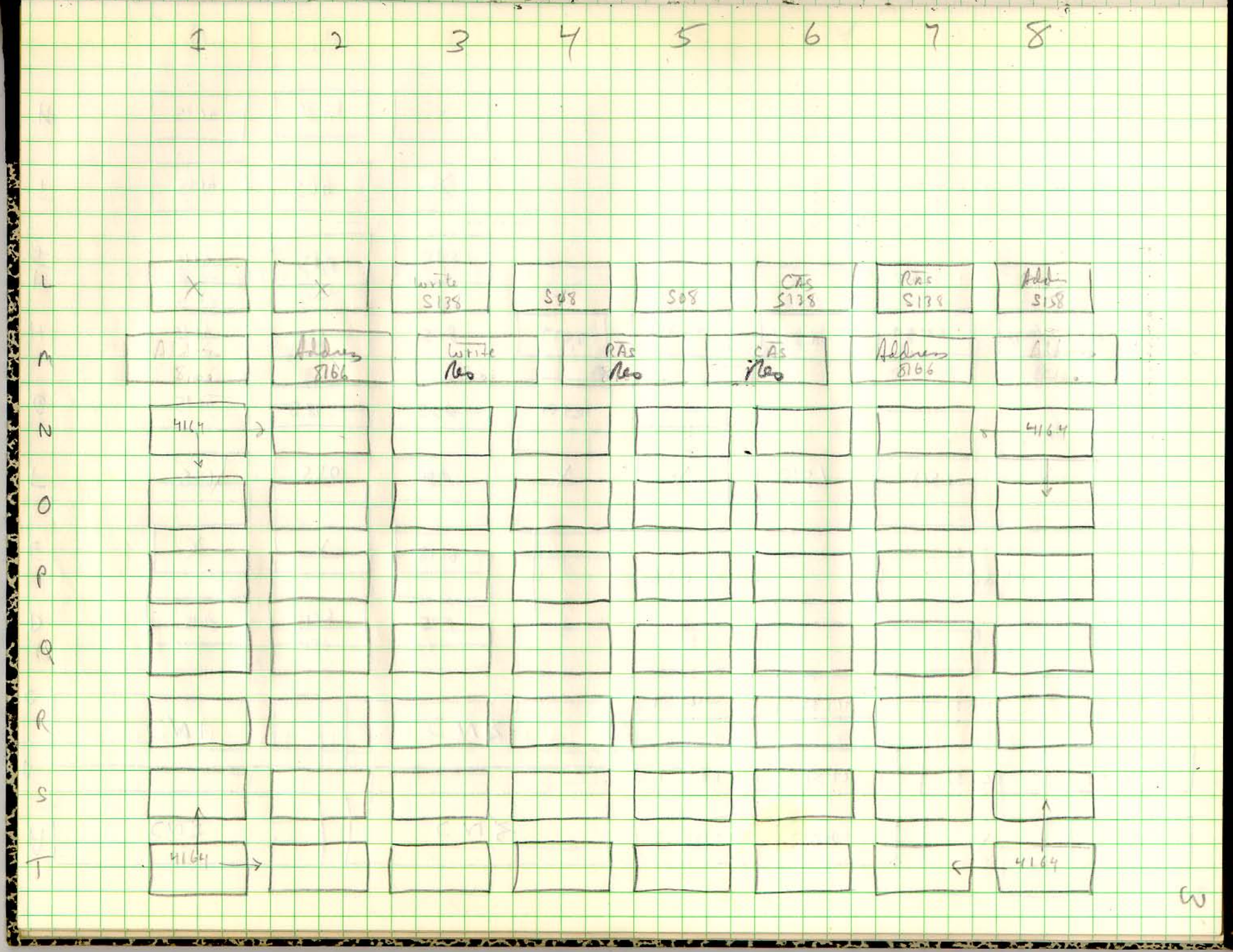
NATIONAL BLANK BOOK COMPANY, INC.
Holyoke, Massachusetts 01040 • Made in USA

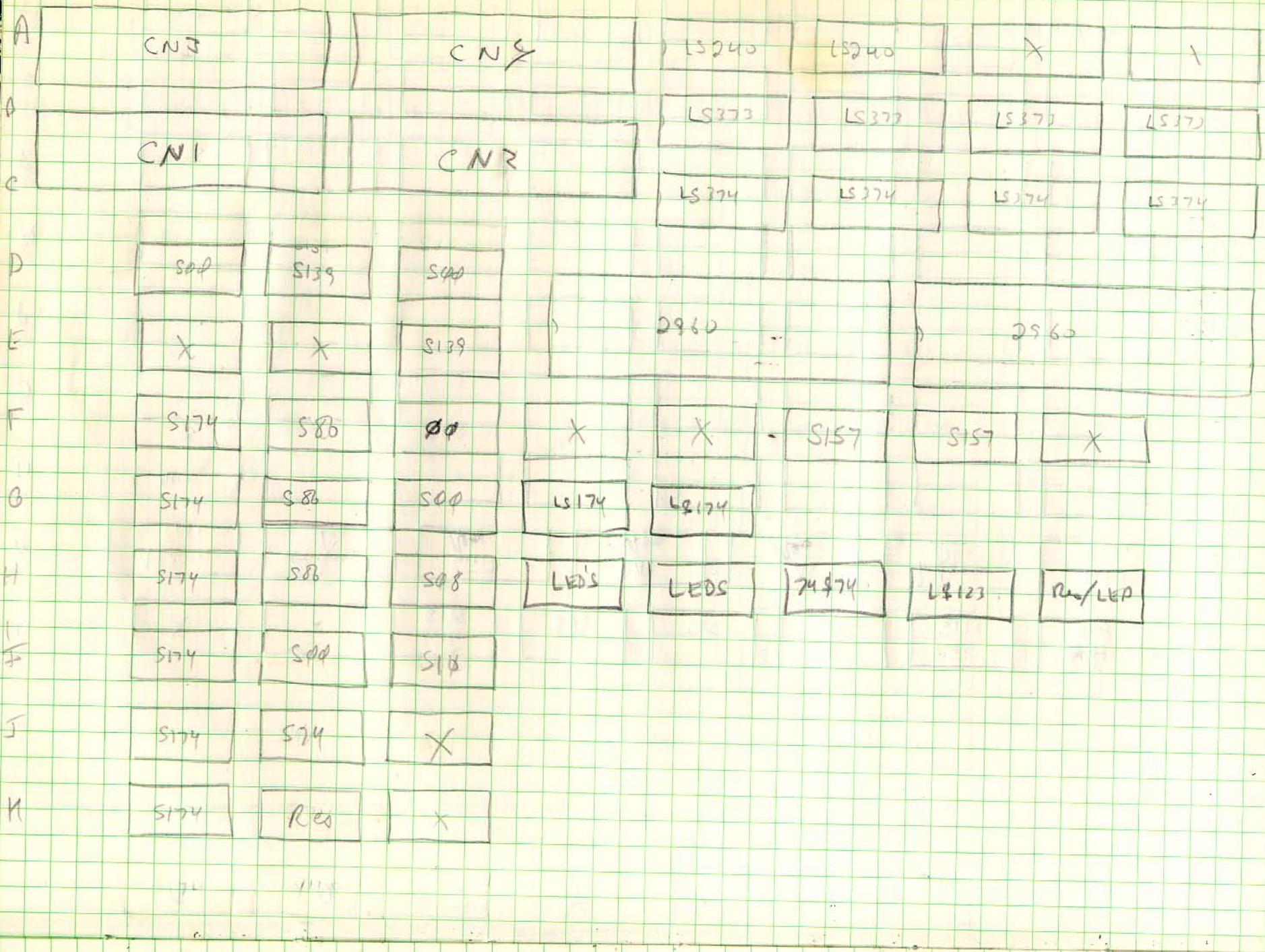
4 megabyte

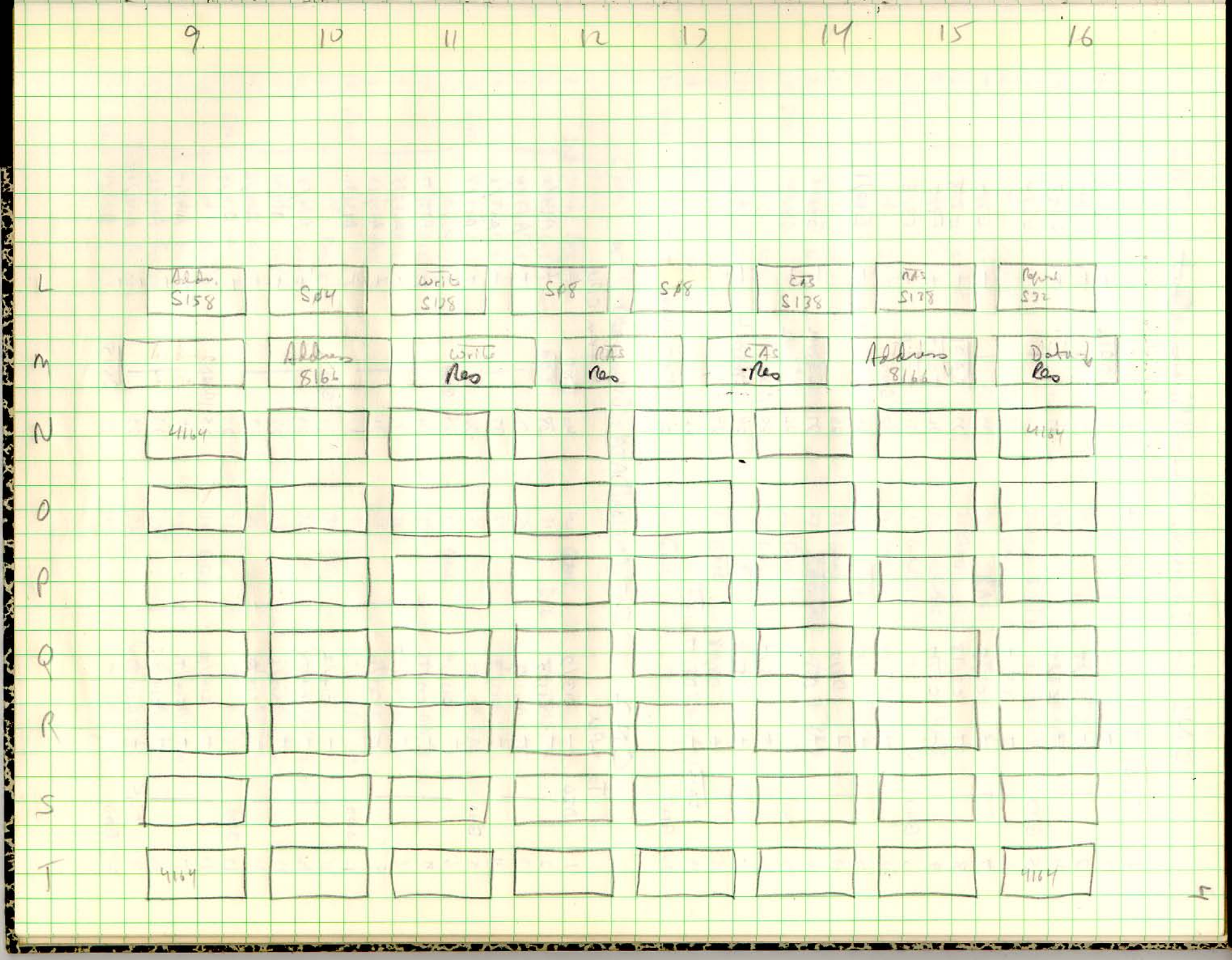
Error Corrected Memory System

Controller/ECC Board




1
2
3
4
5
6
7
8
L
M
N
O
P
Q
R
S
T
X
X
Write
S138
S08
S08
CAS
S138
RAS
S138
Address
S158
Address
8166
Write
RAS
CAS
Address
8166
4164
4164
4164
4164


A
B
C
D
E
F
G
H
J
K
CN3
CN4
CN1
CN2
LS240
LS240
X
1
LS373
LS373
LS373
LS373
LS374
LS374
LS374
LS374
S00
S139
S00
X
X
S139
2960
2960
S174
S86
00
X
X
S157
S157
X
S174
S86
S00
LS174
LS174
S174
S86
S08
LED'S
LEDS
74S74
LS123
RW/LED
S174
S00
S10
S174
S74
X
S174
Res
X
9
10
11
12
13
14
15
16


9
10
11
12
13
14
15
16
L
M
N
O
P
Q
R
S
T
Addr. S158
S#4
Write S138
S#8
S#8
CAS S138
RAS S178
Refresh S32
Address 8164
Write Res
RAS Res
CAS Res
Address 8164
Data Res
41164
41164
41164
41164

CN1 Internal Bus Address Out

1	$\overline{IA0}$	–	
2		–	
3		+	C8/3
4		+	C8/4
5	GND		
6		+	C8/7
7		+	C8/8
8		+	C8/18
9		+	C8/17
10	GND		
11		+	C8/14
12		+	C8/13
13		+	D8/3
14		+	D8/4
15	GND		
16		+	D8/7
17		+	D8/8
18		+	D8/18
19	$\overline{IA15}$	+	D8/17
20	GND		
40	GND		
39	$\overline{IA16}$	+	D8/14
38		+	D8/13
37		–	E8/3
36		–	E8/4
35	GND		
34		+	E8/7
33	$\overline{IA21}$	+	E8/8
32	$\overline{IC\emptyset3}$	+	F8/7
31	$\overline{IC1D}$	+	F8/8
30	GND		
29	IXMSYNL	+	E10/1
28			
27	ISYNL	–	G8/12
26	UBSYNL	–	G8/4
25	GND		
24	IMSYNL	–	NC
23			
22			
21	INIT CLEAR	+	– NC

CN2 INTERNAL DATA BUS OUT/IN

To CPU

1	DI0	+	C13/2
2		+	C13/5
3		+	C13/6
4		–	C13/9
5	GND		
6		+	C13/19
7		+	C13/16
8		+	C13/15
9		+	C13/12
10	GND		
11		+	C14/2
12		+	C14/5
13		+	C14/6
14		+	C14/9
15	GND		
16		+	C14/19
17		+	C14/16
18		+	C14/15
19	DI15	+	C14/12
20	GND		

from CPU

40	GND		
39	$\overline{DO\emptyset}$	+	A13/2
38		+	A13/4
37		+	A13/6
36		+	A13/8
35	GND		
34		+	A13/11
33		–	A13/15
32		+	A13/13
31		+	A13/11
30	GND		
29		–	A14/2
28		+	A14/4
27		+	A14/6
26		+	A14/8
25	GND		
24		+	A14/17
23		+	A14/15
22		+	A14/13
21	$\overline{DO15}$	+	A14/11

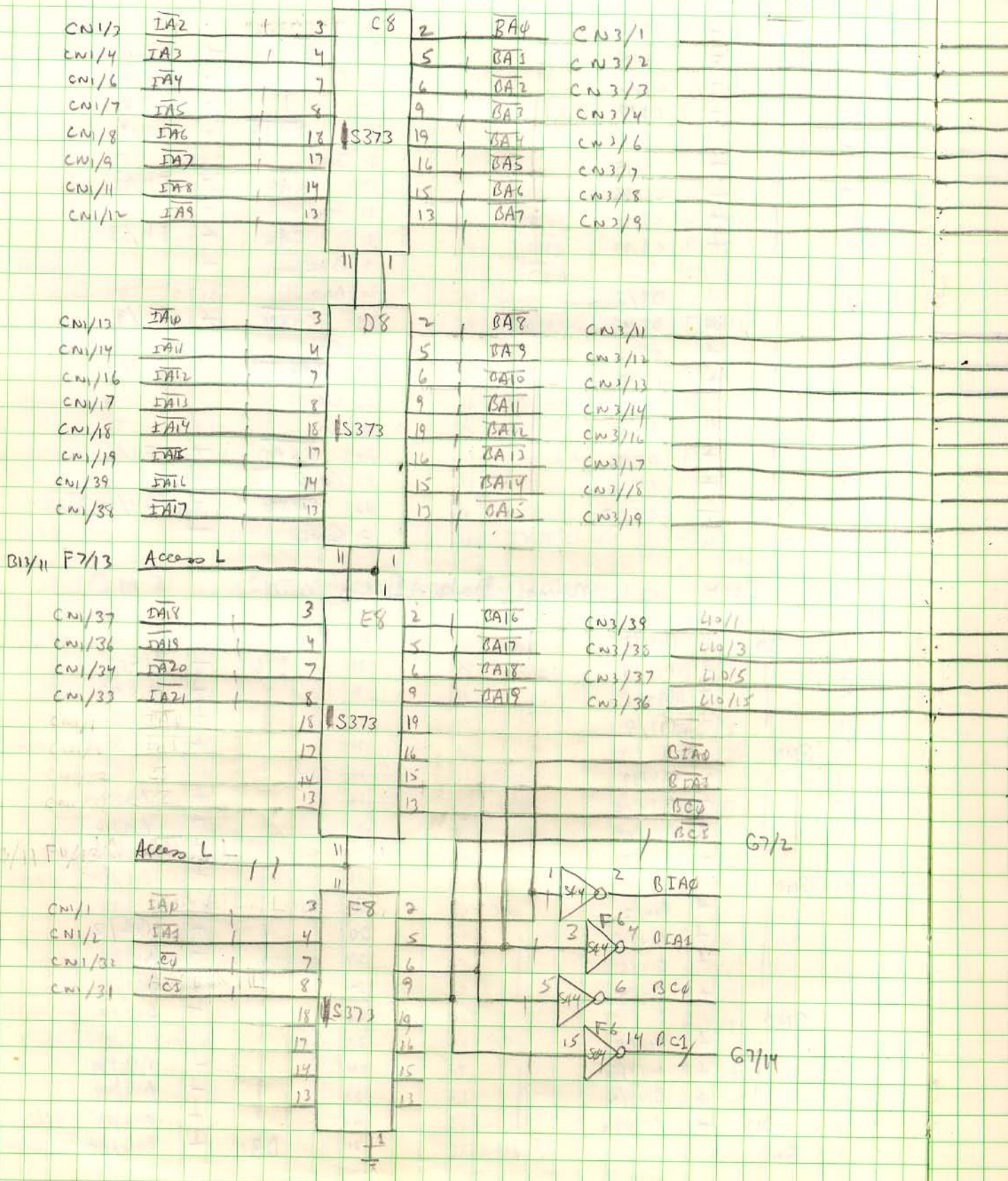
CN3 Memory Address / Control Bus

1	$\overline{BA0}$	+	C8/2	40	GND		
2	|	+	C8/5	39	$\overline{BA16}$	+	E8/2
3	|	+	C8/6	38	|	+	E8/5
4	|	+	C8/9	37	|	+	E8/6
5	GND			36	$\overline{BA19}$	+	E8/9
6	|	+	C8/19	35	GND		
7	|	+	C8/16	34	$\overline{RAS}$	+	F11/13
8	|	+	C8/15	33	GND	-	
9	|	+	C8/13	32	$\overline{CAS}$	+	F11/3
10	GND			31	GND	-	
11	|	+	D8/2	30	GND		
12	|	+	D8/5	29	$\overline{COLADR}$	+	F11/9
13	|	+	D8/6	28	GND		
14	|	+	D8/9	27	~~XRCVR~~	+	F7/9
15	GND			26	GND		
16	|	+	D8/19	25	GND		
17	|	+	D8/16	24	$\overline{READ}$	+	F11/6
18	|	+	D8/15	23	GND	-	
19	$\overline{BA15}$	+	D8/13	22	~~WRITE~~	+	G11/10
20	GND			21	GND	-	

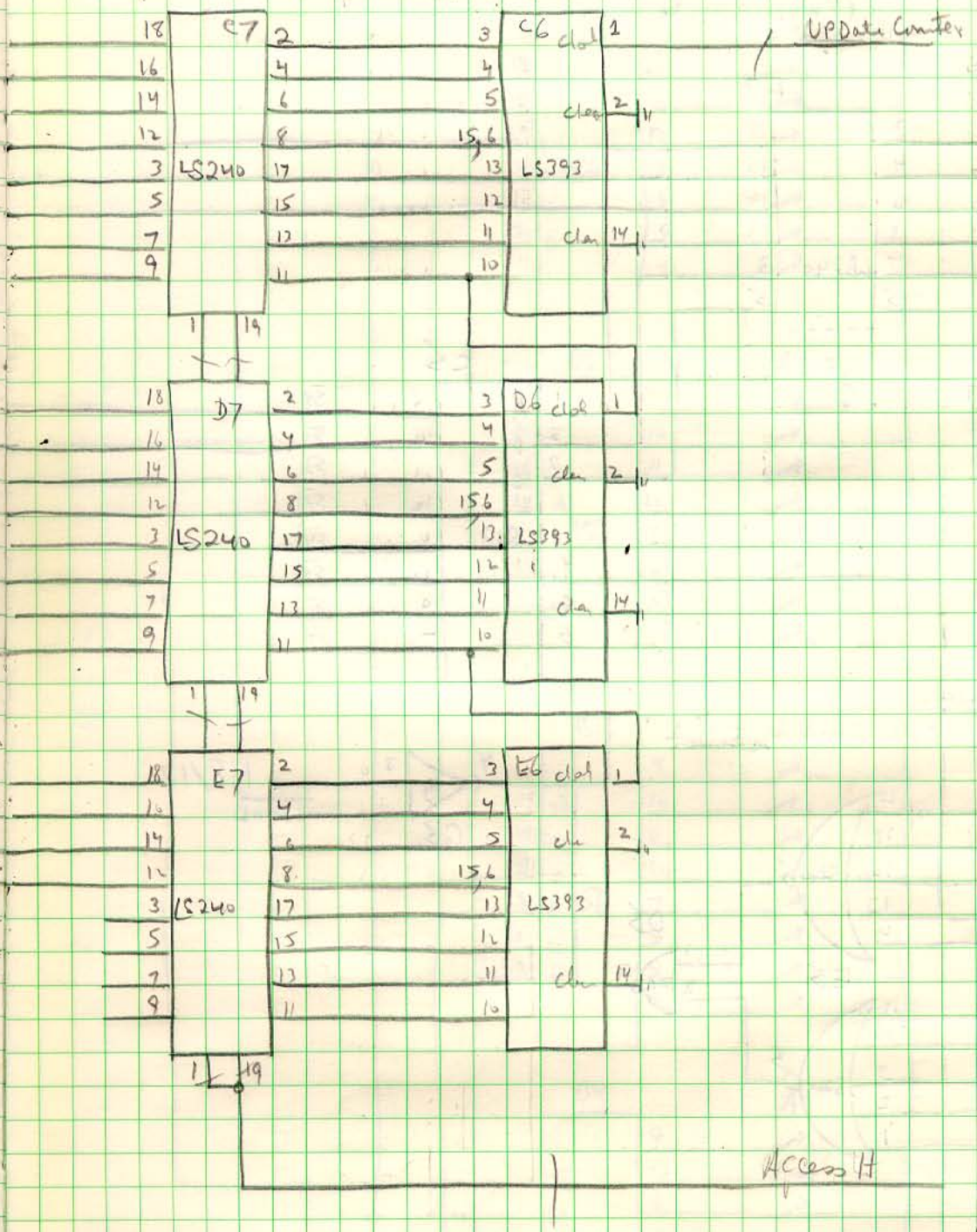
CN4 Memory Data Bus

1	D0	+	B13/2	40	GND		
2	|	+	B13/5	39	D16	+	B15/2
3	|	+	B13/6	38	|	+	B15/5
4	|	+	B13/9	37	|	+	B15/6
5	GND			36	|	+	B15/9
6	|	+	B13/19	35	GND		
7	|	+	B13/16	34	|	+	B15/19
8	|	+	B13/15	33	|	+	B15/16
9	|	+	B15/12	32	|	+	B15/15
10	GND			31	|	+	B15/12
11	|	+	B14/2	30	GND		
12	|	+	B14/5	29	|	+	B16/2
13	|	+	B14/6	28	|	+	B16/5
14	|	+	B14/9	27	|	+	B16/6
15	GND			26	|	+	B16/9
16	|	+	B14/19	25	GND		
17	|	+	B14/16	24	|	+	B16/19
18	|	+	B14/15	23	|	+	B16/16
19	D15	+	B14/12	22	|	+	B16/15
20	GND			21	D31	+	B16/12

CPU Address Latch

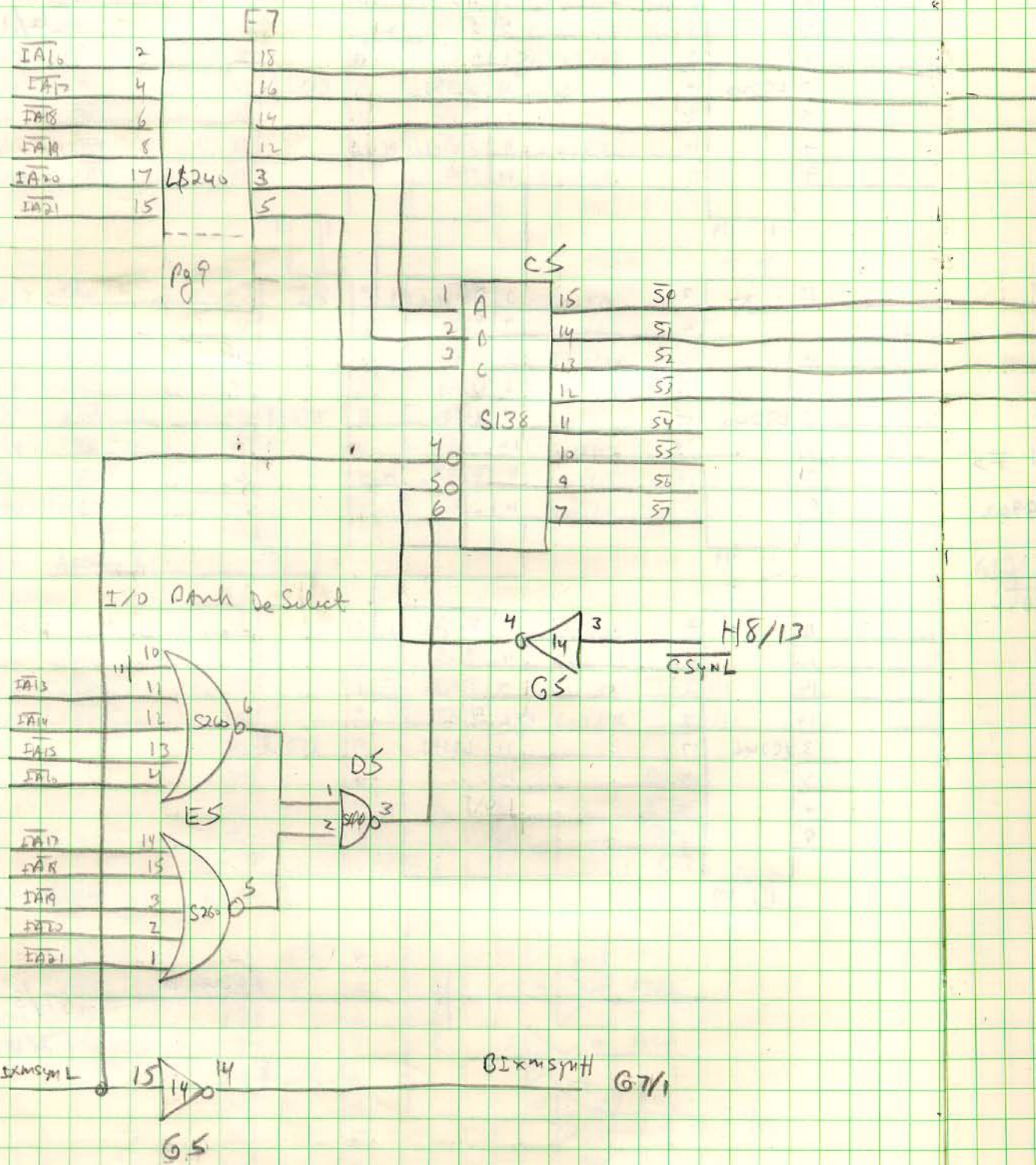


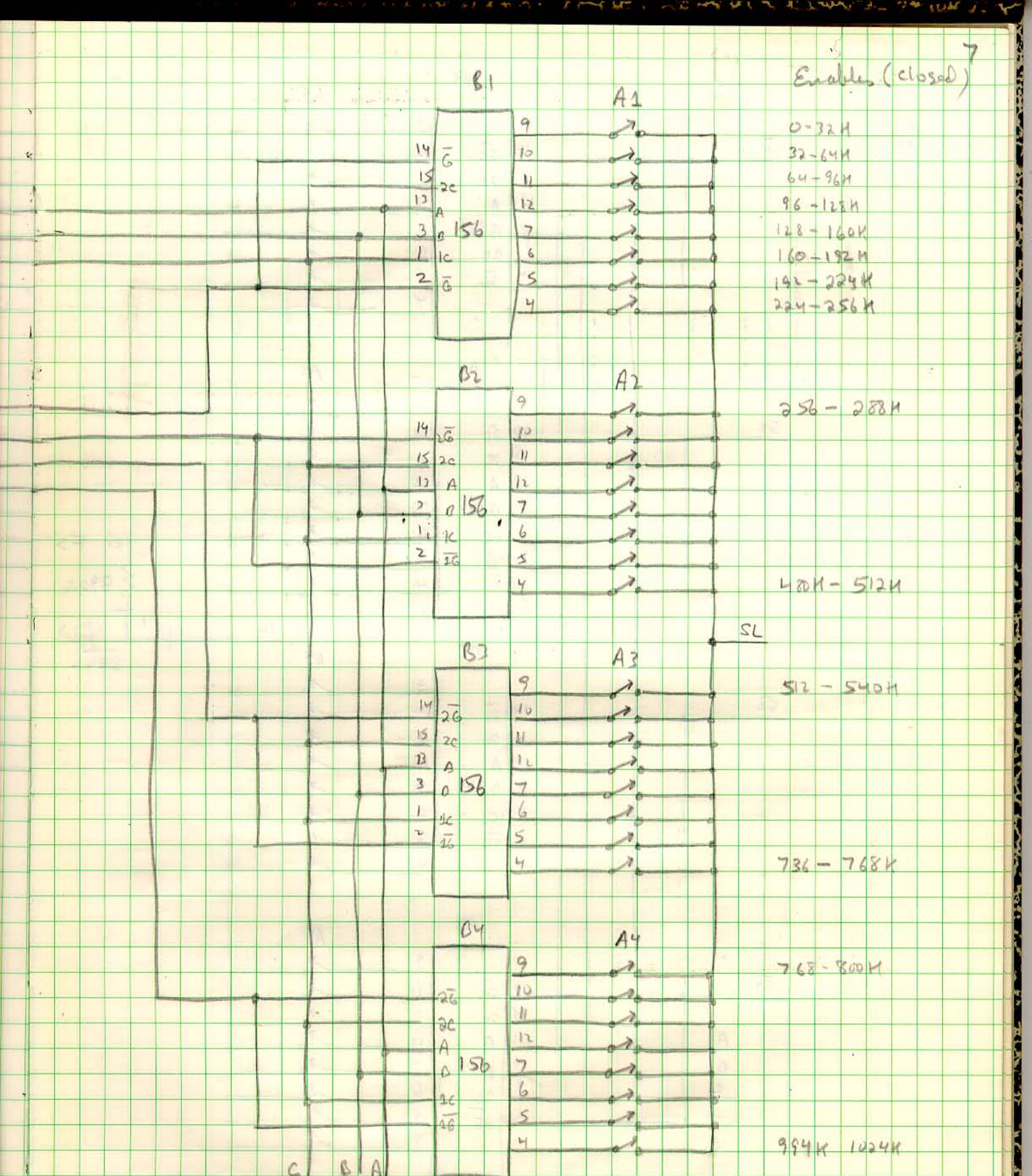
Refresh Address Counter Buffer



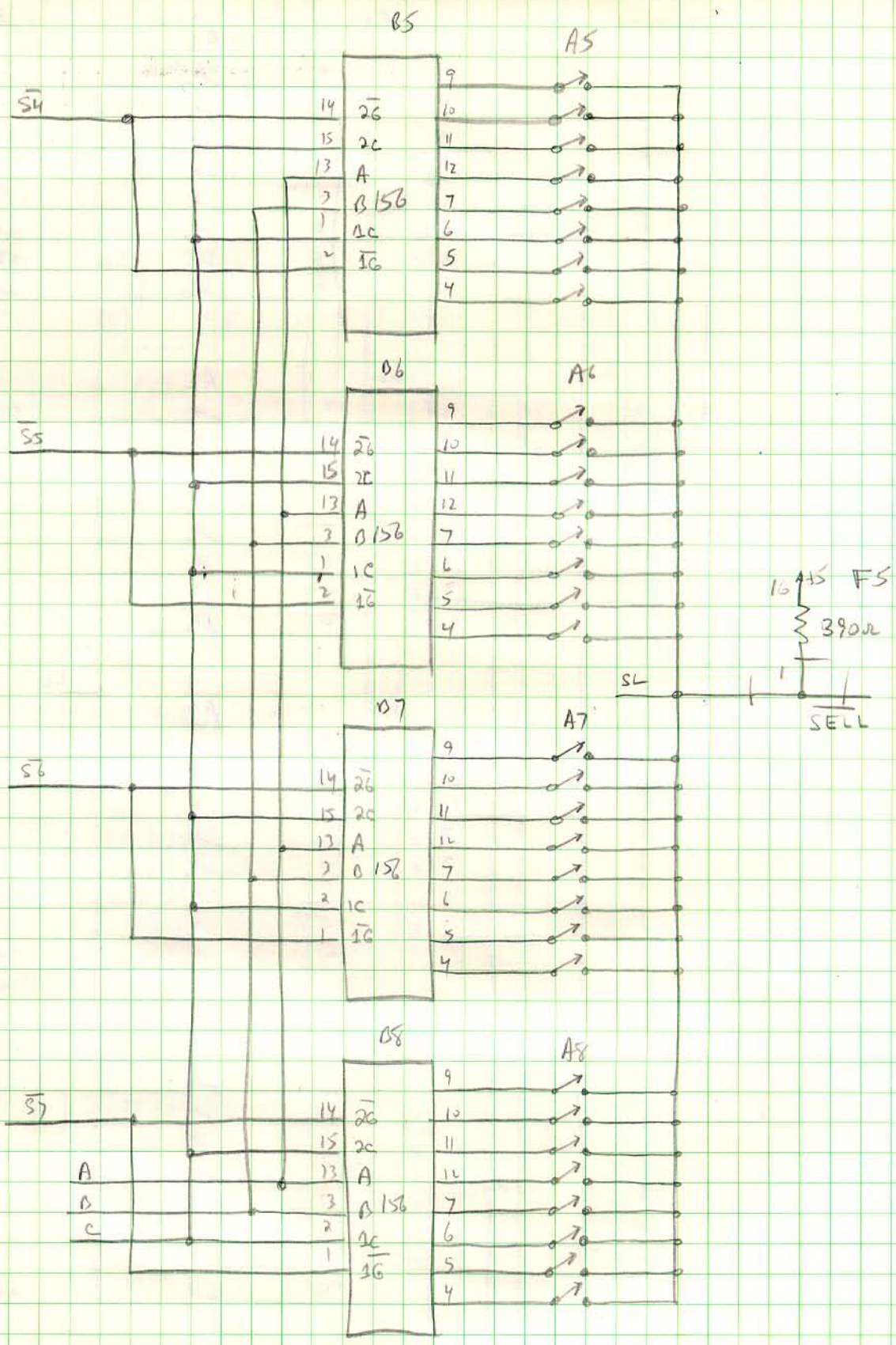
24 Oct 82
ARD

Address Decoding (32K Word Segments)
(64KB Segments)




Enables (closed)
B1
A1
156
0-32K
32-64K
64-96K
96-128K
128-160K
160-192K
192-224K
224-256K
B2
A2
256-288K
480K-512K
SL
B3
A3
512-540K
736-768K
B4
A4
768-800K
992K 1024K
C
B
A

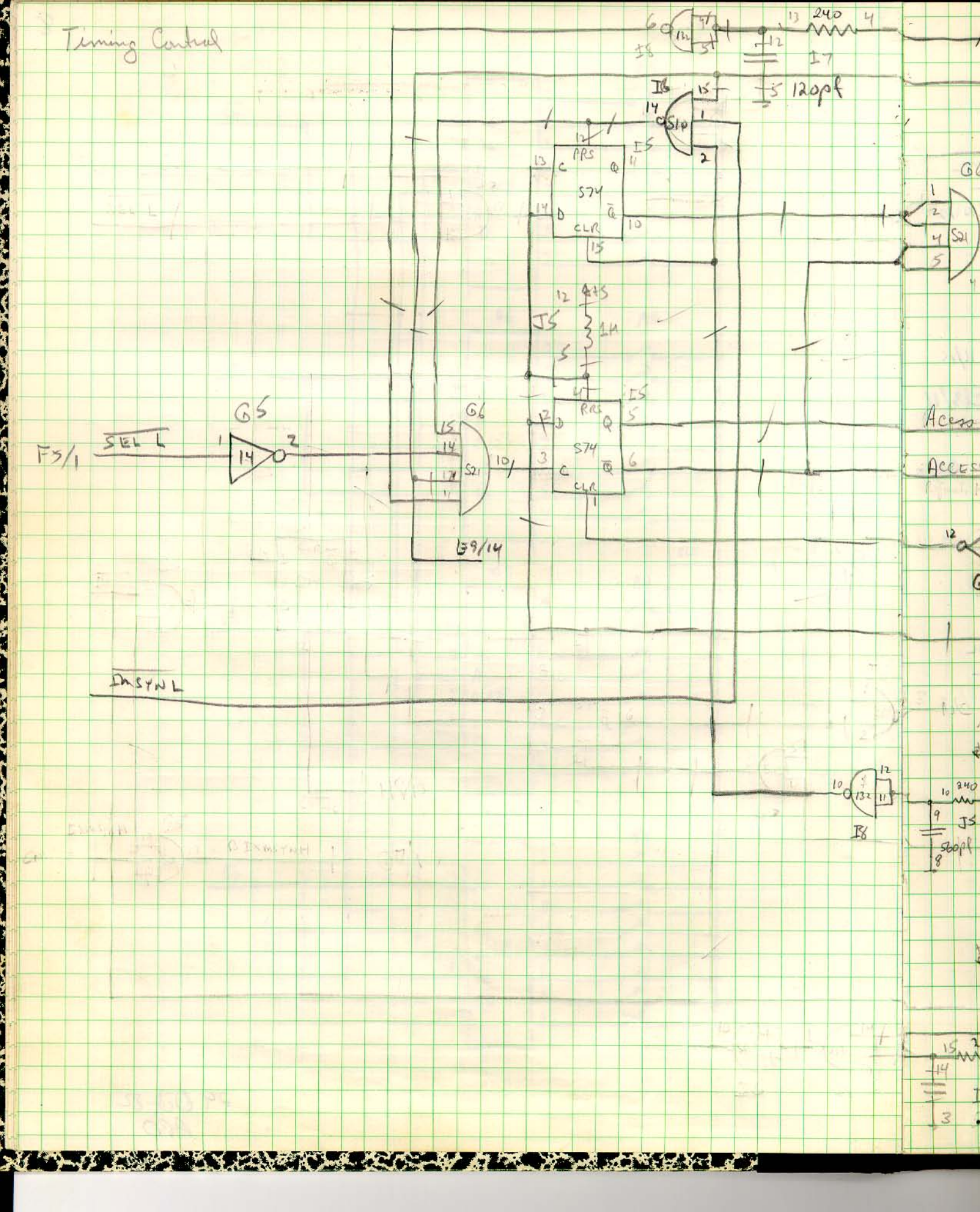
24 Oct 82
ARD

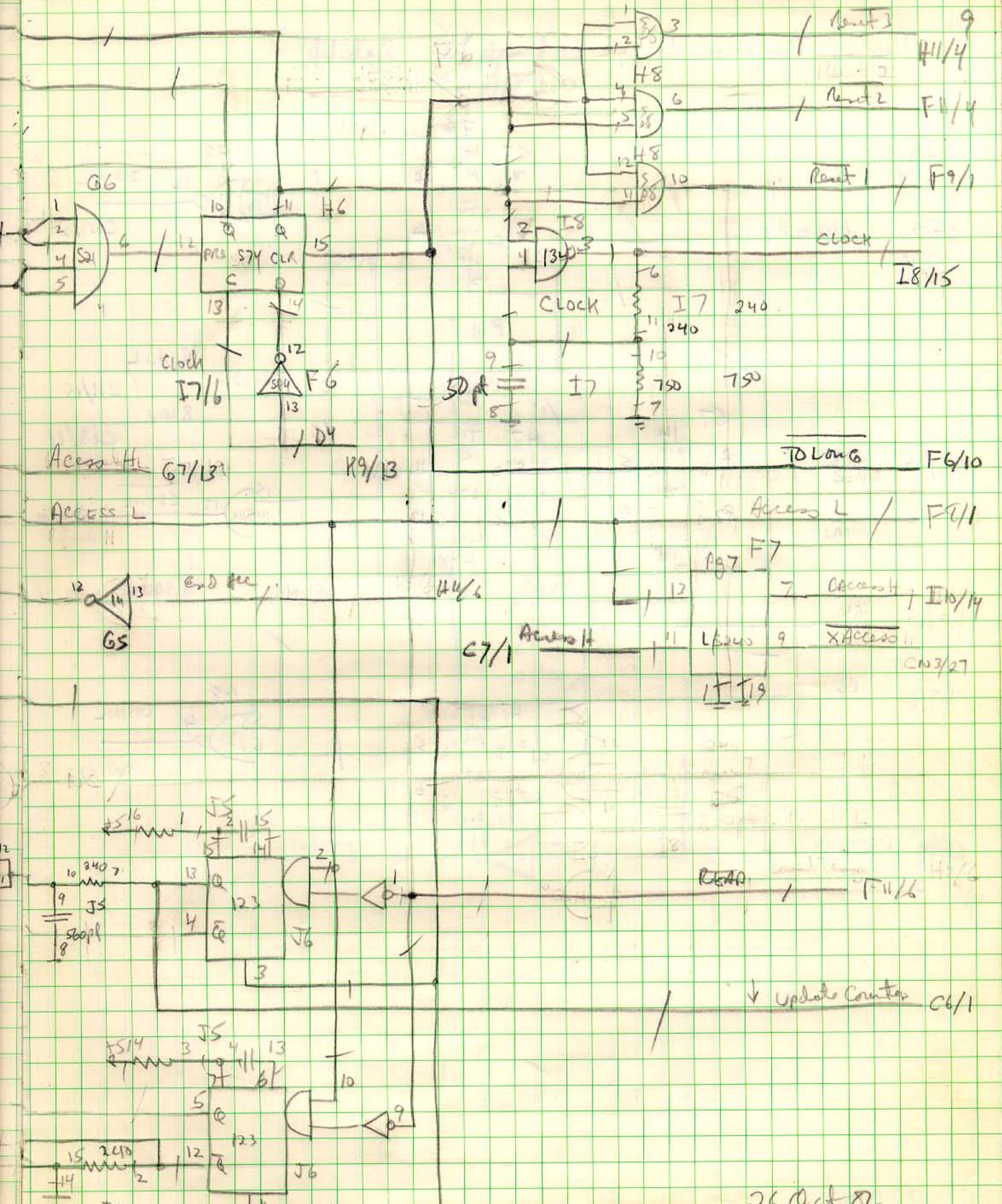

B5
A5
S4
14 2G
15 2C
13 A
3 B 1Sb
1 1C
2 1G
9
10
11
12
7
6
5
4
B6
A6
S5
B7
A7
S6
B8
A8
S7
A
B
C
16 +5 F5
390Ω
SL
SELL

65/1

65/1

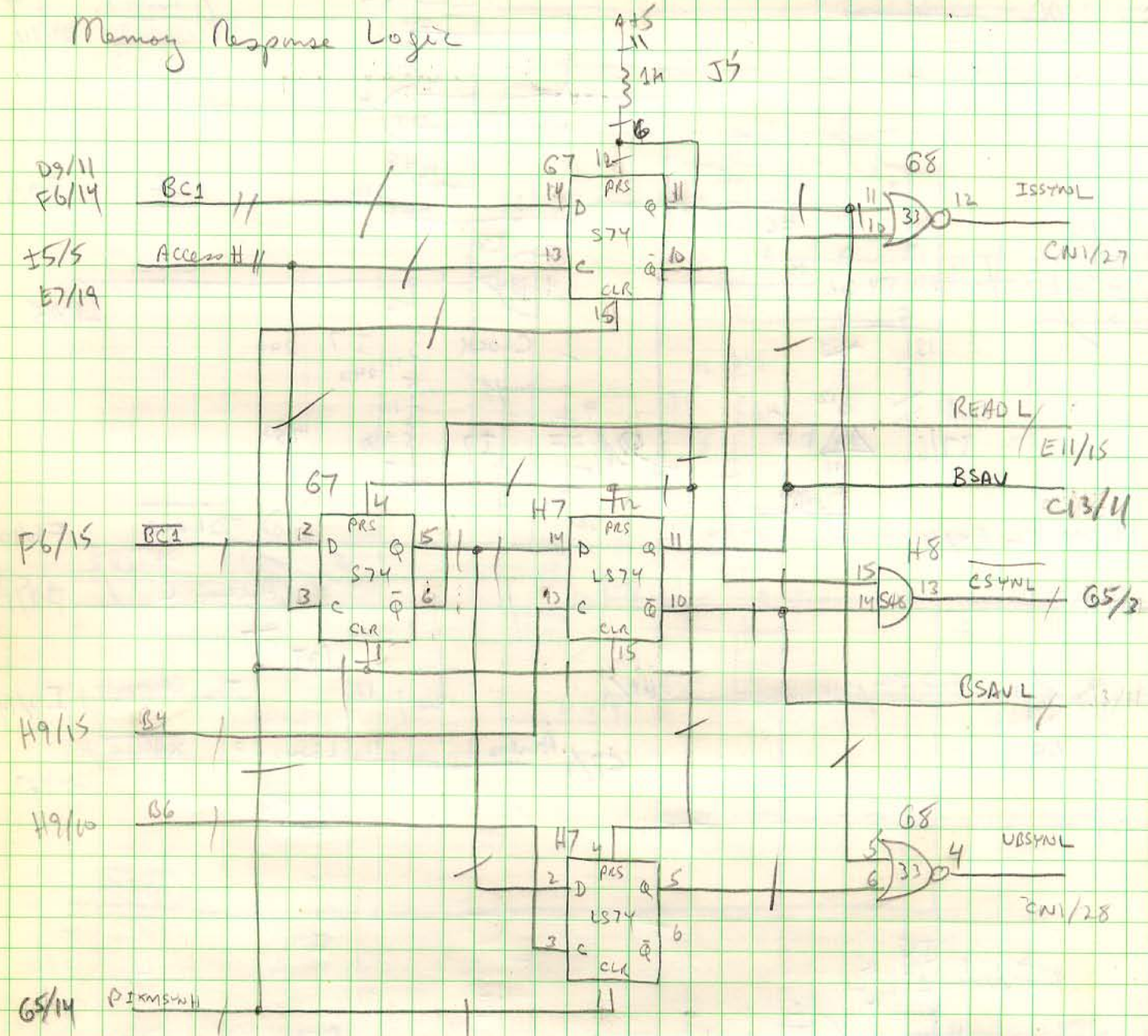
24 Oct 82
ABD


Timing Control
240
120pf
S74
CLR
S21
SEL L
FS/1
DMSYN L
Acces
ACCES

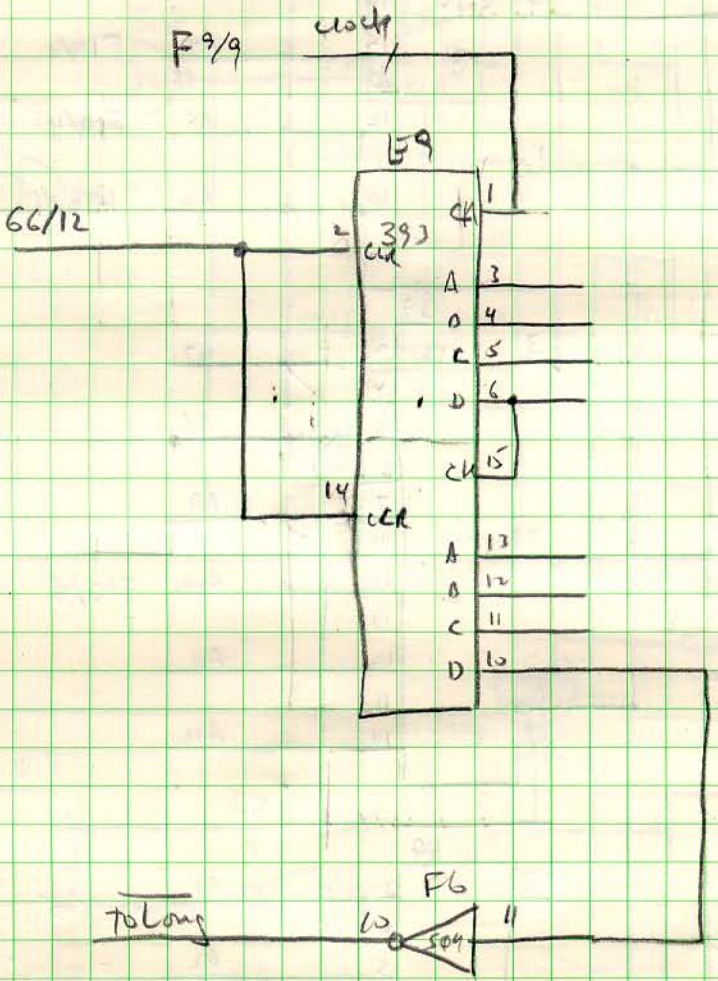

Reset 3
H11/4
H8
Reset 2
F11/4
Reset 1
F9/1
G6
H6
PRS S74 CLR
CLOCK
I8
I8/15
CLOCK
I7
240
750
50 pf
Clock
I7/6
F6
D4
Access HL
G7/13
K9/13
TO LONG
F6/10
ACCESS L
Access L
F7/1
F7
End Acc
H4/6
G5
CAccess H
I10/14
C7/1
Access H
X Access
CN 3/27
I19
J5
J6
560pf
READ
F11/6
Update Counter
C6/1
I7
.01 mf

26 Oct 82
ARB

Memory Response Logic

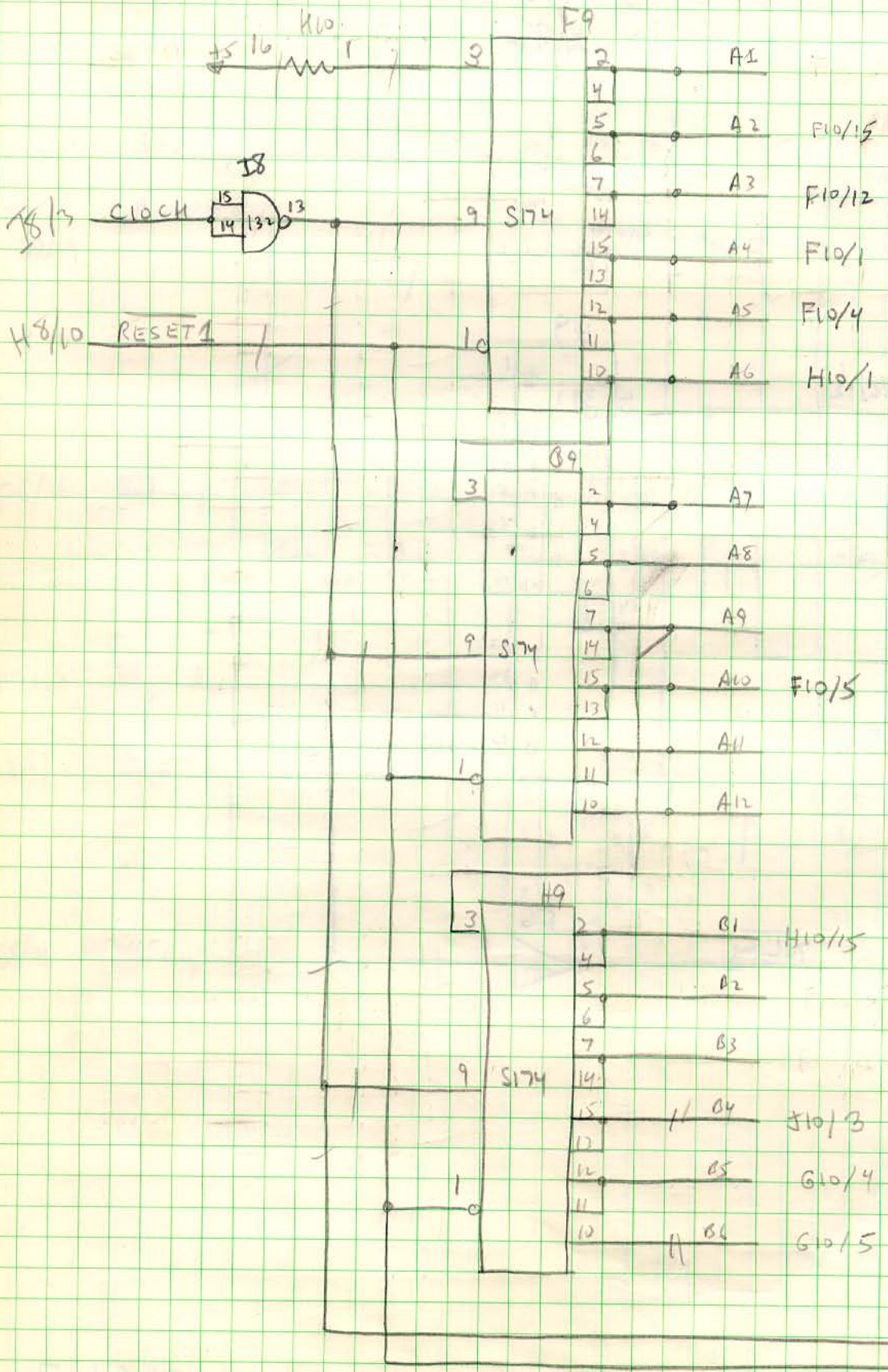


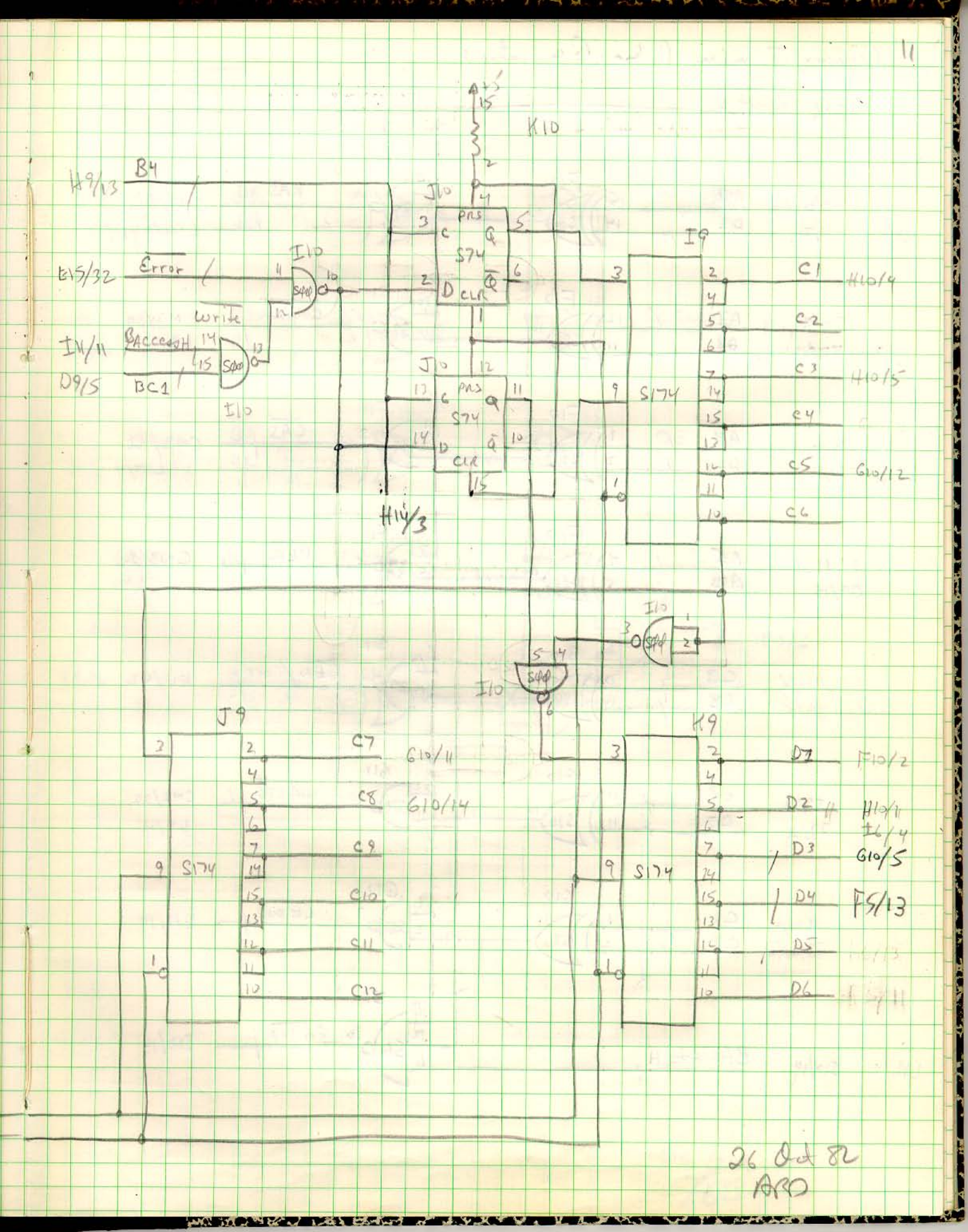
Added 25 April 83
AHD



26 Oct 82
AHD

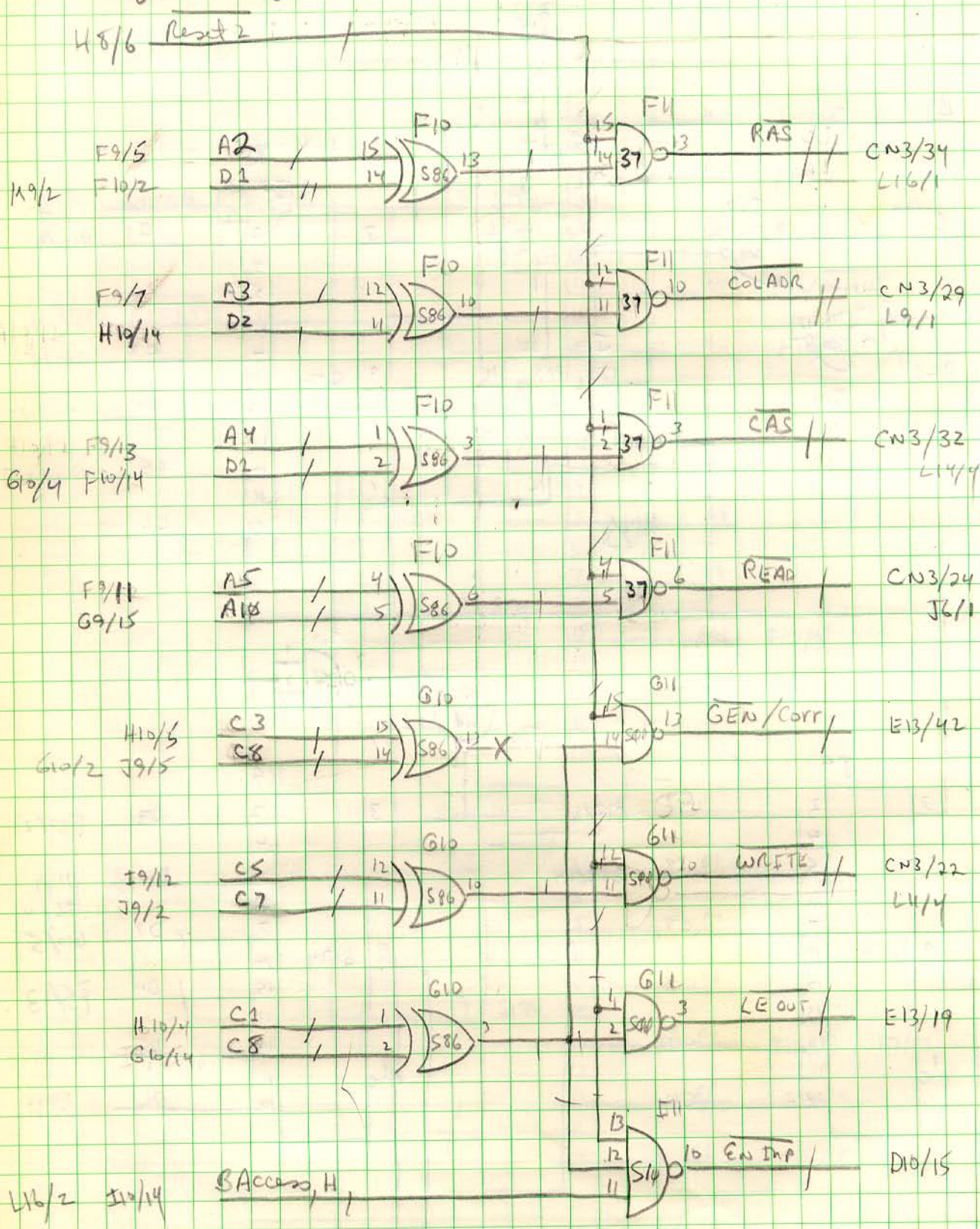
Memory Timing Chain

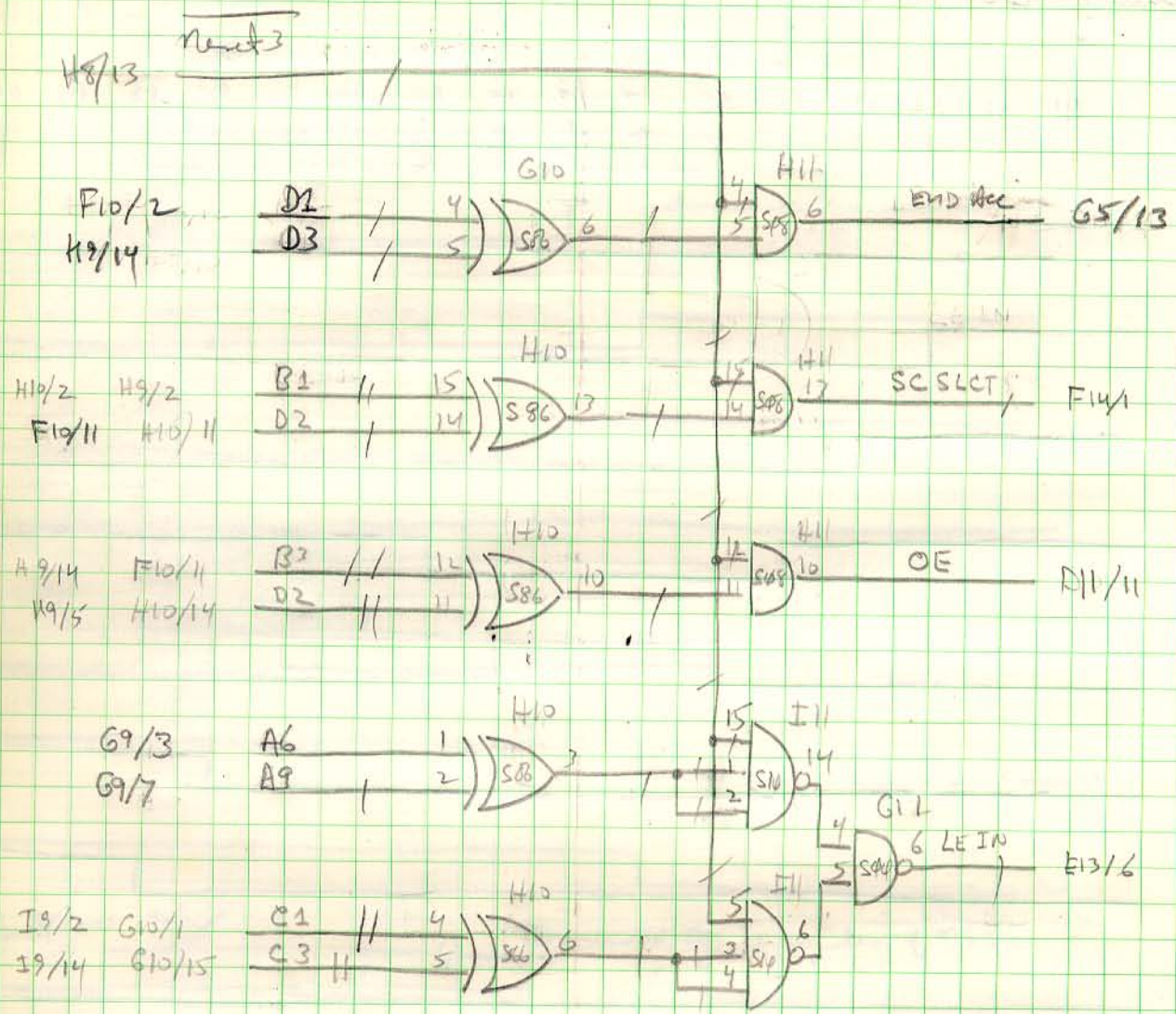



K10
H9/13
B4
J10
S74
I10
S00
Error
E15/32
Write
BAccessH
BC1
I4/11
D9/5
H14/3
I9
S174
C1
C2
C3
C4
C5
C6
H10/4
H10/5
G10/12
J9
C7
C8
C9
C10
C11
C12
G10/11
G10/14
K9
D1
D2
D3
D4
D5
D6
F10/2
H10/11
I6/4
G10/5
F5/13

26 Oct 82
ARO

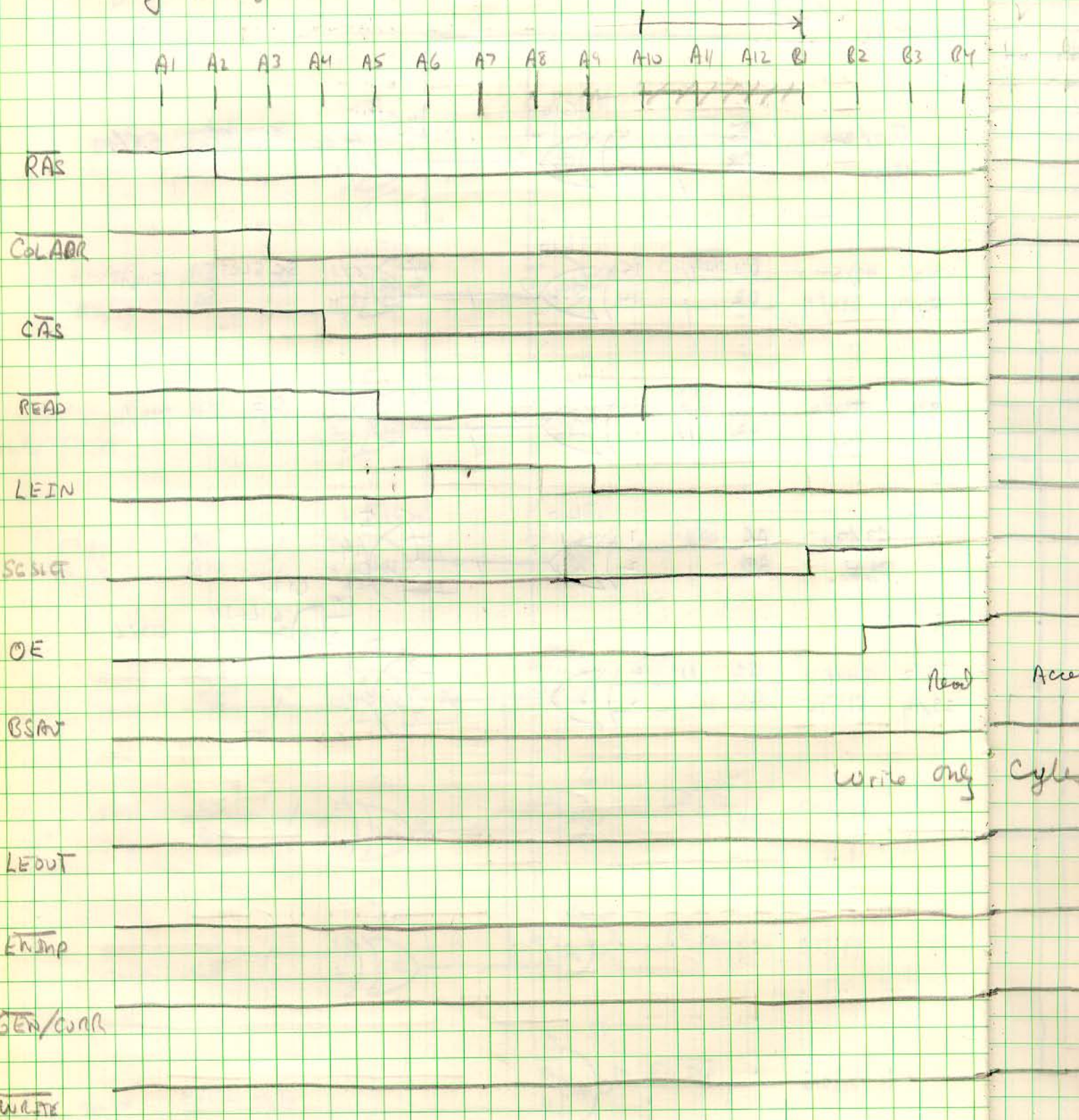
Memory Timing Pulse Generators

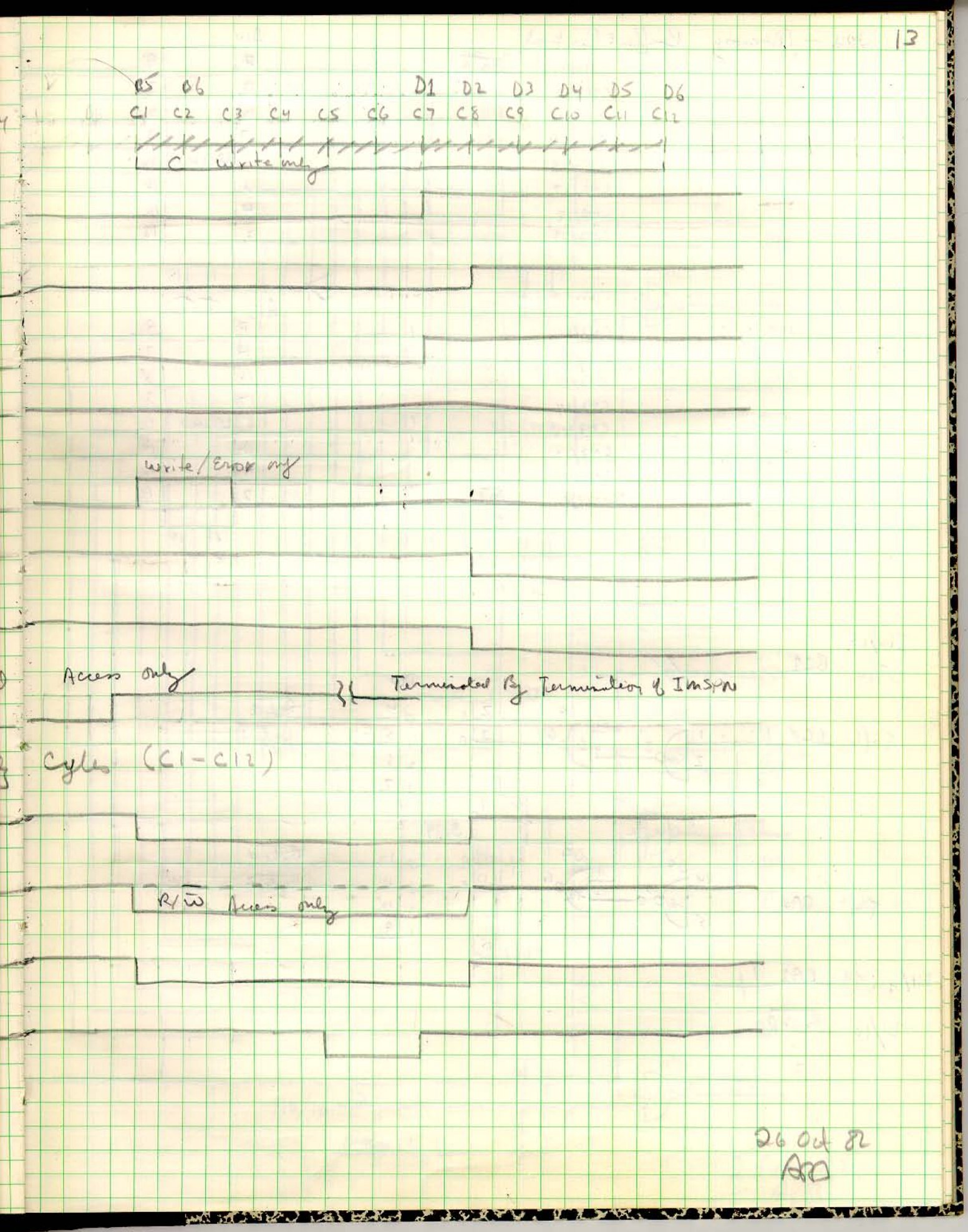



Next3
H8/13
G10
H11
F10/2
H9/14
D1
D3
S86
END ACC
G5/13
H10
H11
H10/2
H9/2
F10/11
H10/11
B1
D2
S86
SC SLCT
F14/1
H10
H11
H9/14
F10/11
H9/5
H10/14
B3
D2
S86
OE
D11/11
H10
I11
G9/3
G9/7
A6
A9
S86
S10
G11
LE IN
E13/6
H10
I11
I9/2
G10/1
I9/14
G10/15
C1
C3
S86
S10

26 Oct 82
ABR

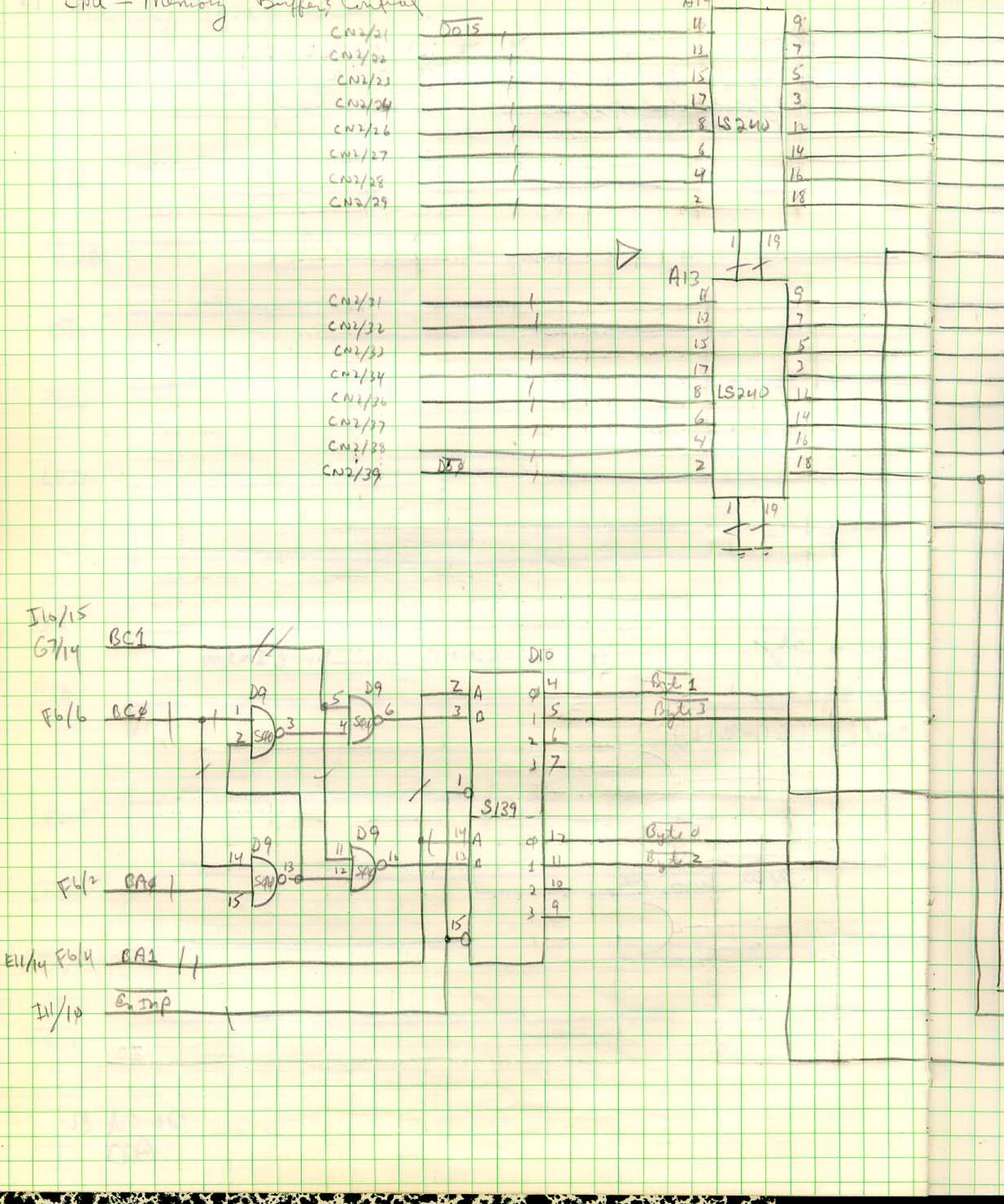
Timing Diagram

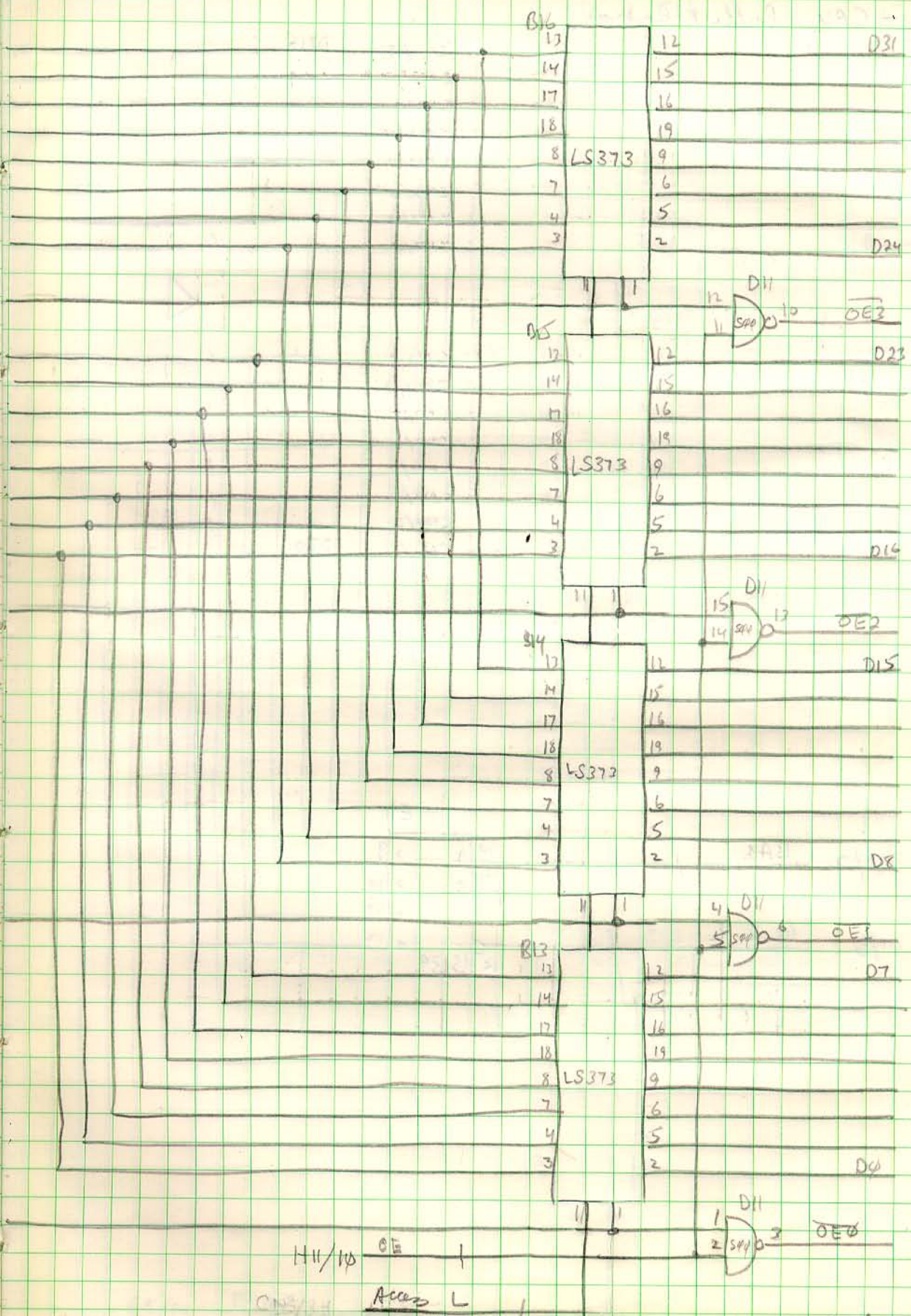



D5 D6 D1 D2 D3 D4 D5 D6
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12
C
Write only
Write/Error only
Access Only
Terminated By Termination of IMSPN
Cycles (C1-C12)
R/W Access only

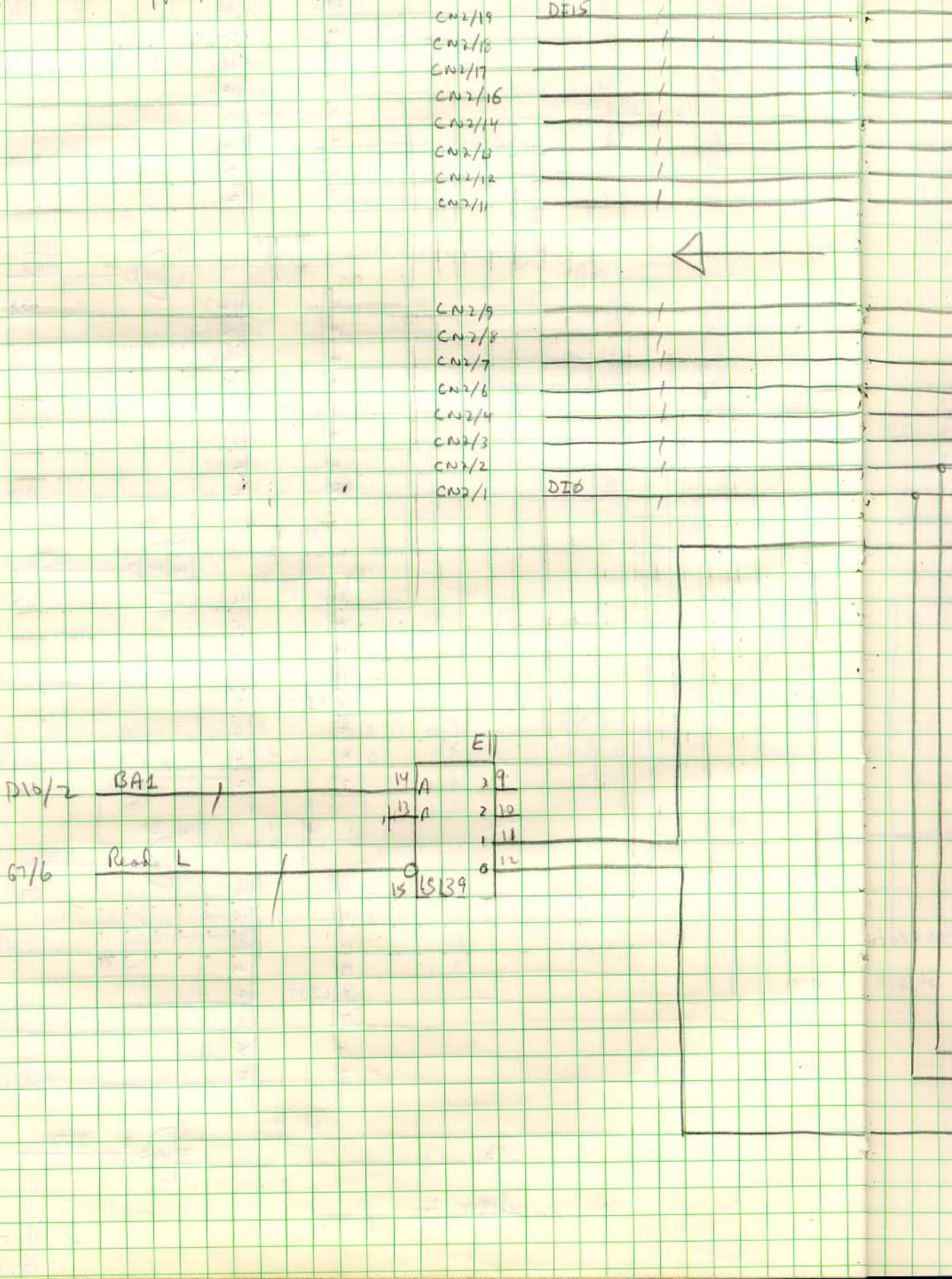
26 Oct 82
ARD

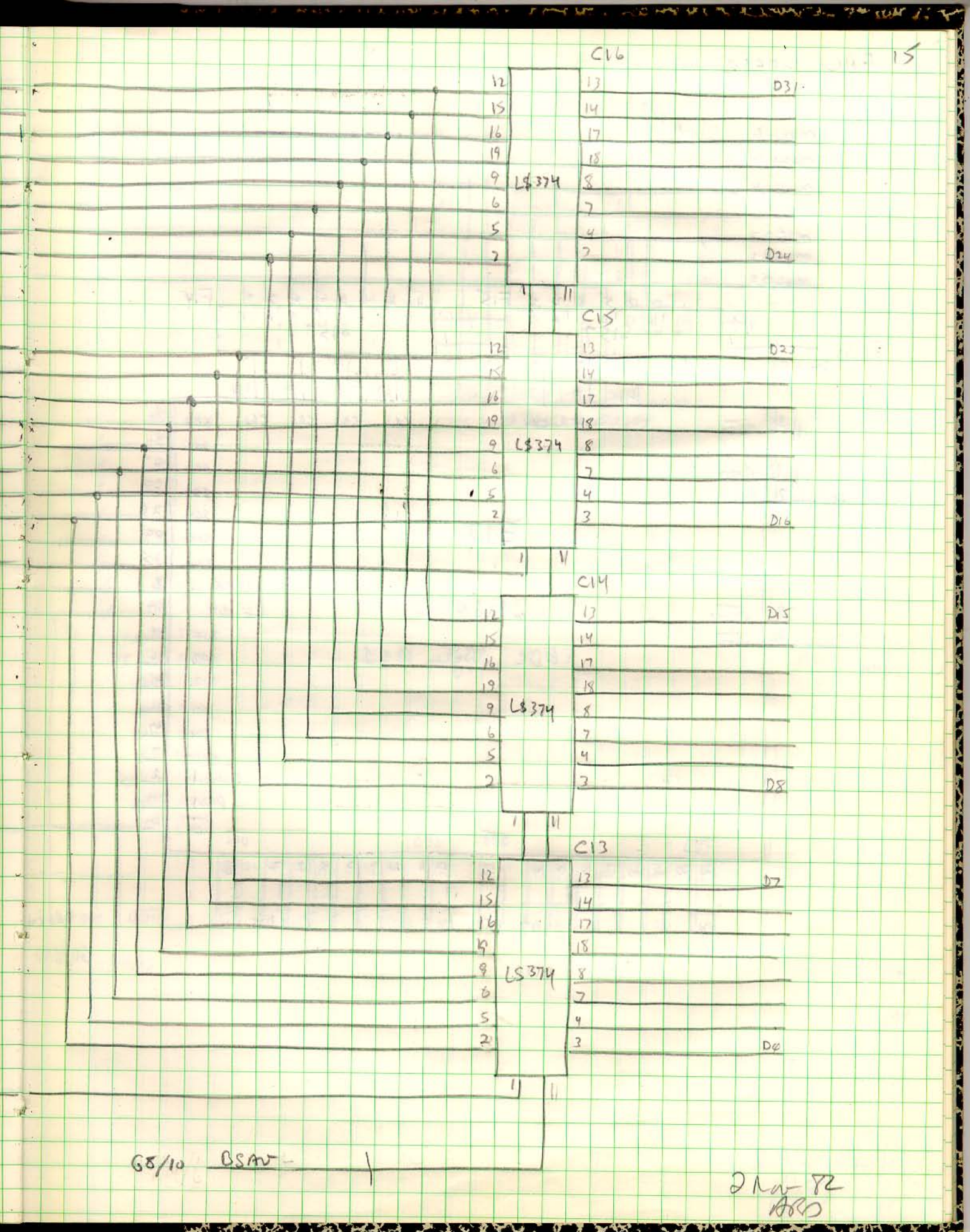
CPU – Memory Buffer & Control



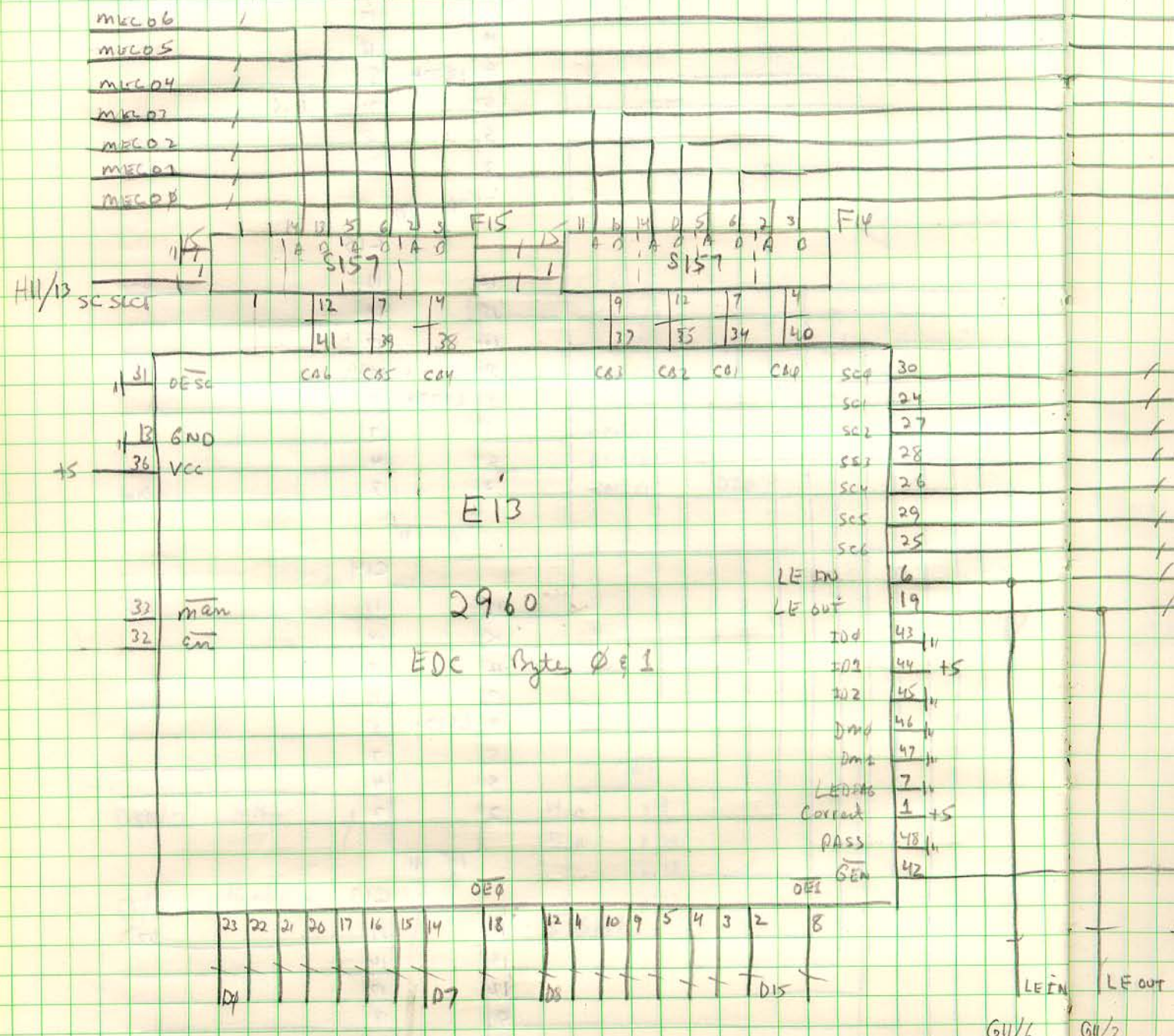

B16
LS373
D31
D24
D11
OE3
D23
LS373
D16
D11
OE2
D15
LS373
D8
D11
OE1
B13
LS373
D7
D0
D11
OE0
H11/10
OE
Access L
2 Nov 82
ARB

Memory - CPU Buffer & Control



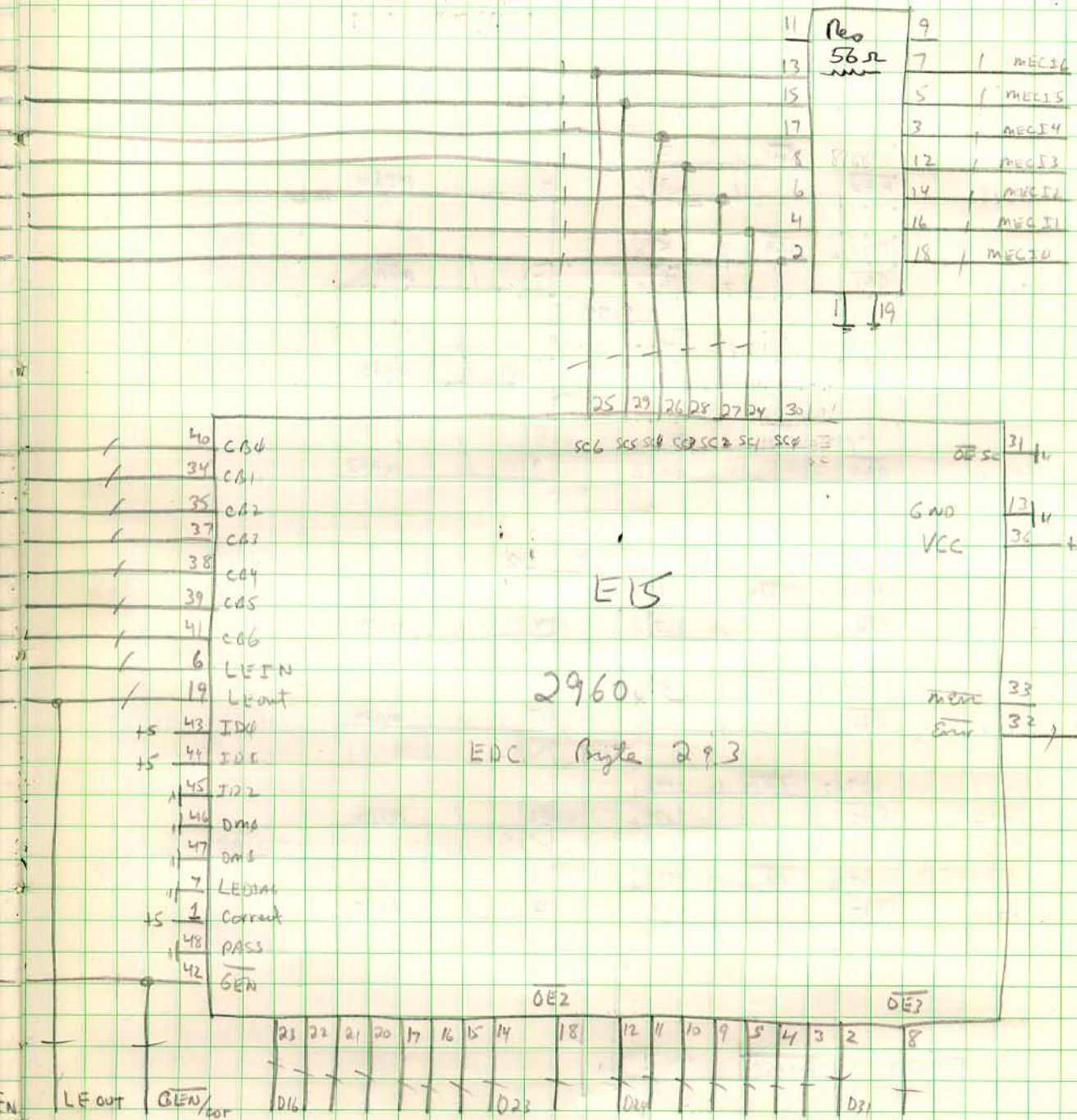

C16
LS374
D31
D24
C15
LS374
D23
D16
C14
LS374
D15
D8
C13
LS374
D7
D0
G8/10 OSAV
21 Nov 82

EDC Logic



27
16
45

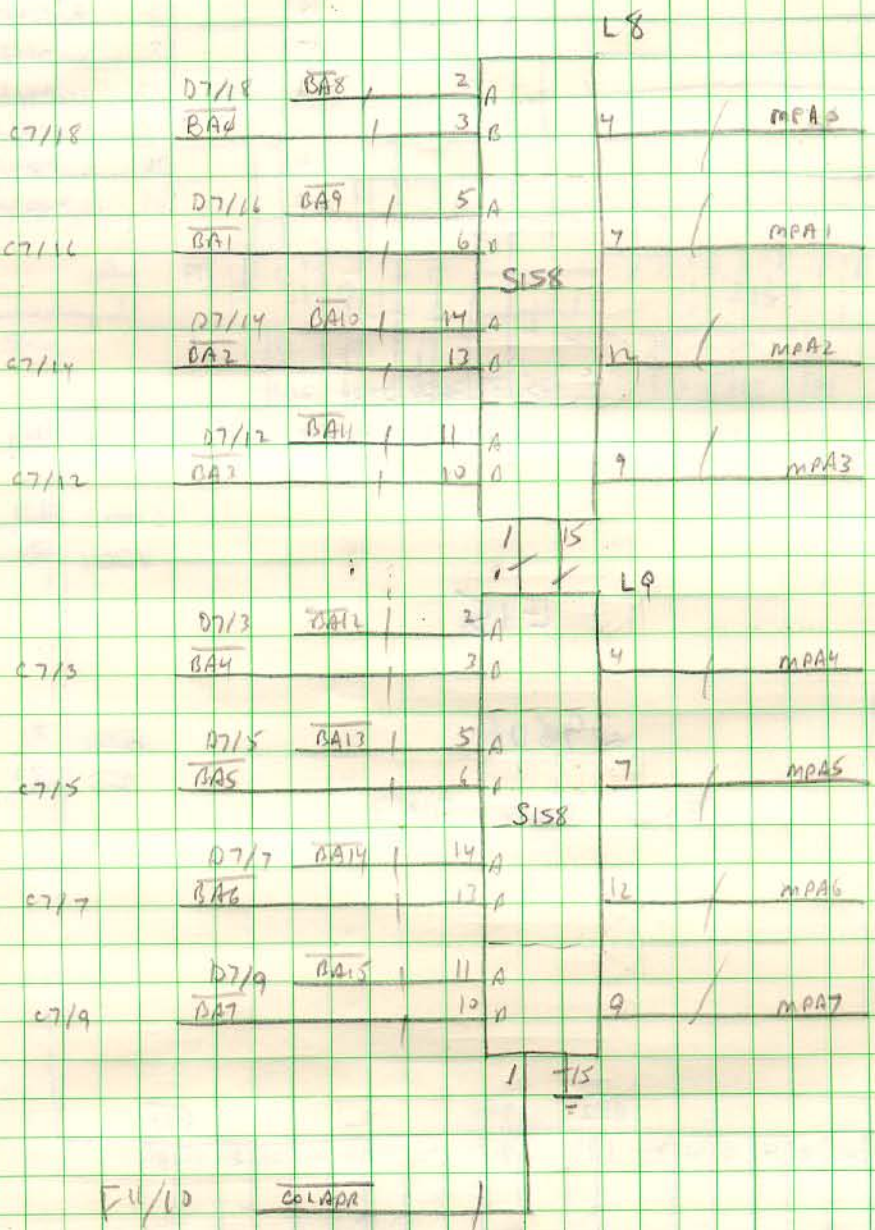
M16

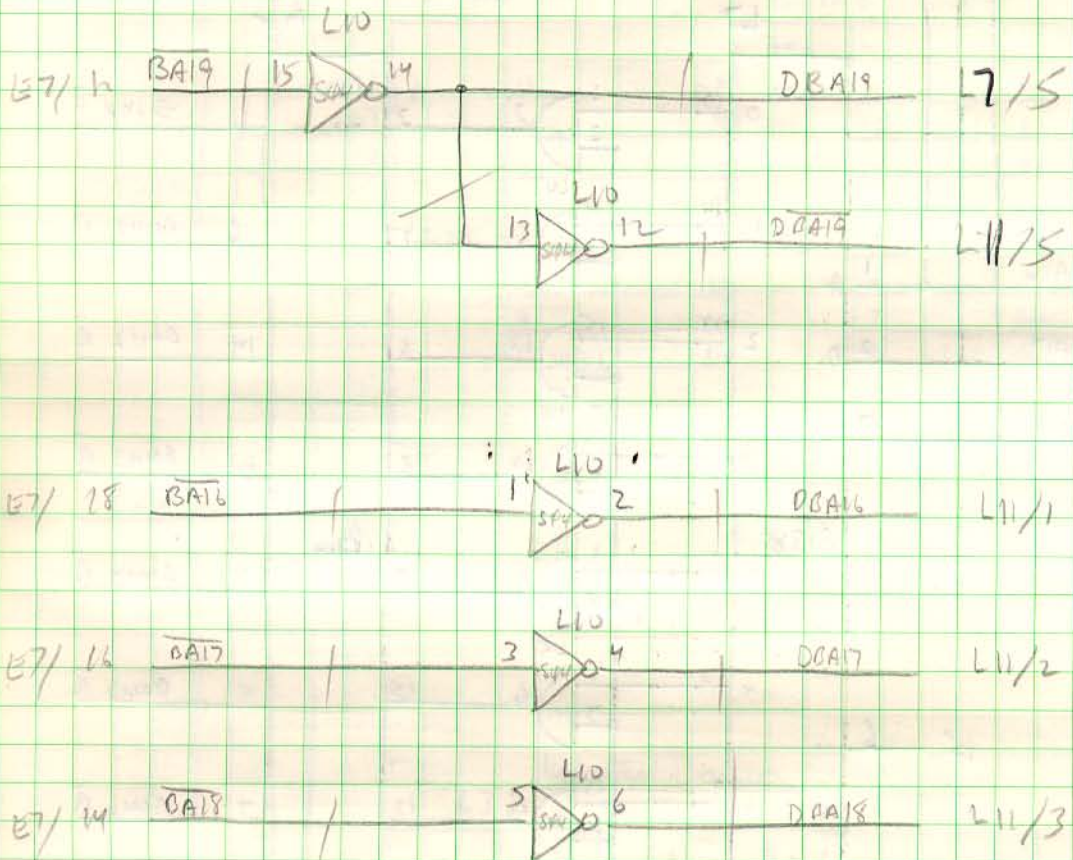


2 Nov 82

ARD

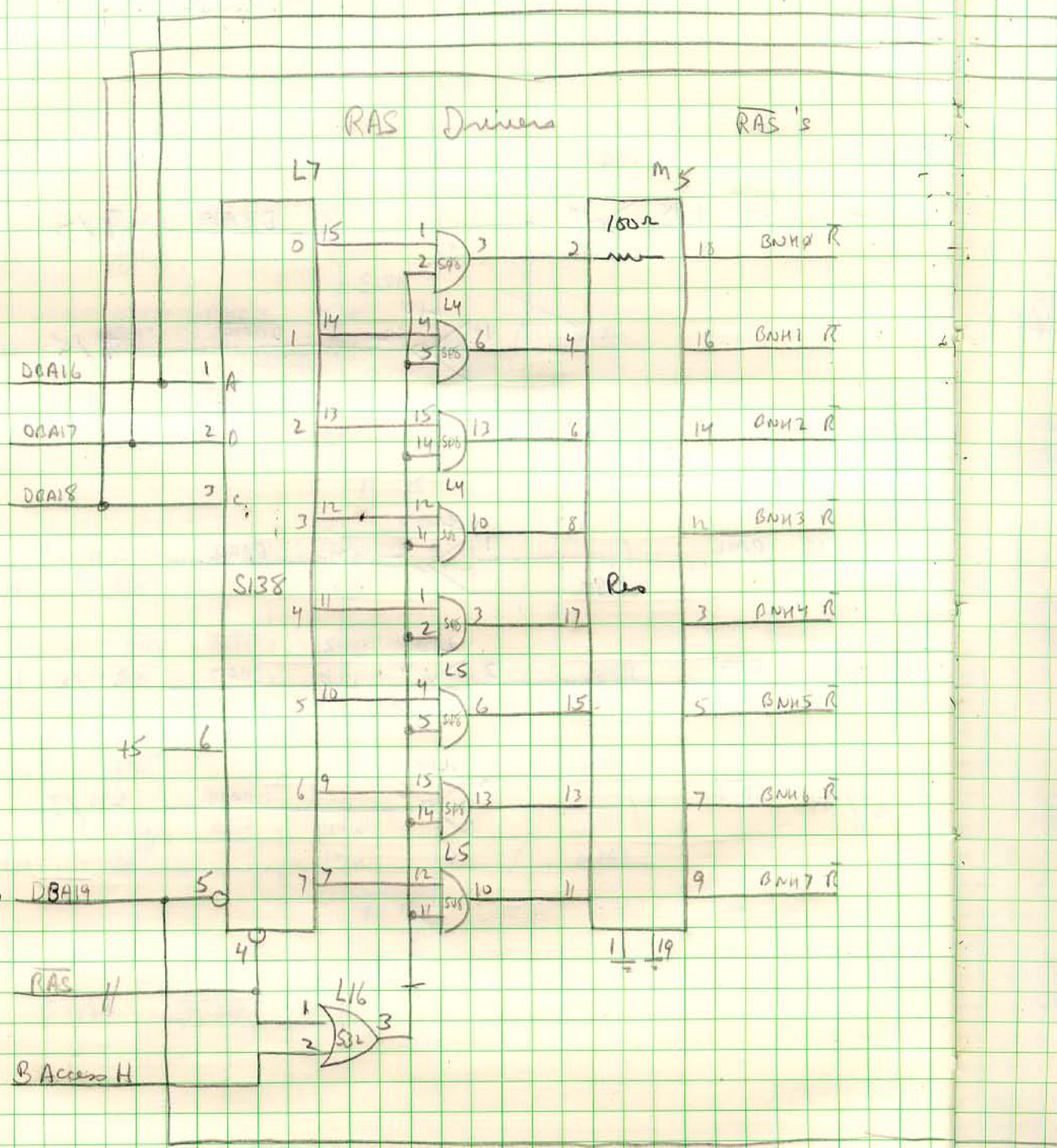
Error Code Memory Logic

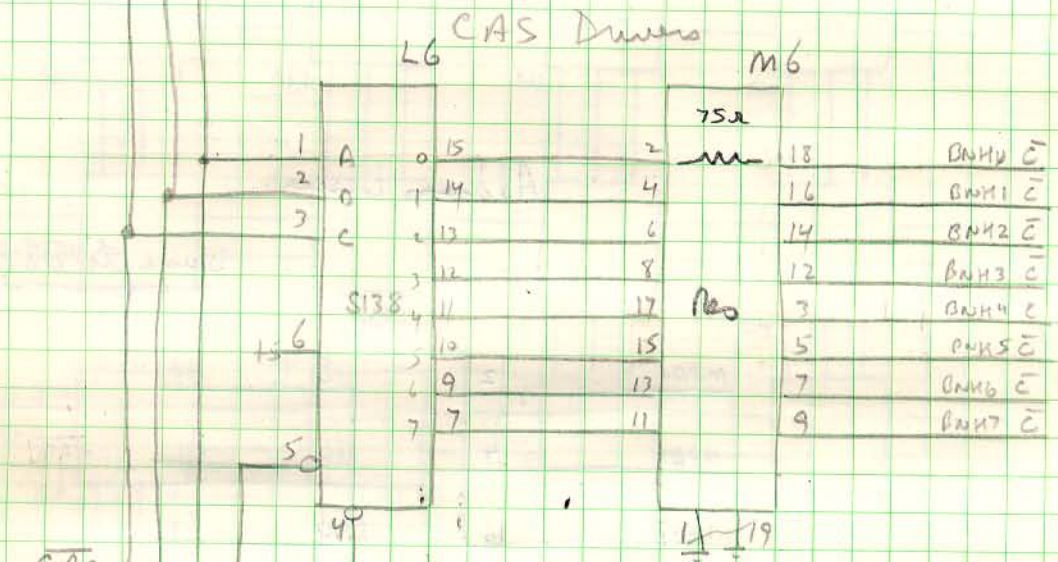


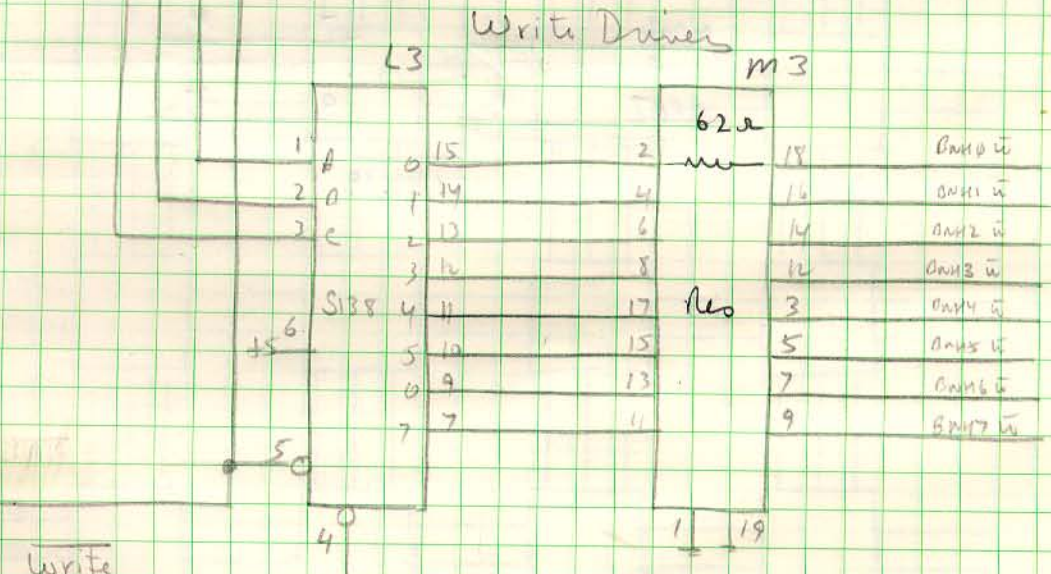

L10
E7/ 12
BA19
15
14
DBA19
L7/5
L10
13
12
DBA19
L11/5
L10
E7/ 18
BA16
1
2
DBA16
L11/1
L10
E7/ 16
BA17
3
4
DBA17
L11/2
L10
E7/ 14
BA18
5
6
DBA18
L11/3

3 Nov 82
ARO

EC Memory 0 – 2 meg Bytes




CAS Drivers
L6
M6
S138
75Ω
Res
CAS

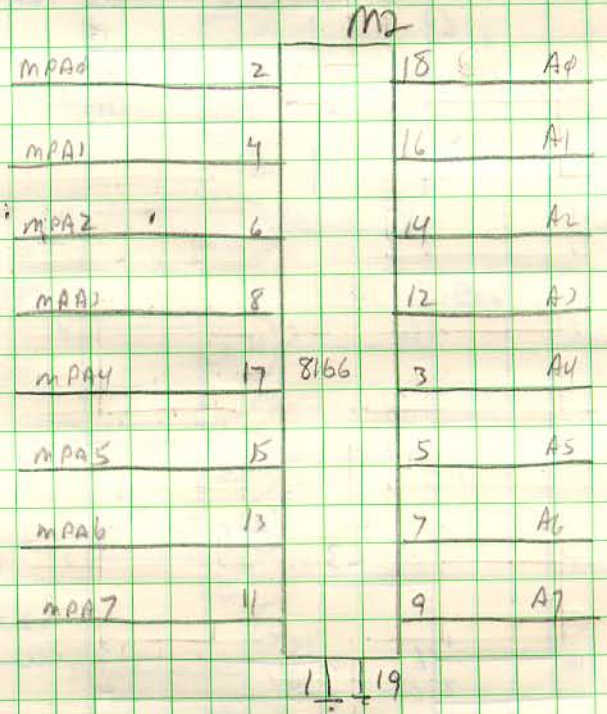

Write Drivers
L3
M3
S138
62Ω
Res
Write

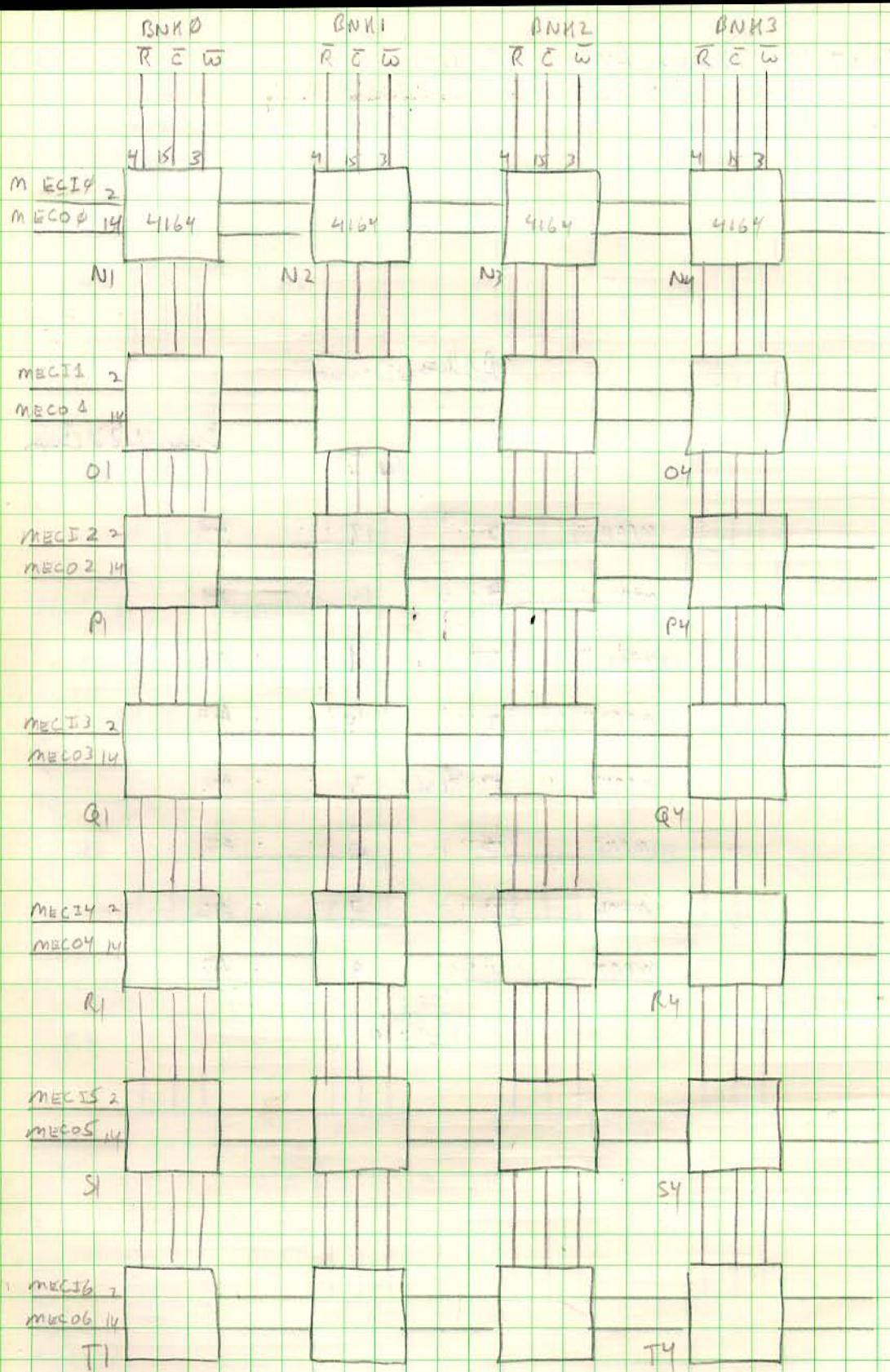
3 Nov 82
ABD

ANH0 – BNH3 Memory Array (ECC)

Address Driver

Drive to 28 chips




BNK0
BNK1
BNK2
BNK3
R C W
4 15 3
MEGI4 2
MEGO4 14
4164
N1
N2
N3
N4
MECI1 2
MECO1 14
O1
O4
MECI2 2
MECO2 14
P1
P4
MECI3 2
MECO3 14
Q1
Q4
MECI4 2
MECO4 14
R1
R4
MECI5 2
MECO5 14
S1
S4
MECI6 2
MECO6 14
T1
T4

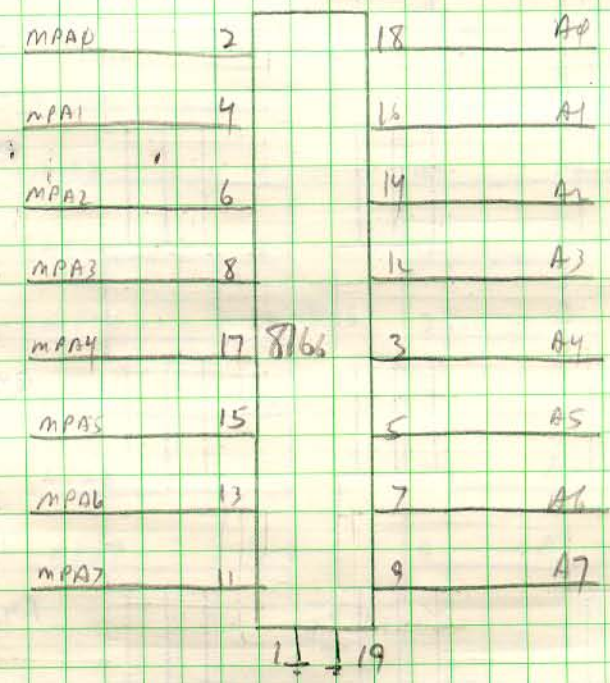
BNW 82
ARD

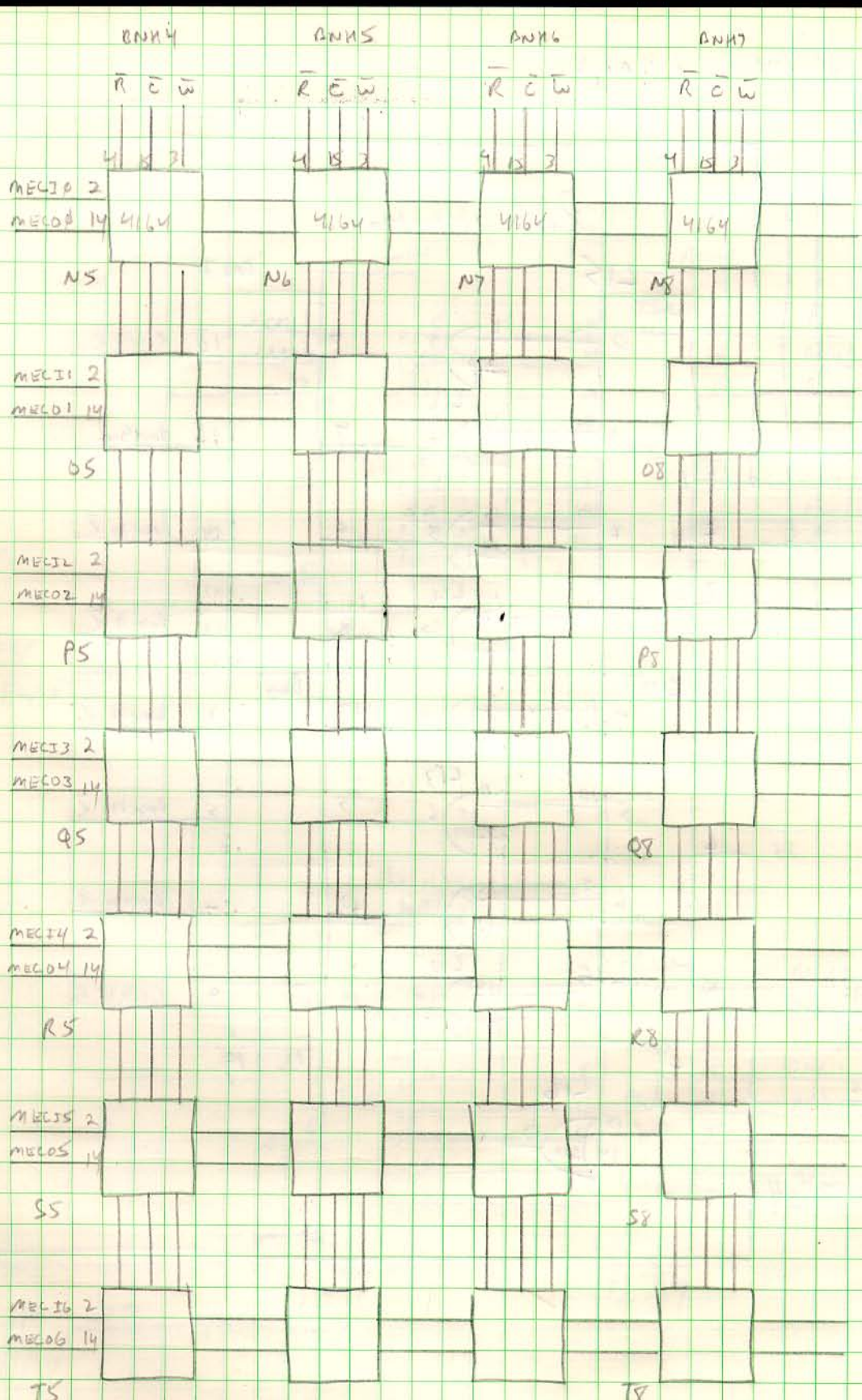
BNH4 - BNH7 Memory Array (EC)

Address Drivers

Drive 128 chips.

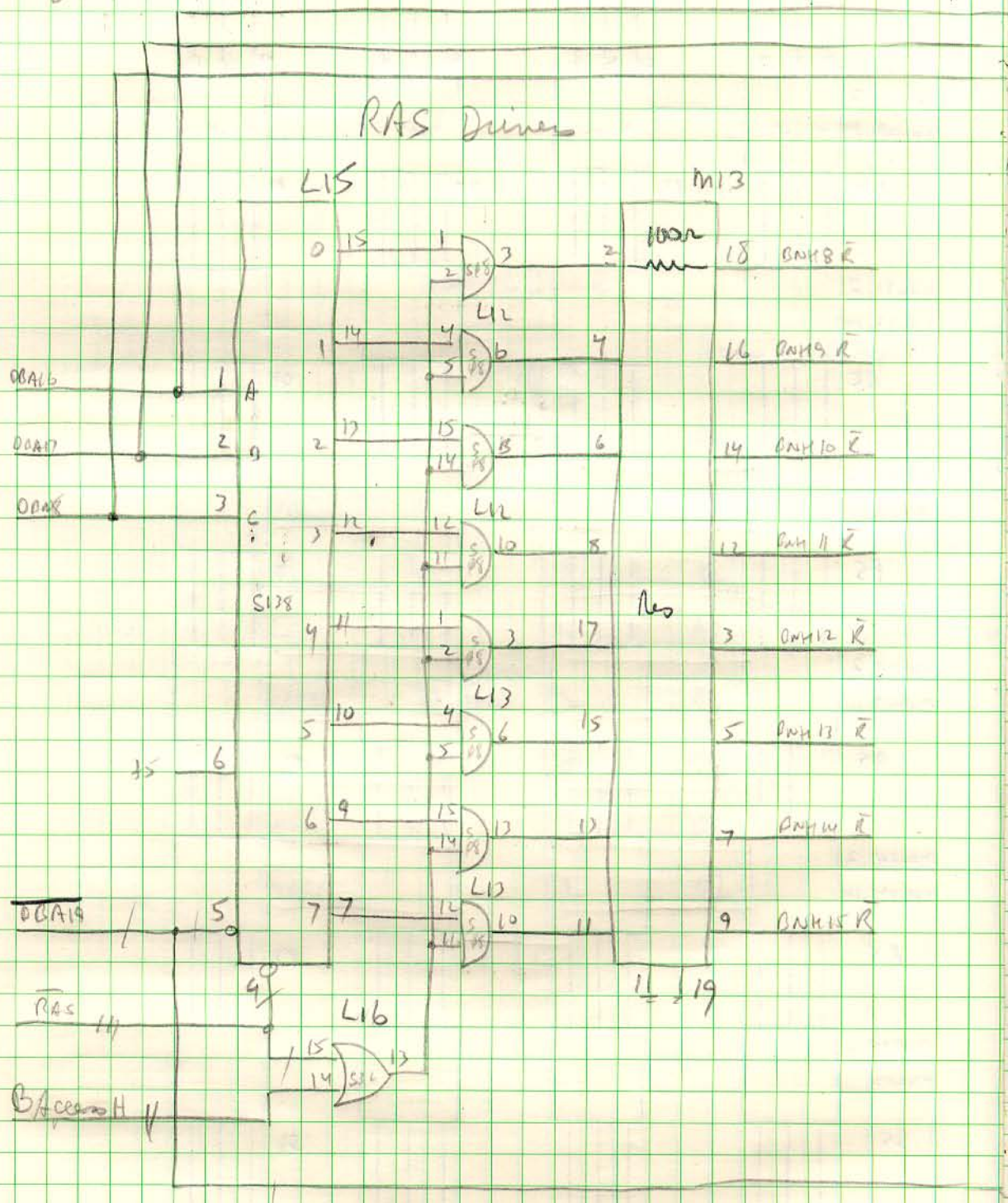
M7

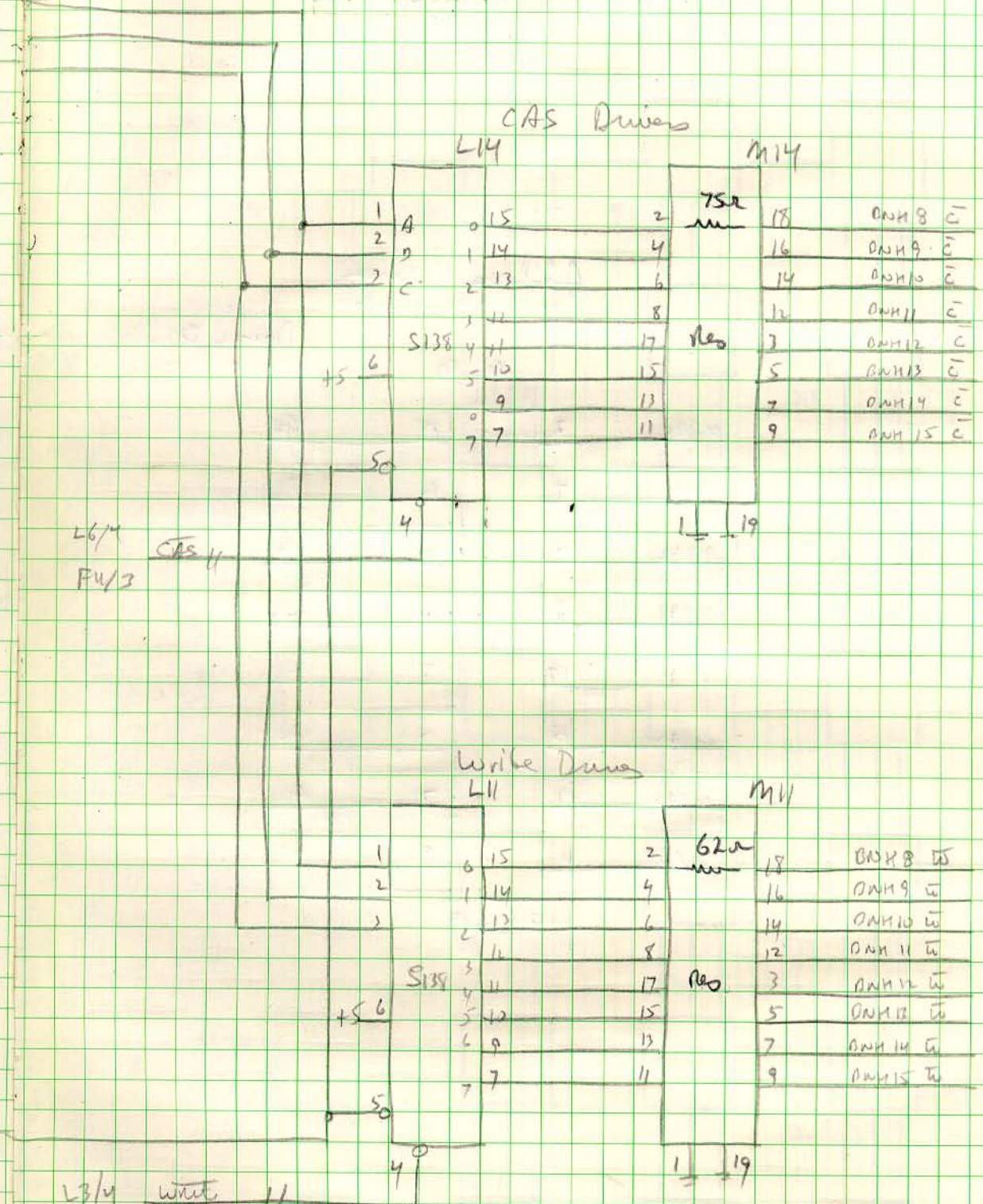



BNH4
BNH5
BNH6
BNH7
R C W
4164
MEC10 2
MEC00 14
N5
N6
N7
N8
MEC11 2
MEC01 14
O5
O8
MEC12 2
MEC02 14
P5
P5
MEC13 2
MEC03 14
Q5
Q8
MEC14 2
MEC04 14
R5
R8
MEC15 2
MEC05 14
S5
S8
MEC16 2
MEC06 14
T5
T8

6 Nov 82
BRO

EC Memory Control 2-4 mbyte



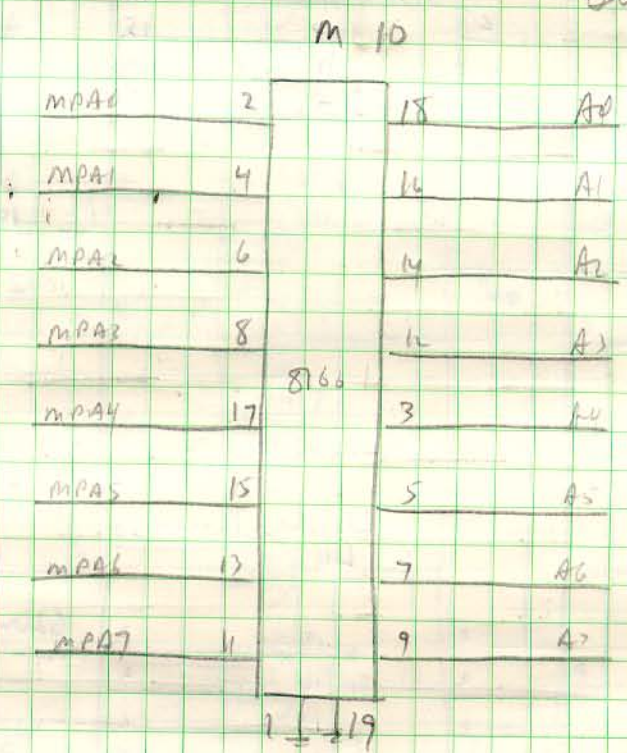

CAS Drive
L14
M14
75Ω
S138
+5
L6/4
CAS H
FH/3
Write Drive
L11
M11
62Ω
S138
+5
L3/4
WRITE H
GH/10

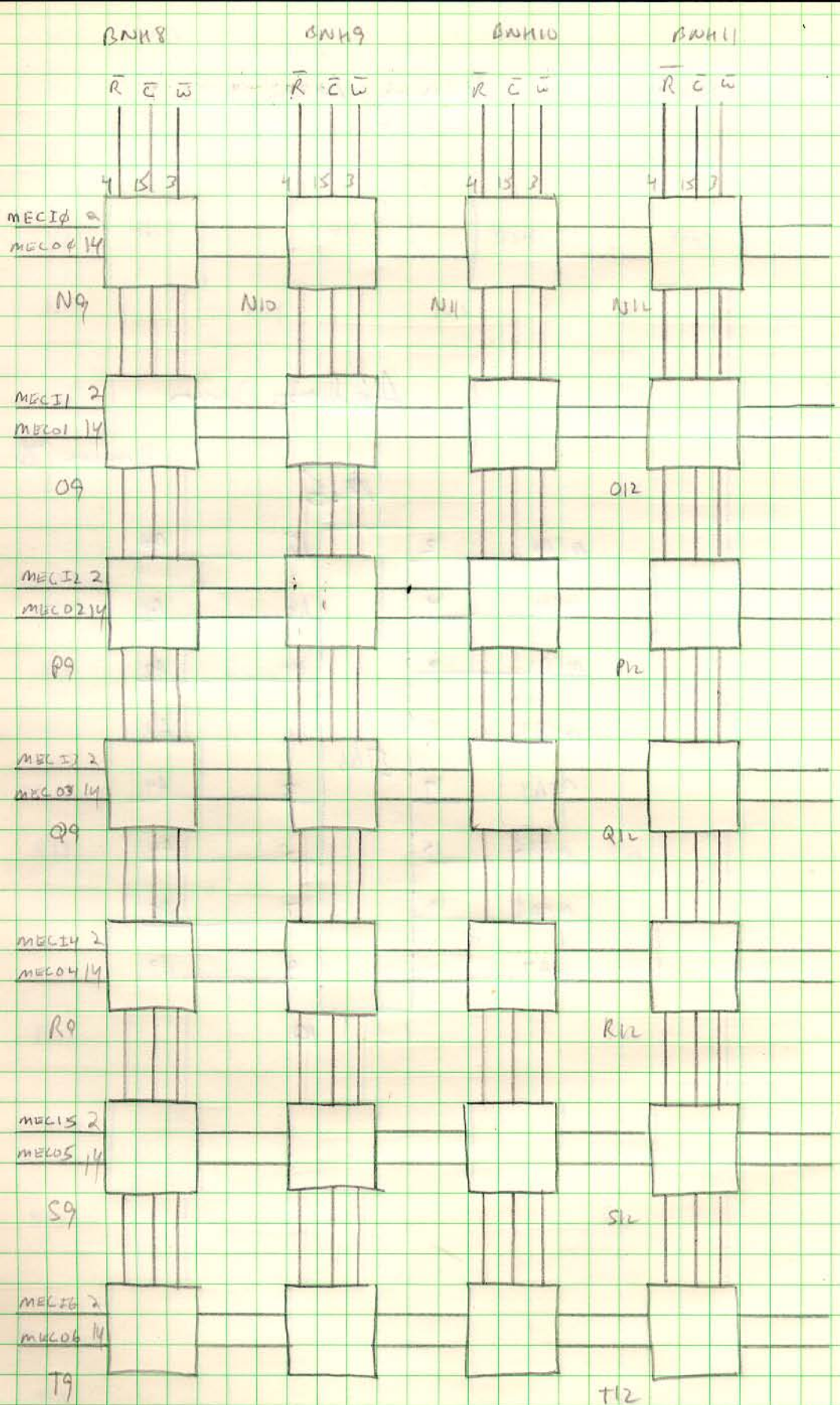
7 Nov 82
ABO

BNK 8 – 0BH 11 Memory Array EC

Address Drives

Drive to 28 chips




BNH8
BNH9
BNH10
BNH11
R̄ C̄ W̄
4 15 3
MECIØ 2
MECOØ 14
NØ
N10
N11
N1L
MECI1 2
MECO1 14
O9
O12
MECI2 2
MECO2 14
P9
P12
MECI3 2
MECO3 14
Q9
Q12
MECI4 2
MECO4 14
R9
R12
MECI5 2
MECO5 14
S9
S12
MECI6 2
MECO6 14
T9
T12

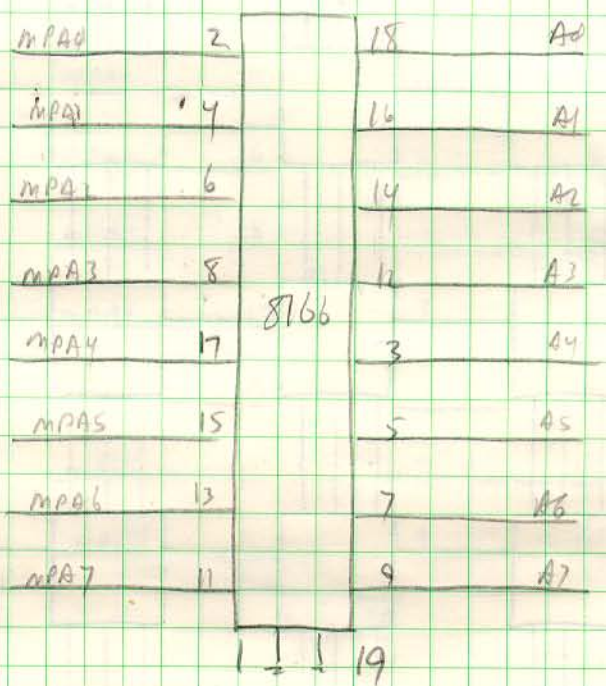
7 Nov 82
ARD

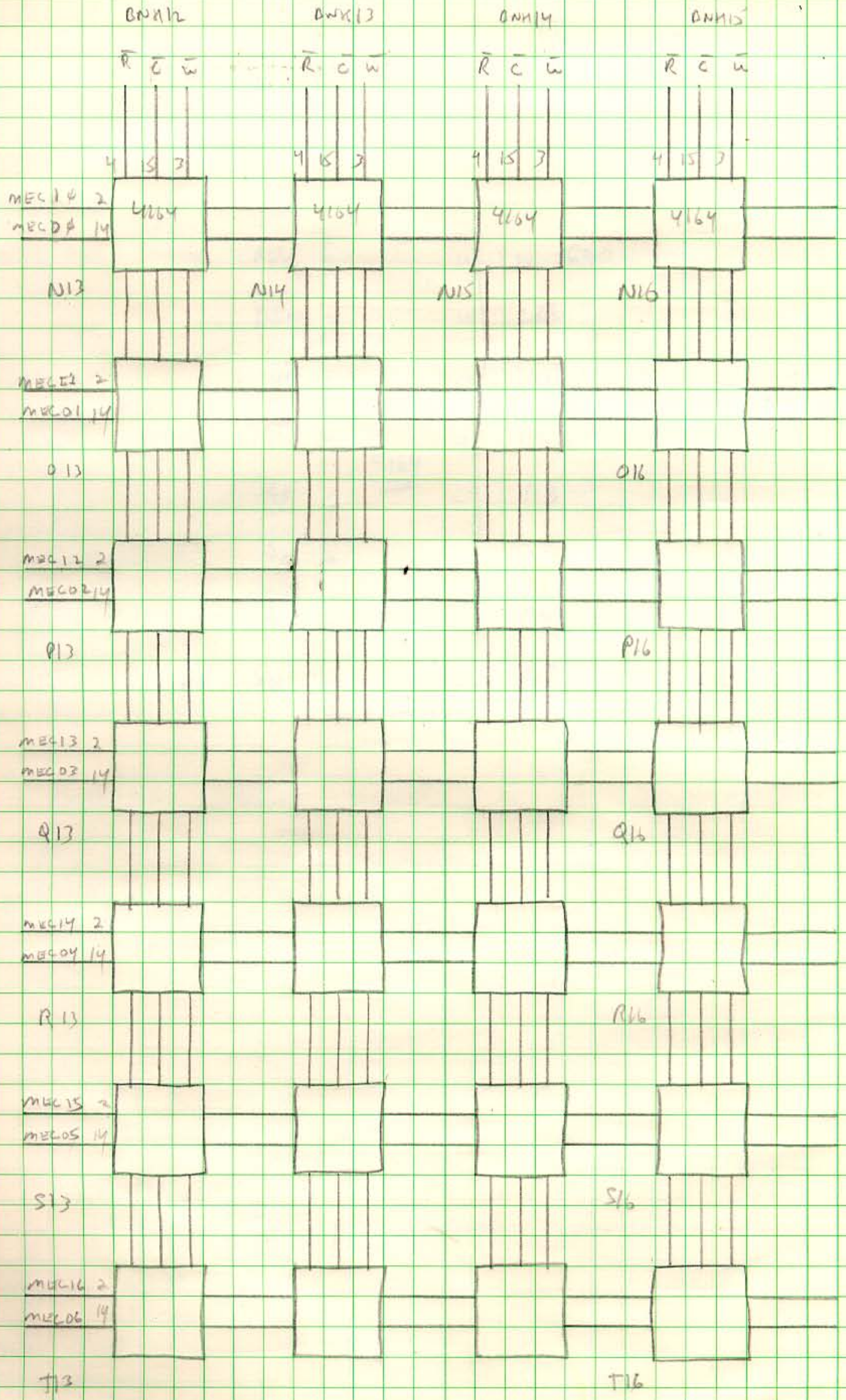
CPU 1/2 – Buffers : Memory Arb (4C)

Address Drivers

Driven by 28 chips

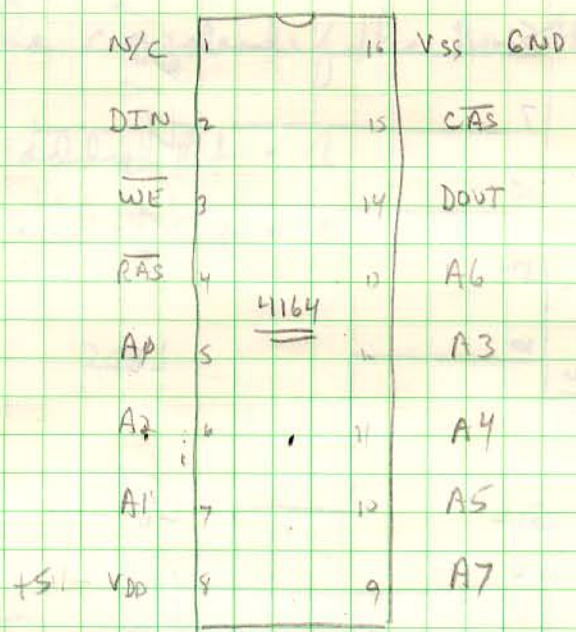
M15




DNH12
DNH13
DNH14
DNH15
$\bar{R}$ $\bar{C}$ $\bar{W}$
4 15 3
4164
MECIØ 2
MECDØ 14
N13
N14
N15
N16
MECI1 2
MECO1 14
O13
O16
MECI2 2
MECO2 14
P13
P16
MECI3 2
MECO3 14
Q13
Q16
MECI4 2
MECO4 14
R13
R16
MECI5 2
MECO5 14
S13
S16
MECI6 2
MECO6 14
T13
T16

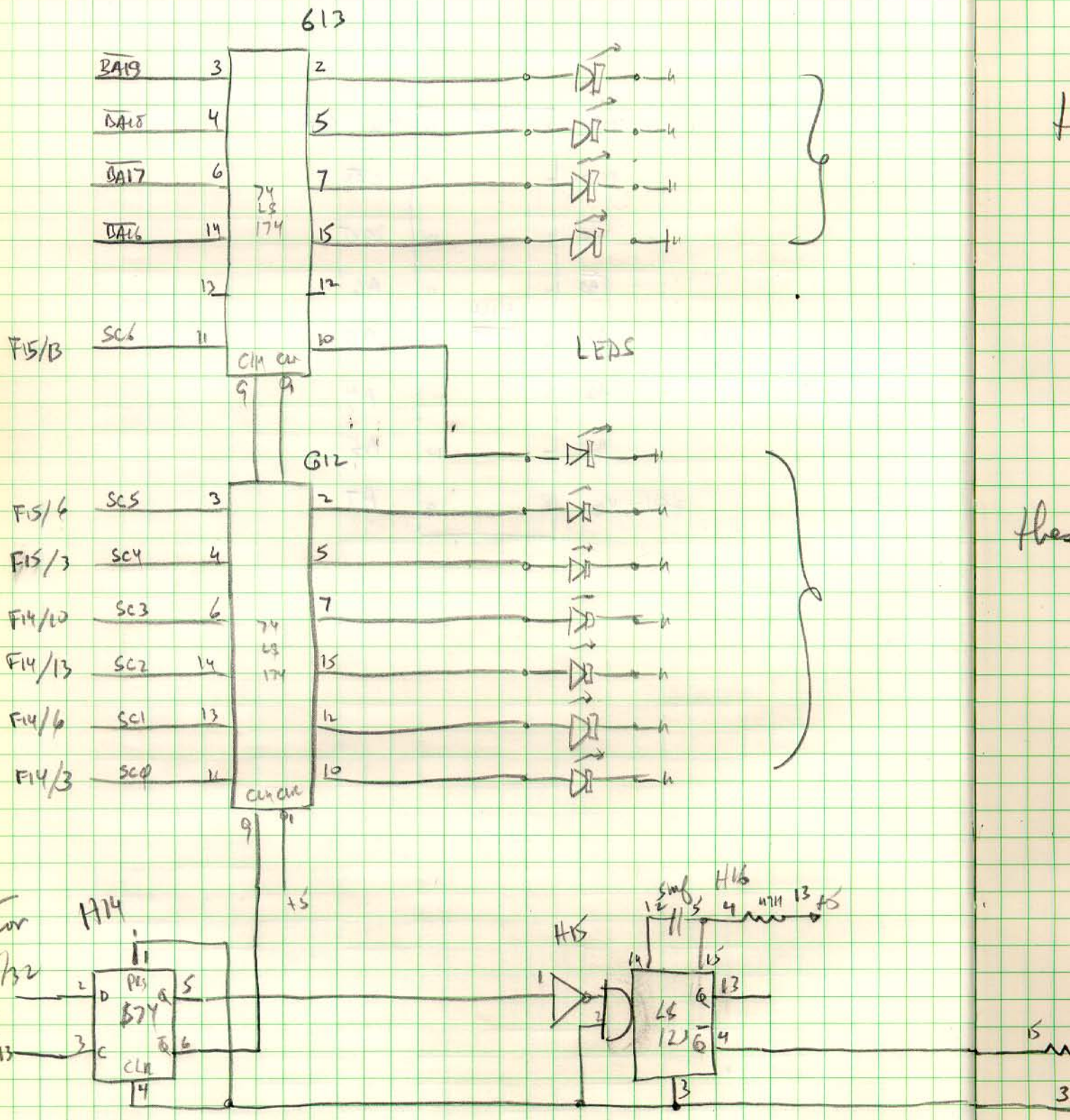
7 Mar 82
ORS

64K Dynamic Pin Out



7 Nov 82

Syndrome / Address Monitor



these may be used to find the
Memory Bank and Bit # for correction Action.

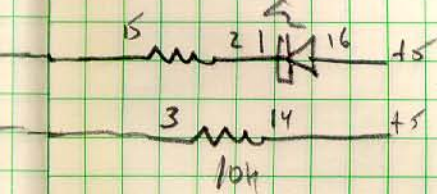
these are the complends of the Bus Address

Off means address bit = 1
On means address bit = 0

these indicate the Syndrome Code for the Error

See the Table (# VII) in the AmD 2960
Data Book

H16 Pulsed when an Error Detected



3 Feb 93
ABC

IC Location Index

A1	7				
A2	7				
A3	7				
A4	7				
A5	8	C5	7		E5
A6	8	C6	6		E6
A7	8	C7	6		E7
A8	8	C8	6		E8
					E9
					E10
					E11
A13	14	C13	15		E13
A14	14	C14	15		
A15		C15	15		E15
A16		C16	15		
B1	7				
B2	7				
B3	7				
B4	7				
B5	8	D5	7		F5
B6	8	D6	6		F6
B7	8	D7	6		F7
B8	8	D8	6		F8
		D9	14		F9
		D10	14		F10
		D11	14		F11
					F12
B13	14				F13
B14	14				F14
B15	14				F15
B16	14				F16

E5	7	G6	9
E6	6	G7	10
E7	6	G8	10
E8	6	G9	11
E9	7	G10	12
E10	15	G11	12
E11			
E13	16		
E14			
E15	16		
F5		H6	9
F6	6	H7	10
F7	7.9	H8	9, 10
F8	6	H9	11
F9	11	H10	12
F10	12	H11	12
F11	12		
F12			
F13			
F14	16		
F15	16		
F16			

8 Nov 82
ARD

FC Location Index

I1	—	K1	—
I2	—	K2	—
I3	—	K3	—
I4	—	K4	—
I5	9	K5	—
I6	9	K6	—
I7	9	K7	—
I8	9	K8	—
I9	11	K9	11
I10	11	K10	11
I11	12	K11	—
I12	—	K12	—
I13	—	K13	—
I14	—	K14	—
I15	—	K15	—
I16	—	K16	—
J1	—	L1	—
J2	—	L2	—
J3	—	L3	18
J4	—	L4	17
J5	9	L5	17
J6	9	L6	18
J7	—	L7	17
J8	—	L8	17
J9	11	L9	17
J10	11	L10	17
J11	—	L11	21
J12	—	L12	21
J13	—	L13	21
J14	—	L14	21
J15	—	L15	21
J16	—	L16	18, 21

M1
M2
M3
—
M5
M6
M7
M8
M9
M10
M11
—
M13
M14
M15
M16

N1
N2
N3
N4
N5
N6
N7
N8
N9
N10
N11
N12
N13
N14
N15
N16

M1	49
M2	
M3	18
M4	
M5	18
M6	18
M7	
M8	20
M9	22
M10	
M11	21
M12	
M13	21
M14	21
M15	16
M16	23

O1	19
O2	19
O3	19
O4	19
O5	20
O6	20
O7	20
O8	20
O9	22
O10	22
O11	22
O12	22
O13	23
O14	23
O15	23
O16	23

N1	19
N2	19
N3	19
N4	19
N5	20
N6	20
N7	20
N8	20
N9	22
N10	22
N11	22
N12	22
N13	23
N14	23
N15	23
N16	23

P1	19
P2	19
P3	19
P4	19
P5	20
P6	20
P7	20
P8	20
P9	22
P10	22
P11	22
P12	22
P13	23
P14	23
P15	23
P16	23

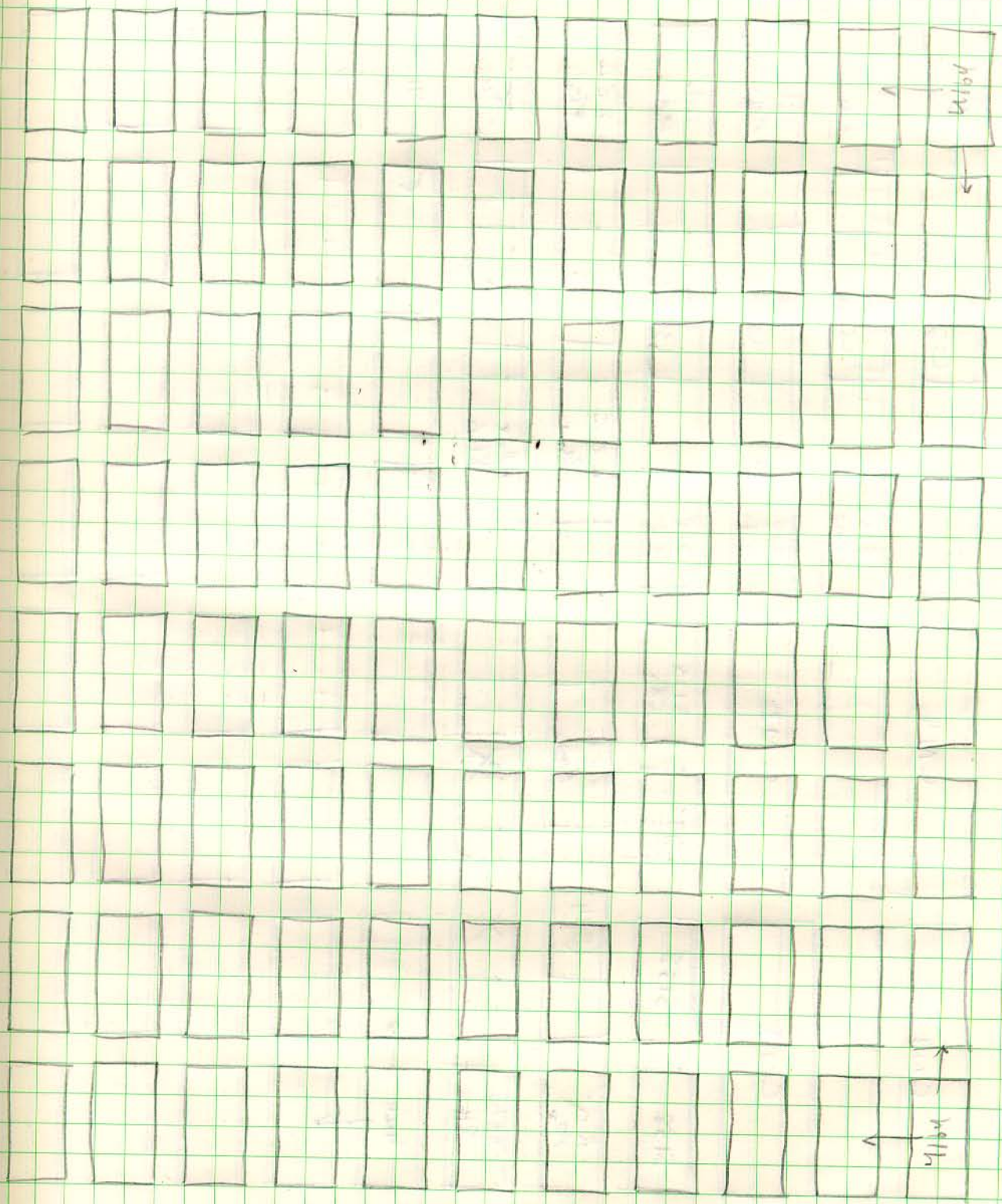
IC Location Index

Q1	19	S1	19
Q2	19	S2	19
Q3	19	S3	19
Q4	19	S4	19
Q5	20	S5	20
Q6	20	S6	20
Q7	20	S7	20
Q8	20	S8	20
Q9	22	S9	22
Q10	22	S10	22
Q11	22	S11	22
Q12	22	S12	22
Q13	23	S13	23
Q14	23	S14	23
Q15	23	S15	23
Q16	23	S16	23

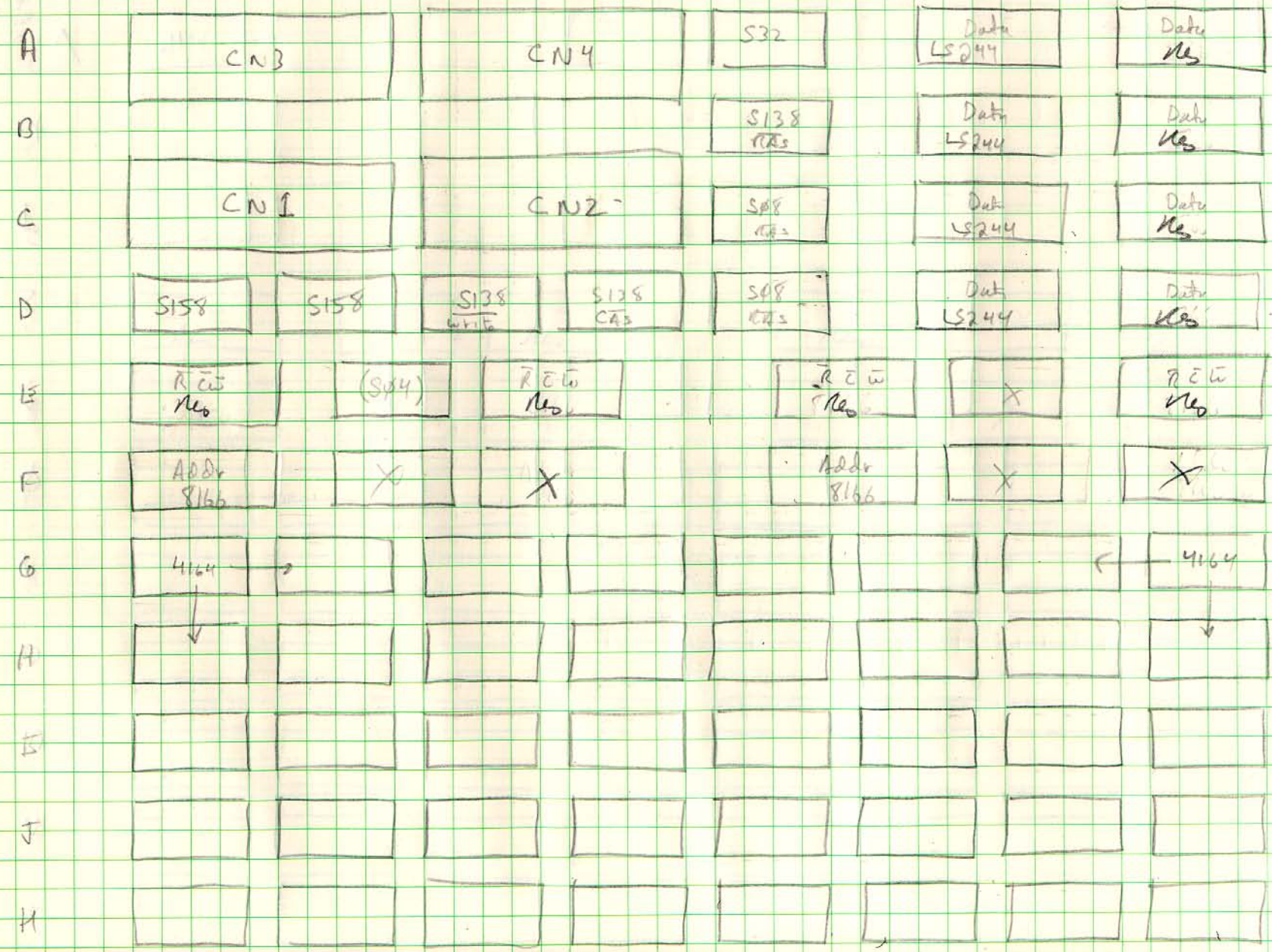
R1	19	T1	19
R2	19	T2	19
R3	19	T3	19
R4	19	T4	19
R5	20	T5	20
R6	20	T6	20
R7	20	T7	20
R8	20	T8	20
R9	22	T9	22
R10	22	T10	22
R11	22	T11	22
R12	22	T12	22
R13	23	T13	23
R14	23	T14	23
R15	23	T15	23
R16	23	T16	23

2 megabyte Memory Boards


REG
REG
REG
REG
ADDR
8166
ADDR
8166
4164
4164



9 Nov 82
ARD


A
B
C
D
E
F
G
H
J
H
L
CN3
CN4
CN1
CN2
S32
S138
S08
S158
S158
S138
S138
S08
S08
Data
LS244
Data
LS244
Data
LS244
Data
LS244
Data
Addr
8166
Addr
8166
4164
4164

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

AB

Memory Address/Control

CN1 & CN3

1	$\overline{BA0}$	—	D9/3	40	GND		
2		—	D9/6	39	$\overline{BA16}$	—	E10/1
3		—	D9/13	38	|	—	E10/3
4		—	D9/10	37		—	E10/5
5	GND			36	$\overline{DA19}$	—	E10/15
6		—	D10/3	35	GND		
7		—	D10/6	34	$\overline{RAS}$	—	B13/4
8		—	D10/13	33	GND		
9		—	D10/10	32	$\overline{CAS}$	—	D12/4
10	GND			31	GND		
11		—	D9/2	30	GND		
12		—	D9/5	29	$\overline{CoLADL}$	—	D9/1
13		—	D9/14	28	GND		
14		—	D9/11	27	$\overline{X\ Access\ H}$	—	A13/15
15	GND			26	GND		
16		—	D10/2	25	GND		
17		—	D10/5	24	$\overline{Read}$	—	A13/11
18		—	D9/14	23	GND		
19	$\overline{BA15}$	—	D9/11	22	$\overline{Write}$	—	D11/4
20	GND			21	GND		

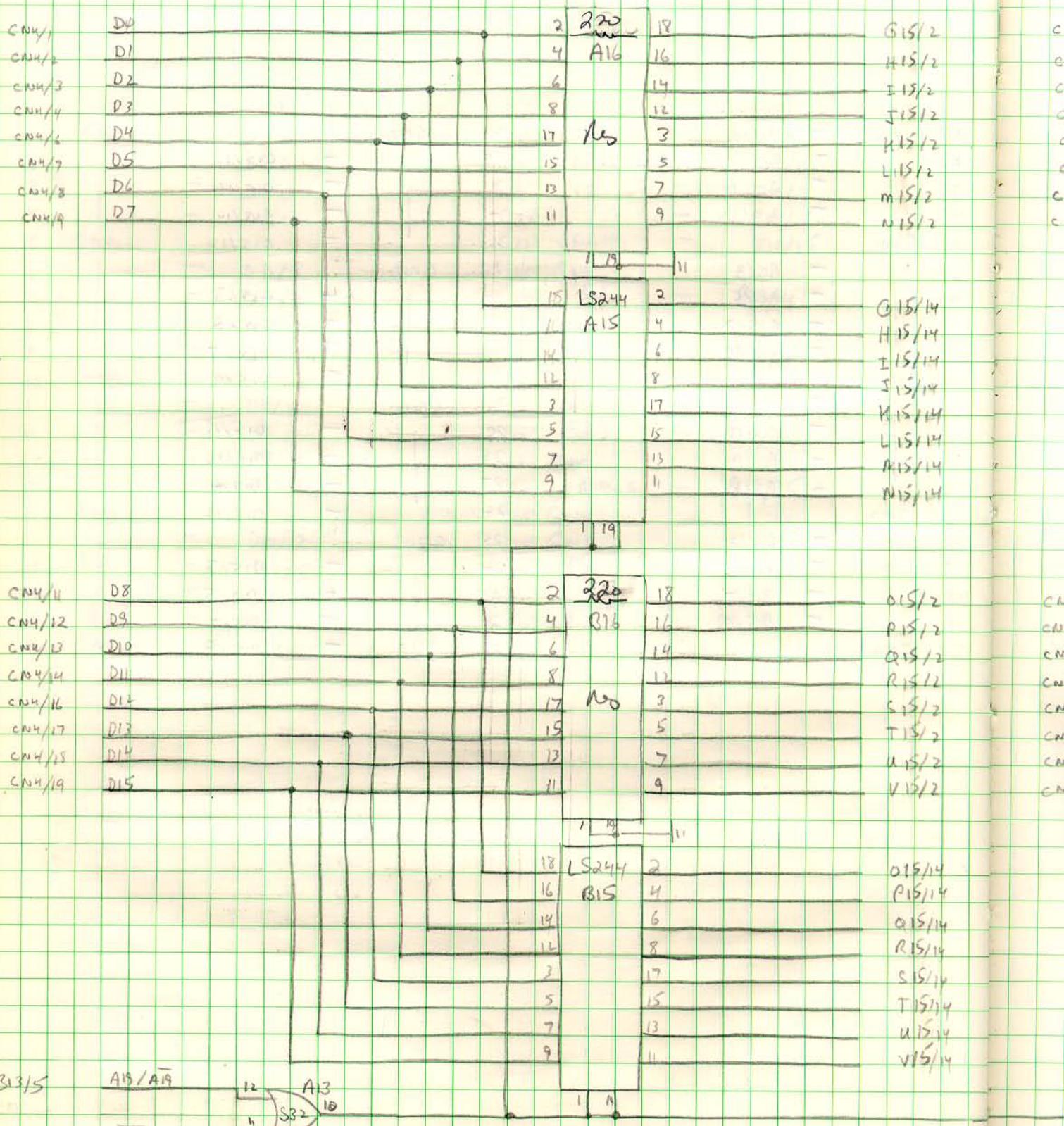
Memory Data Bus

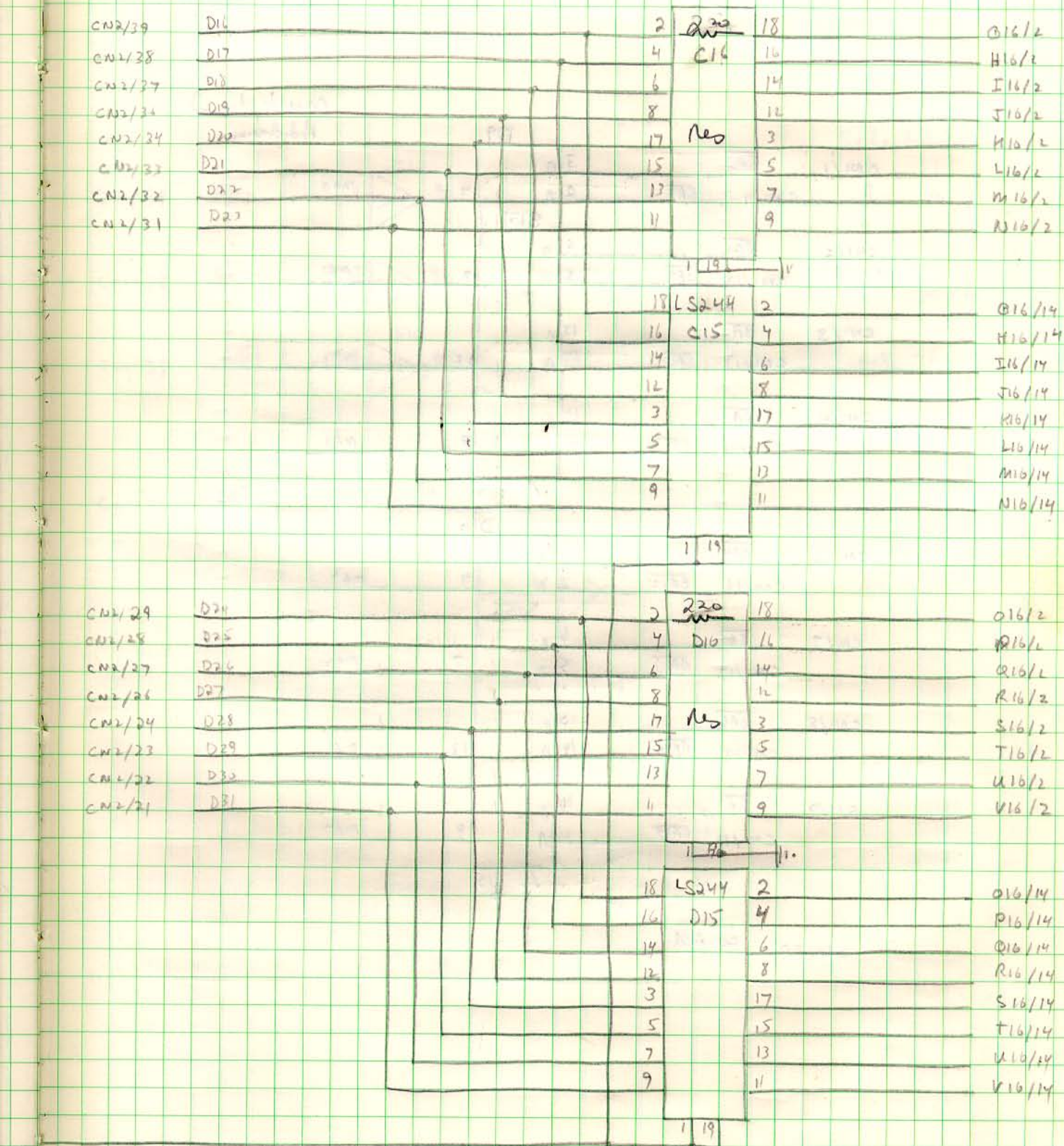
CN2 & CN4

1	D0	—	A15/18	40	GND		
2	|	—	A15/16	39	D16	—	C15/18
3	|	—	A15/14	38	|	—	C15/16
4	|	—	A15/12	37	|	—	C15/14
5	GND			36	|	—	C15/12
6	|	—	A15/3	35	GND		
7	|	—	A15/5	34	|	—	C15/3
8	|	—	A15/7	33	|	—	C15/5
9	|	—	A15/9	32	|	—	C15/7
10	GND			31	|	—	C15/9
11	|	—	B15/18	30	GND		
12	|	—	B15/16	29	|	—	D15/18
13	|	—	B15/14	28	|	—	D15/16
14	|	—	B15/12	27	|	—	D15/14
15	GND			26	|	—	D15/12
16	|	—	D15/3	25	GND		
17	|	—	B15/5	24	|	—	D15/3
18	|	—	B15/7	23	|	—	D15/5
19	D15	—	B15/9	22	|	—	D15/7
20	GND			21	D31	—	D15/9

11 Nov 82
ABO

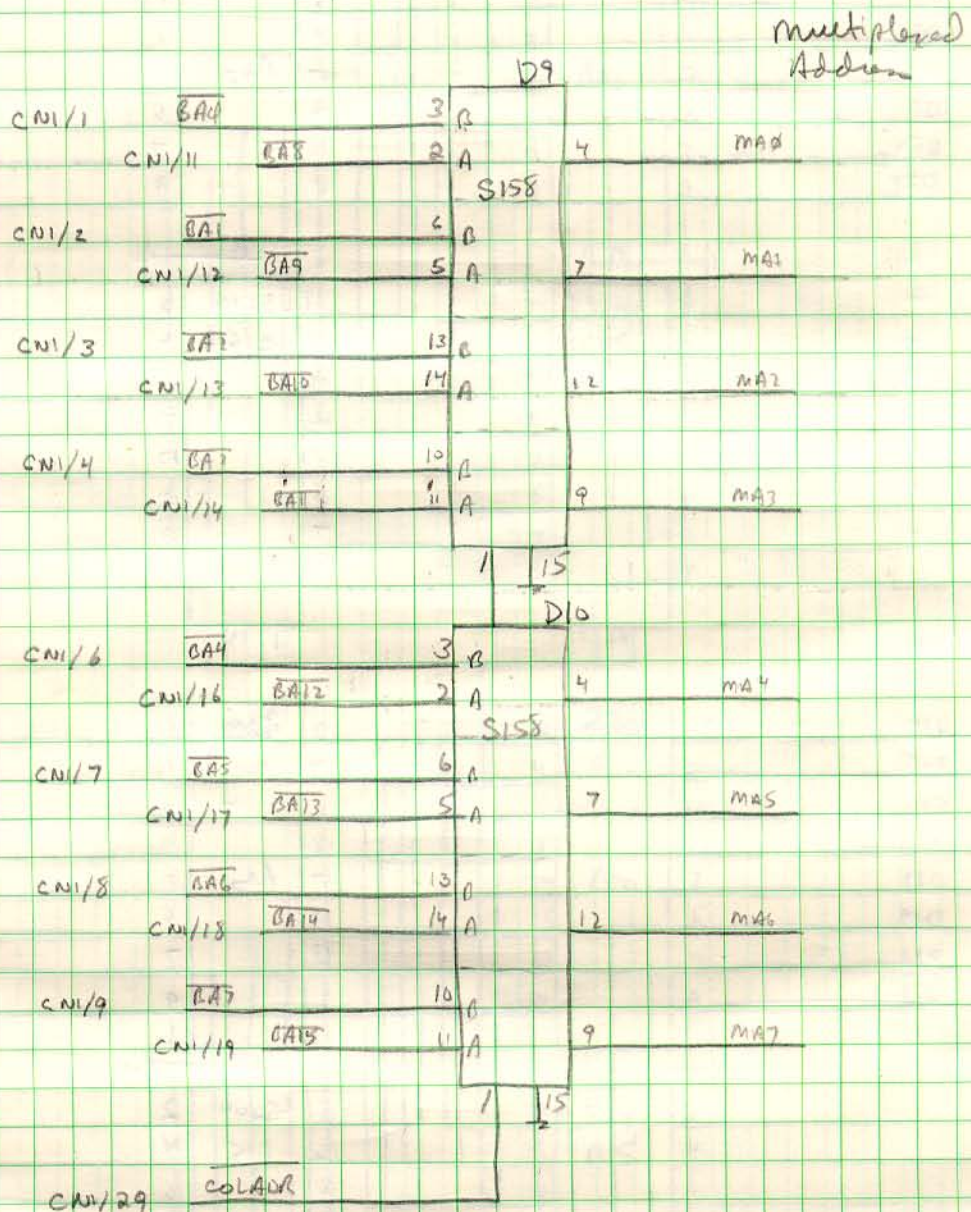
Data Bus Buffers



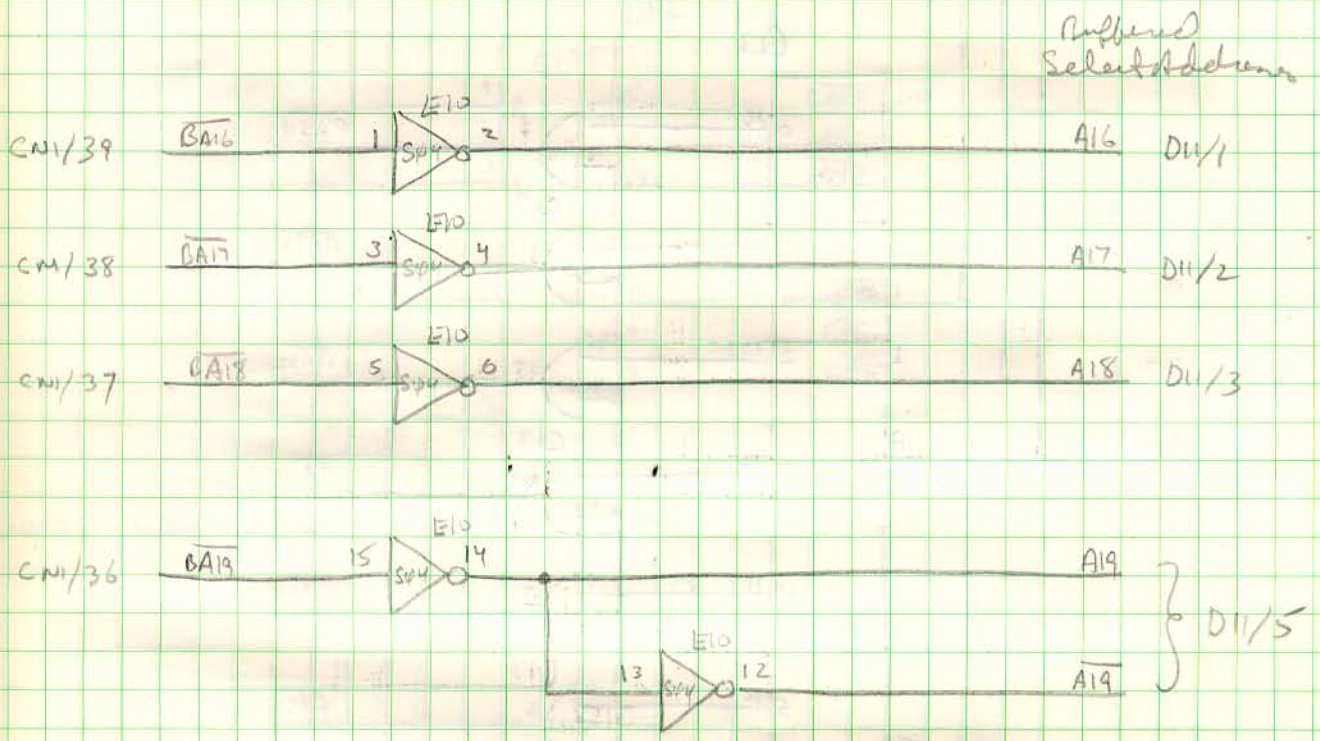

CN2/39
CN2/38
CN2/37
CN2/36
CN2/34
CN2/33
CN2/32
CN2/31
D16
D17
D18
D19
D20
D21
D22
D23
220
C16
Res
LS244
C15
CN2/29
CN2/28
CN2/27
CN2/26
CN2/24
CN2/23
CN2/22
CN2/21
D24
D25
D26
D27
D28
D29
D30
D31
220
D16
Res
LS244
D15

31 Dec 82
ABA

Address multiplexer

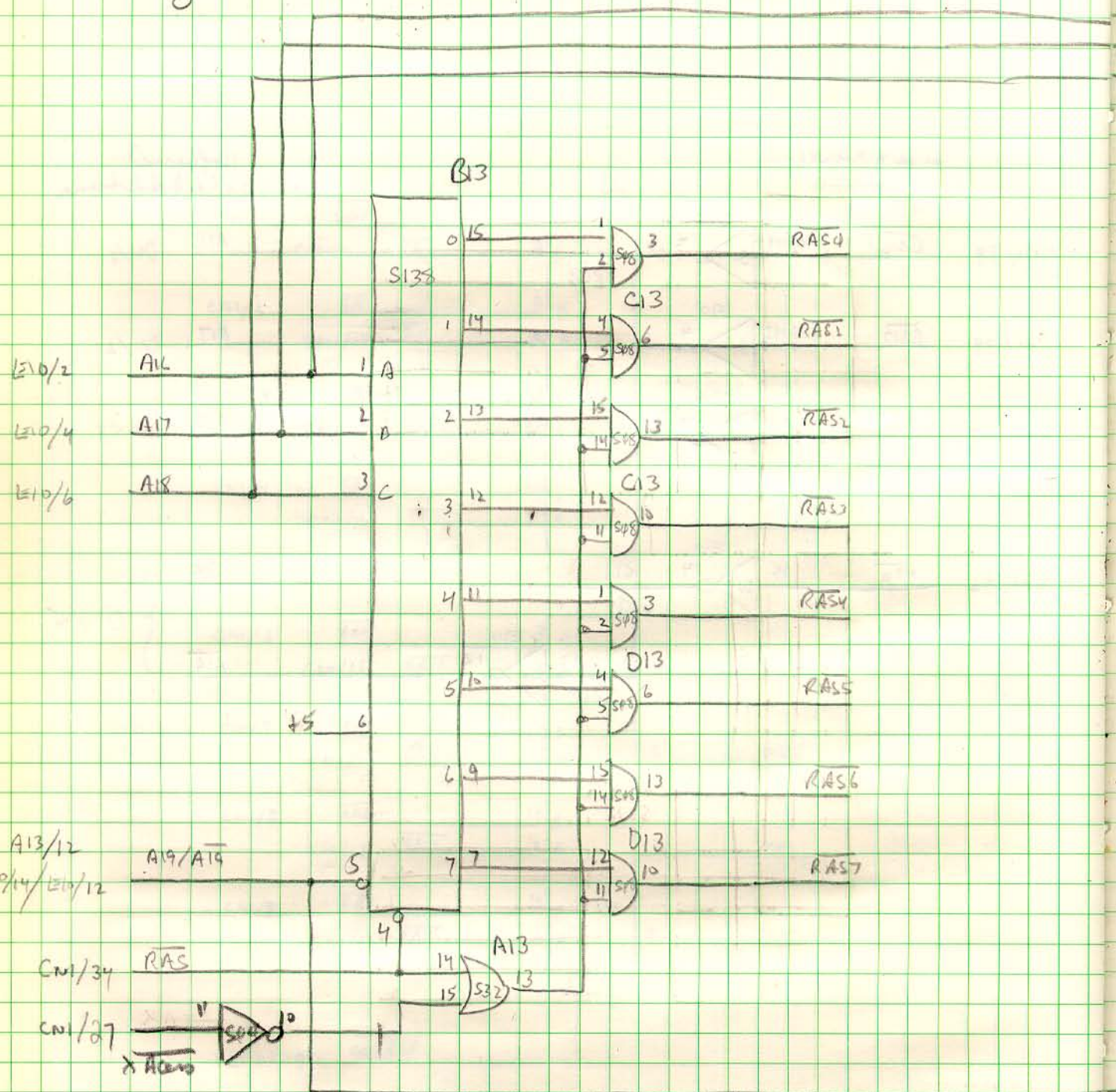


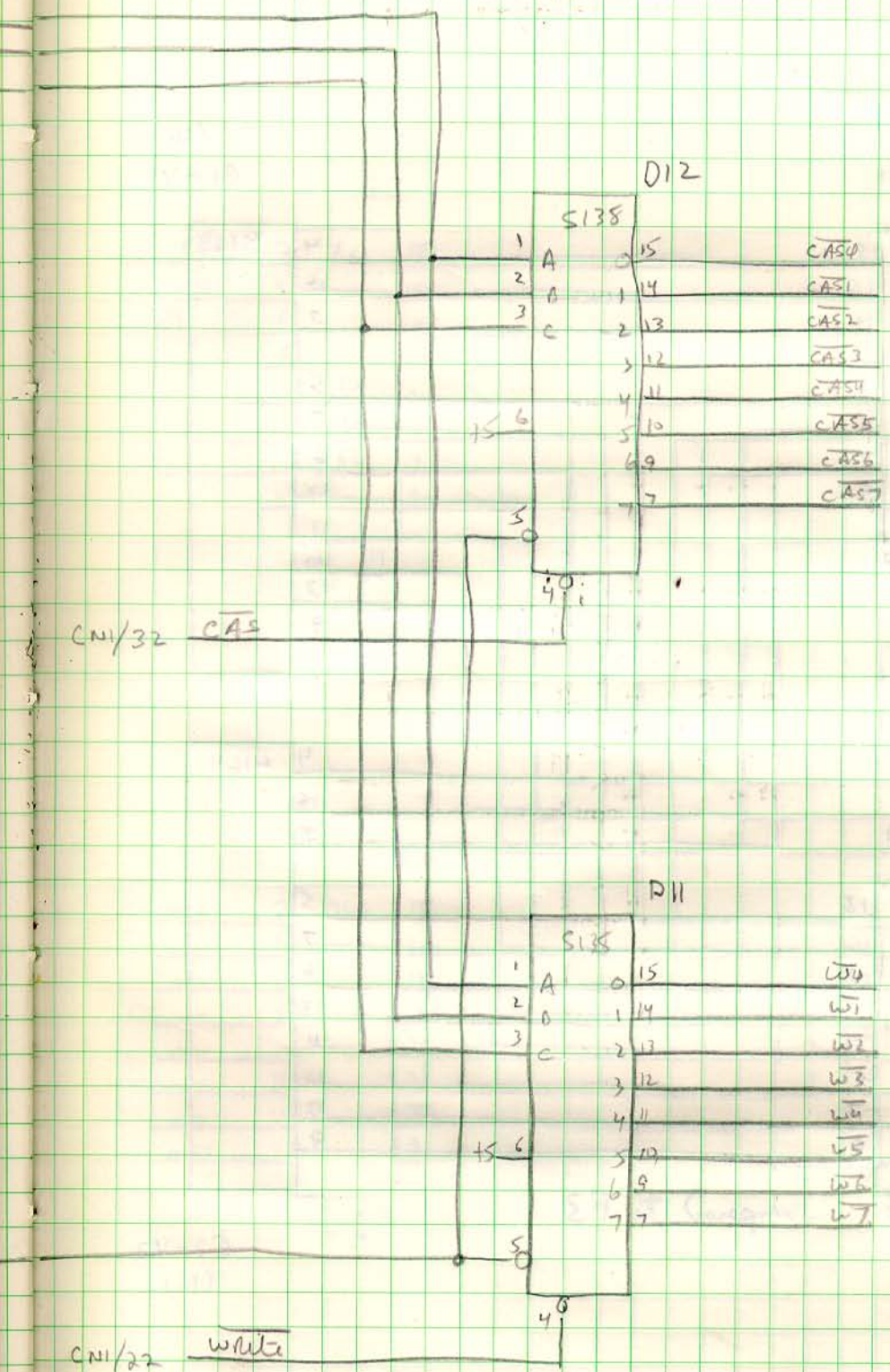
Address Buffers



1 Jan 83
ARB

Memory Control

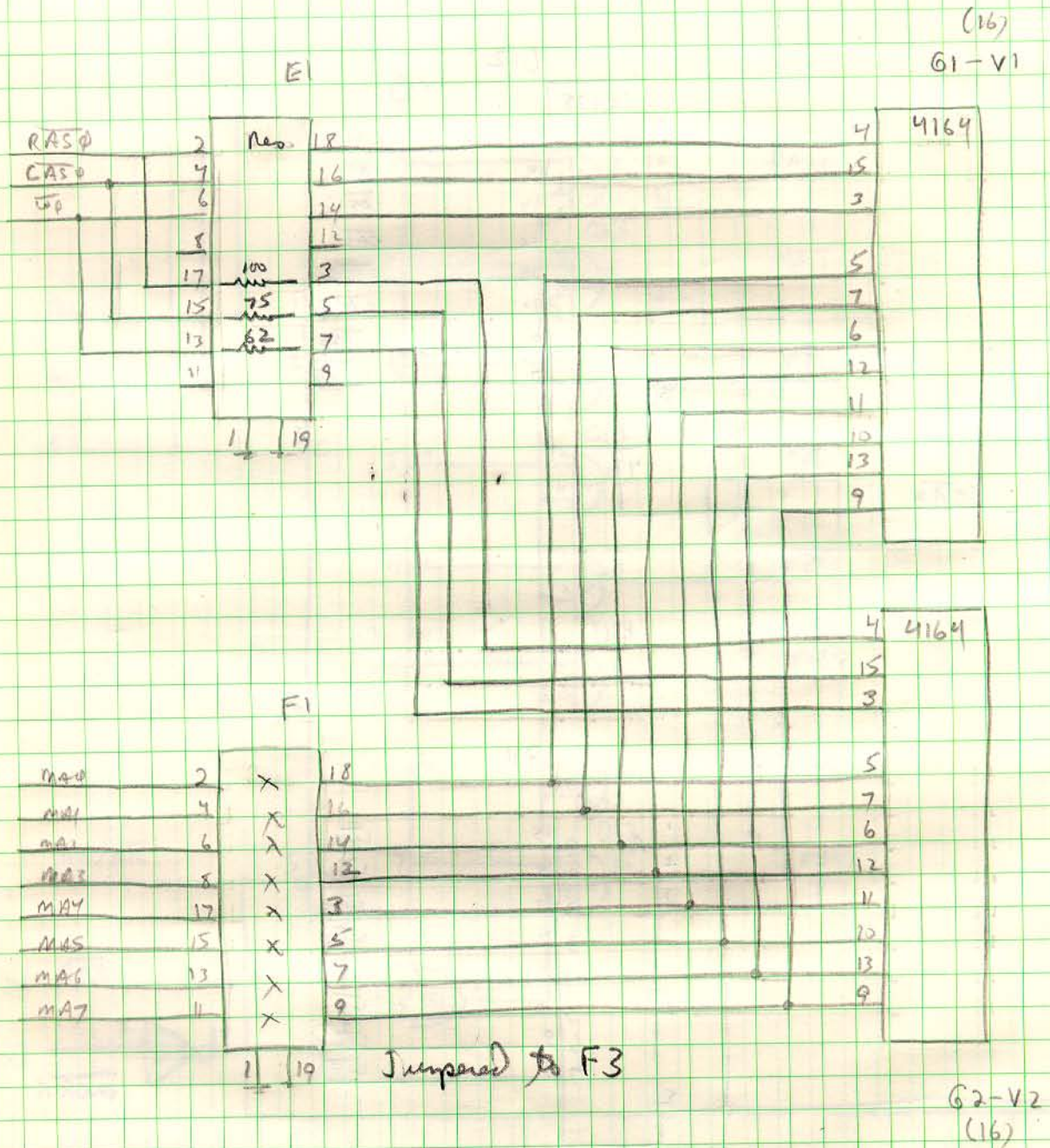



D12
S138
A
B
C
G1
G2A
G2B
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
CAS0
CAS1
CAS2
CAS3
CAS4
CAS5
CAS6
CAS7
CN1/32
CAS
D11
S138
WR0
WR1
WR2
WR3
WR4
WR5
WR6
WR7
CN1/22
write

1 Jan 82
ARO

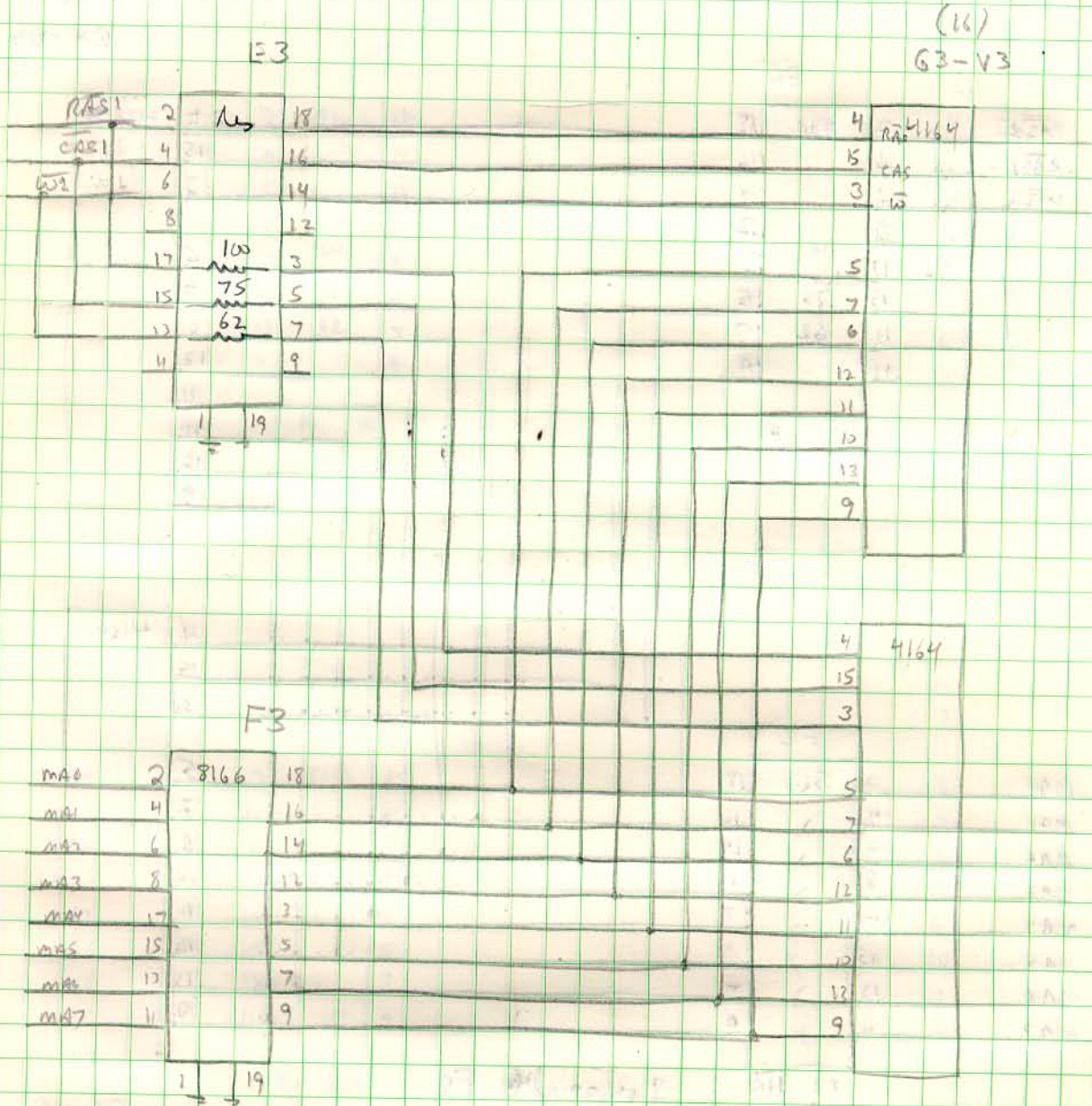
BNHP

Memory & Drivers



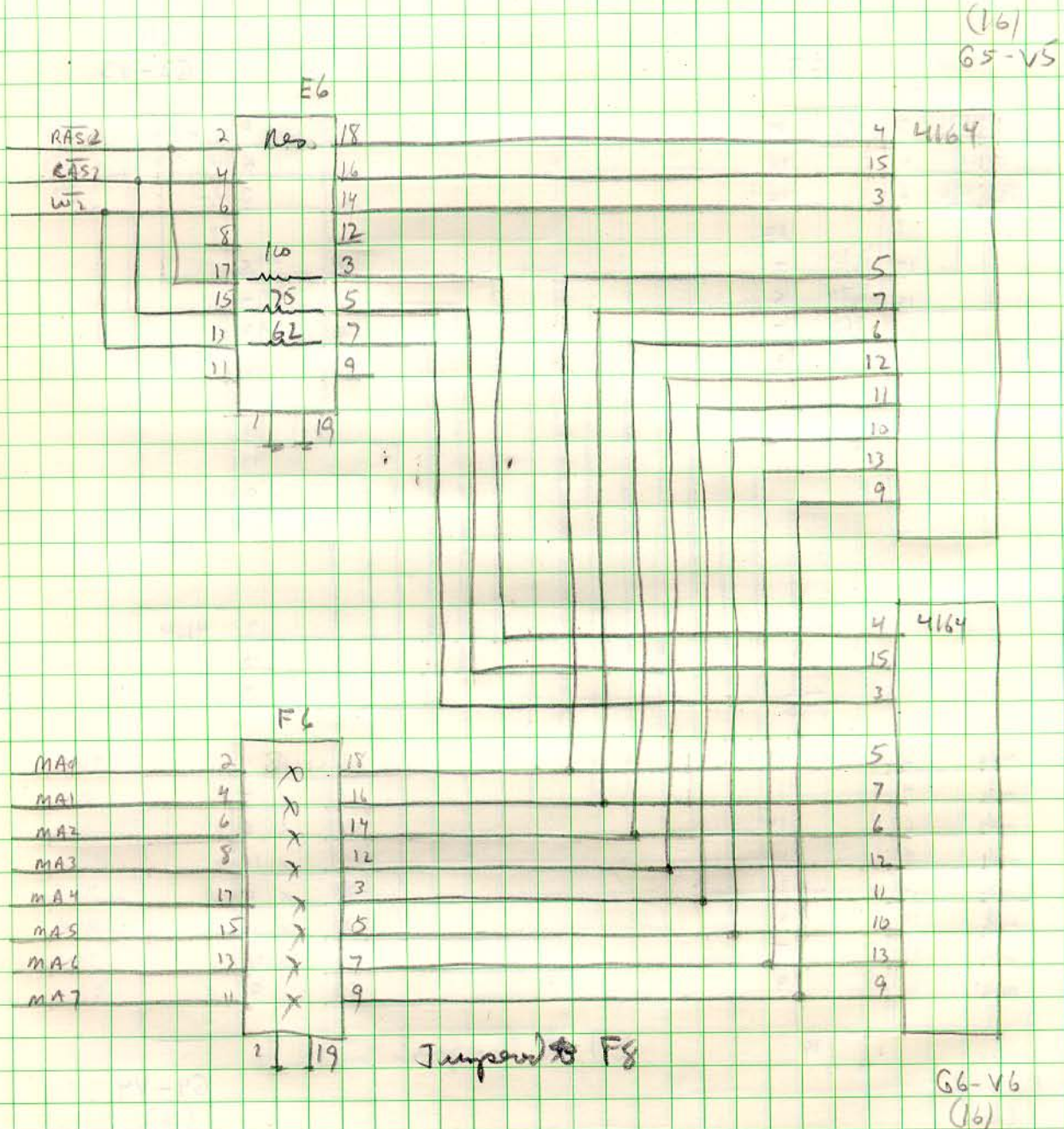
BNK 1

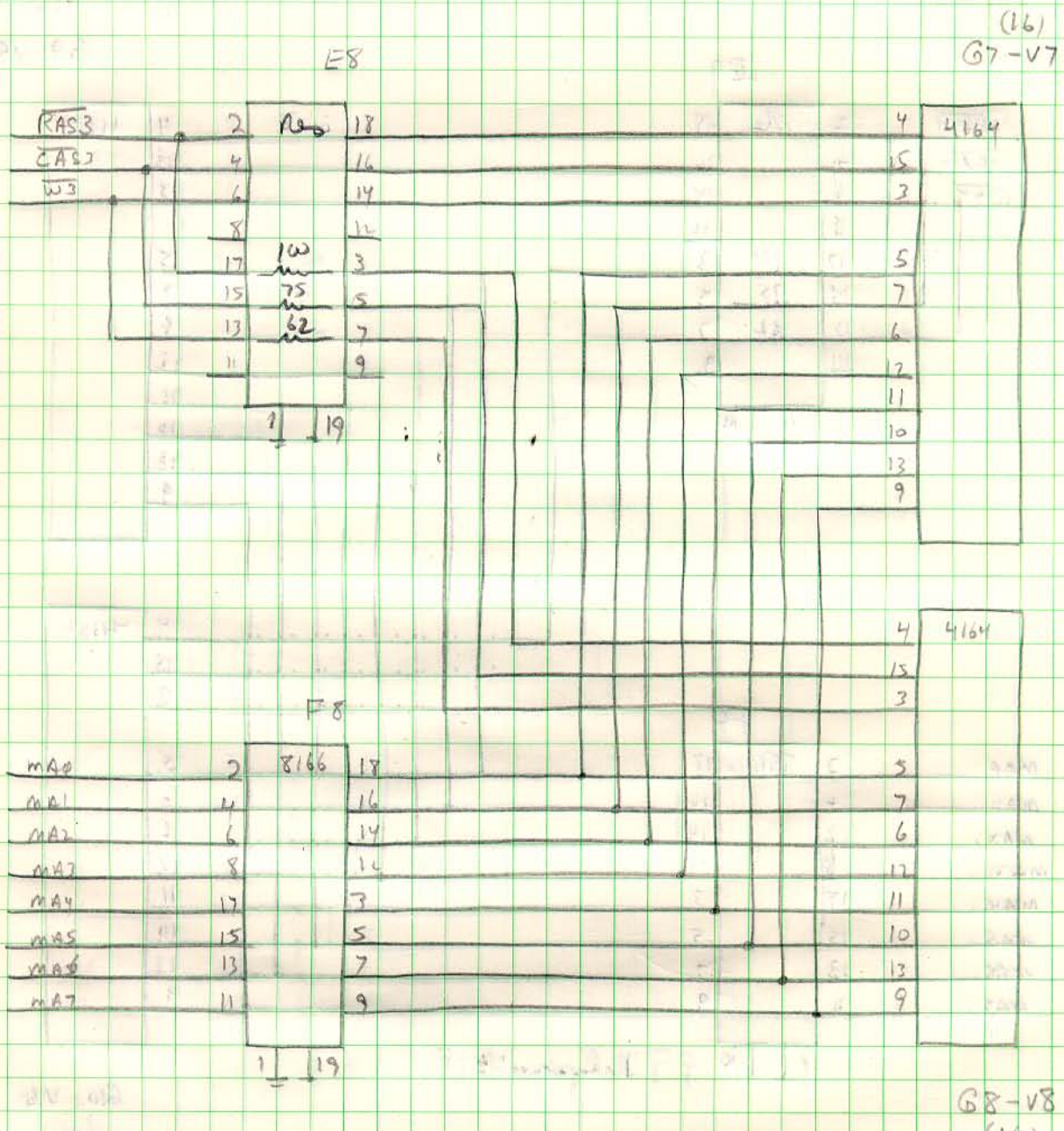
35



3 Jan 83
AAB

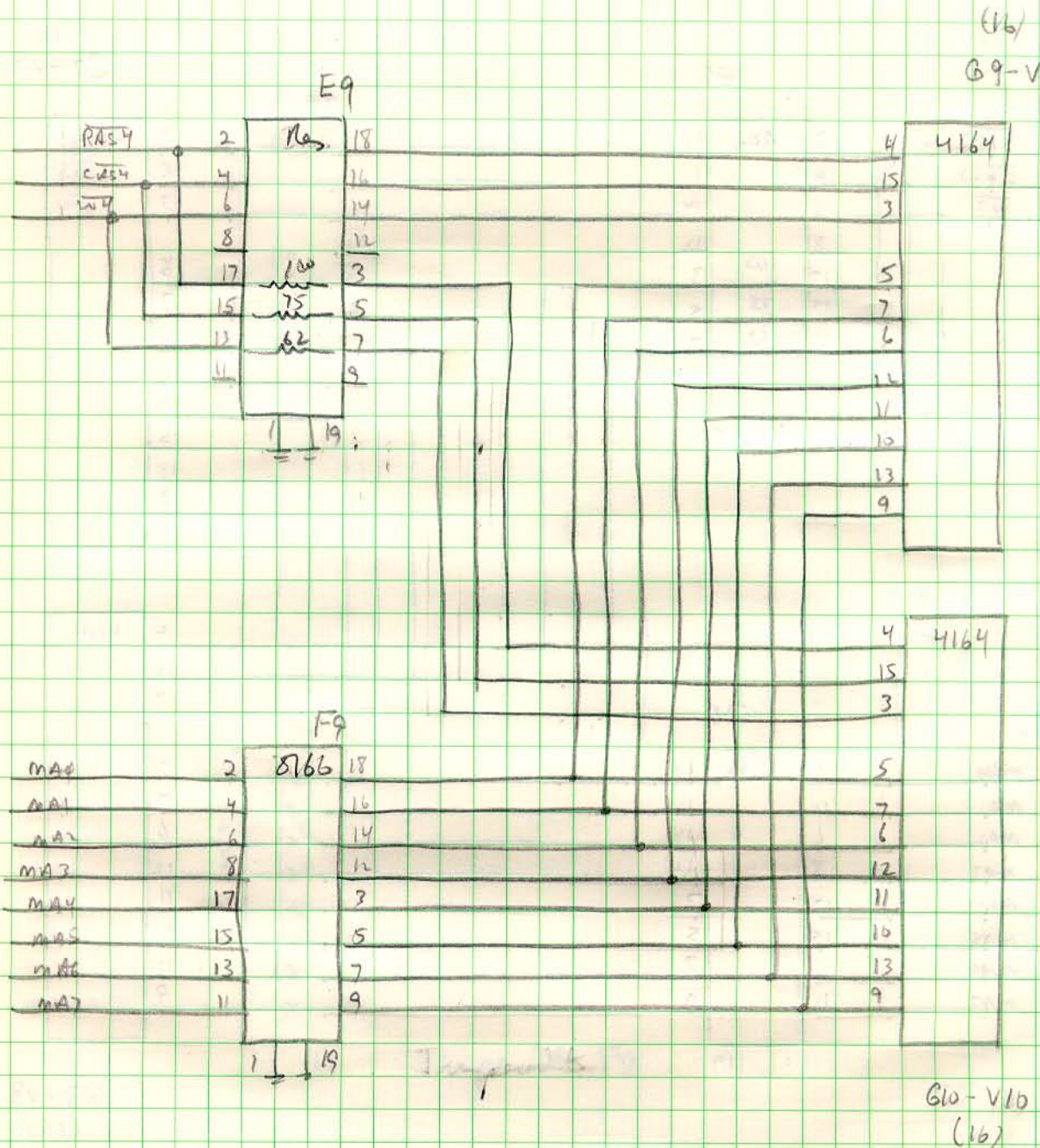
BNH2


(16)
G5-V5
E6
RAS2
CAS2
WE2
2
4
6
8
17
15
13
11
18
16
14
12
3
5
7
9
1
19
100
75
62
4164
4
15
3
5
7
6
12
11
10
13
9
F6
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7
Jumpers F8
G6-V6
(16)


E8
(16)
G7 - V7
RAS3
CAS3
W3
2
4
6
8
17
15
13
11
18
16
14
12
3
5
7
9
1
19
100
75
62
4164
4
15
3
5
7
6
2
11
10
13
9
F8
8166
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7
G8 - V8
(16)

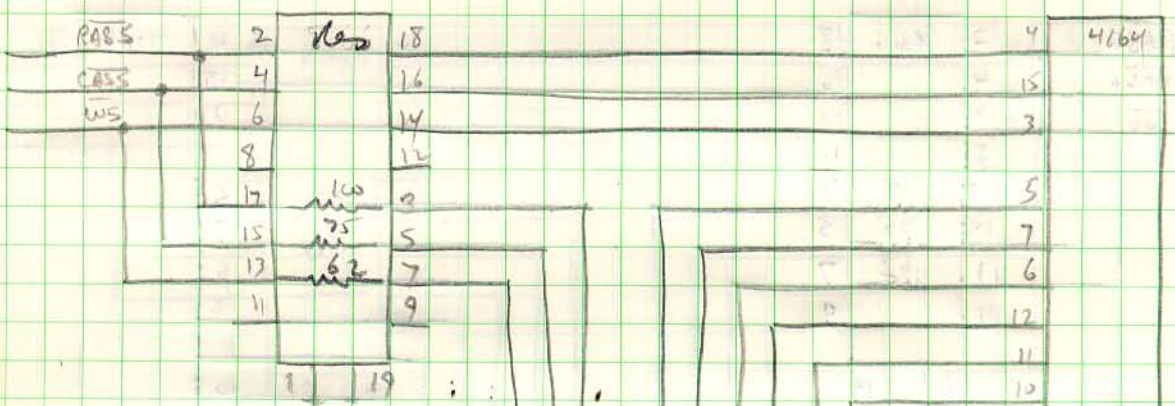
3 Jan 83
ARD

BNK4

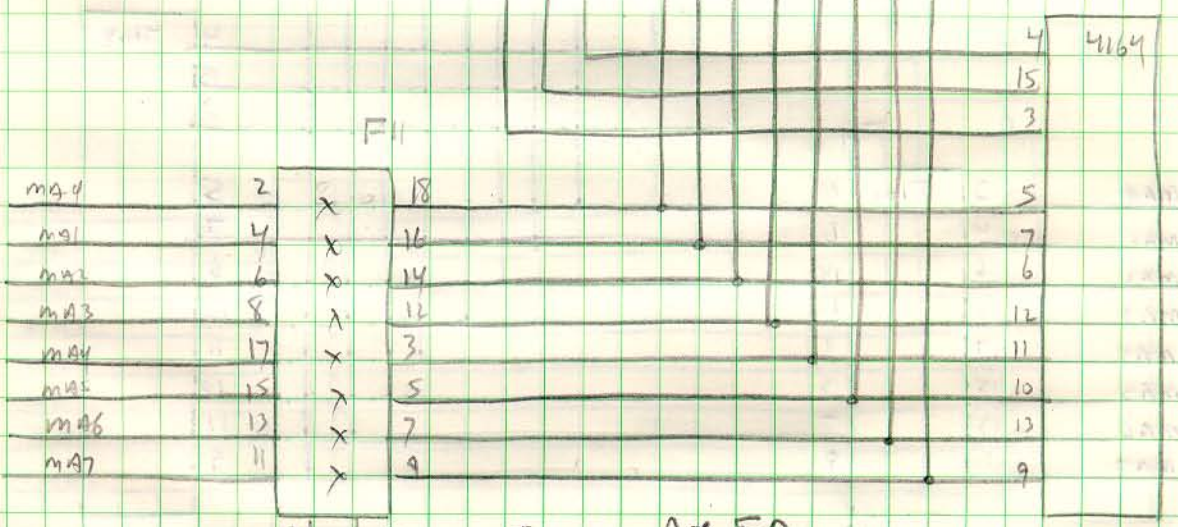

E9
F9
RAS4
CAS4
WE4
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7
8766
4164
4164
(16)
G9-V9
G10-V10
(16)

(16)
GU-VU

E11



F11

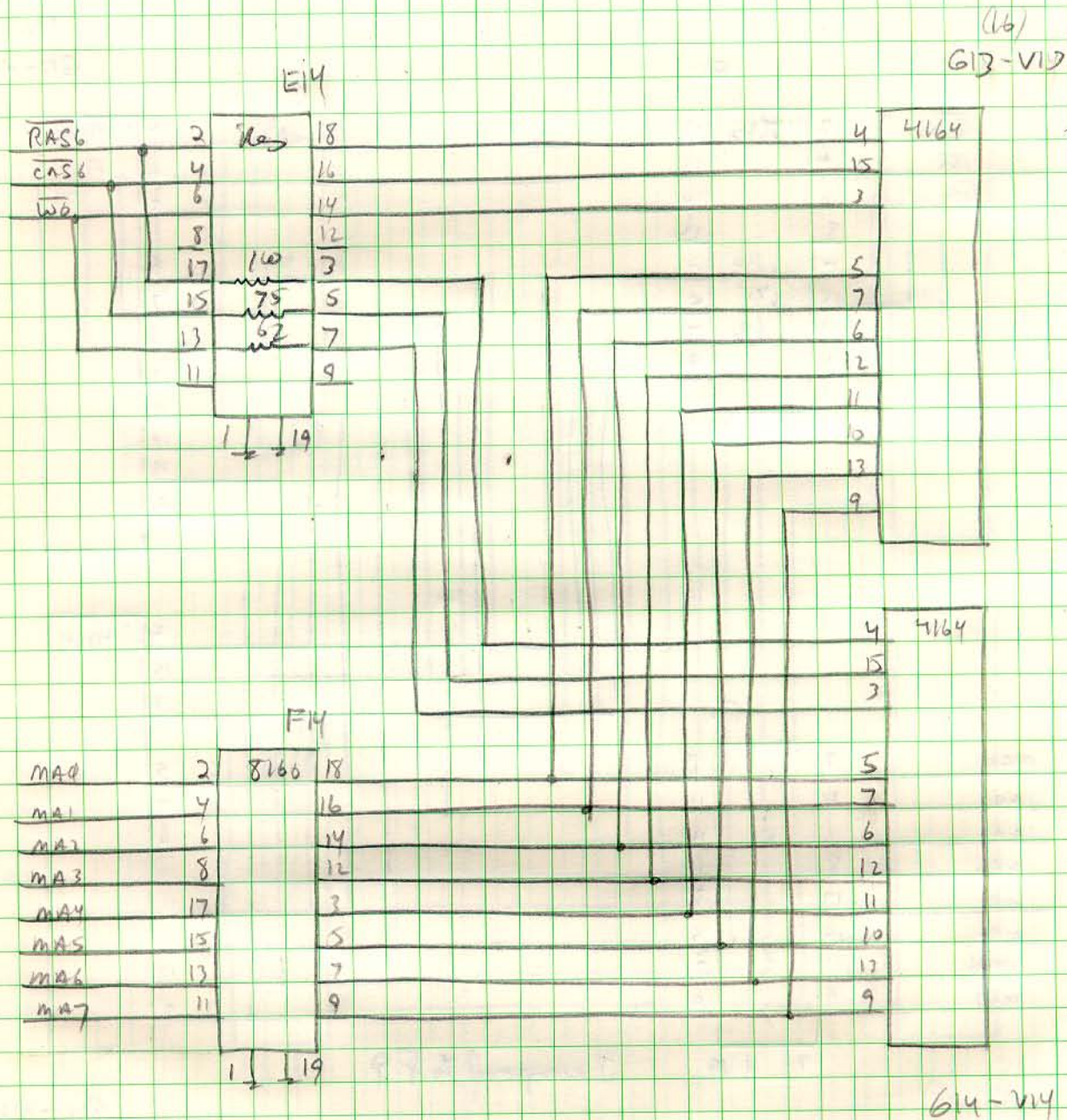


Jumpered to F9

G12-V12
(16)

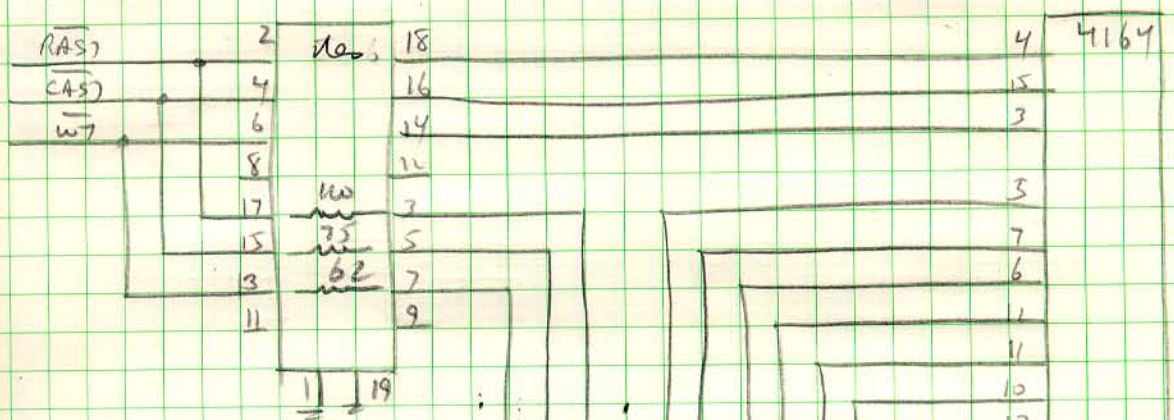
3 Jan 83
AGD

BNH6

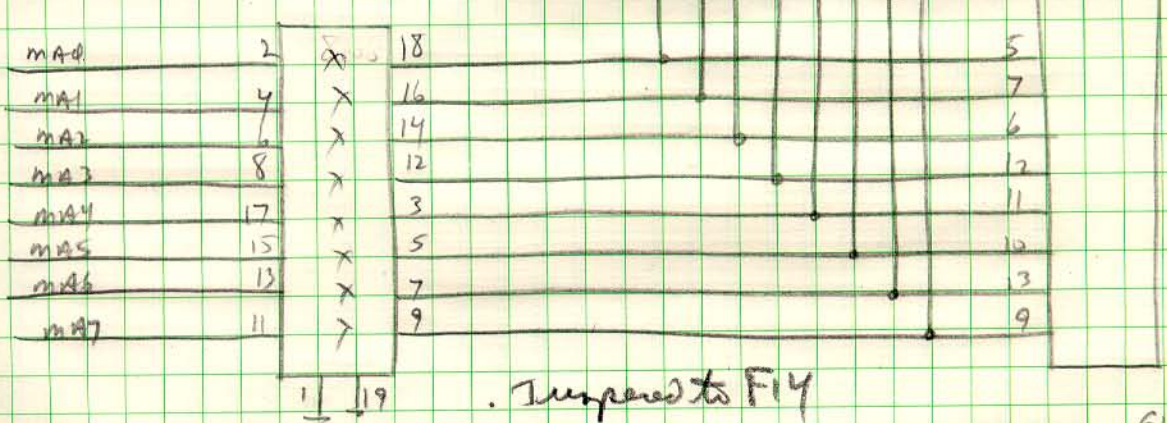

(16)
G13 - V13
E14
RAS6
CAS6
WE6
2
4
6
8
17
15
13
11
10
75
62
18
16
14
12
3
5
7
9
1
19
4164
4
15
3
5
7
6
12
11
10
13
9
F14
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7
8166
G14 - V14
(16)

(16)
G15-V15

E16



F16



. Jumpers to F14

G16-V16
(16)

3 Jan 83
ABB

IC LOCATION Index

A13	S32	35, 37
A15	S244	35
A16	8166	35
B13	S138	37
B15	S244	35
B16	8166	35
C13	S08	37
C15	S244	35
C16	8166	35
D9	S158	36
D10	S158	36
D11	S138	37
D12	S138	37
D13	S08	37
D15	S244	35
D16	8166	35

E1	8166	38
E2	—	
E3	8166	38
E6	8166	38
E7	—	
E8	8166	38
E9	8166	39
E10	S04	36
E11	8166	39
E14	8166	39
E15	—	
E16	8166	39

F1	8166	40
F2	—	
F3	8166	40
F6	8166	40
F7	—	
F8	8166	40
F9	8166	41
F10	—	
F11	8166	41
F14	8166	41
F15	—	
F16	8166	41

G(1,16) – V(1,16) 4164 38, 39, 40, 41

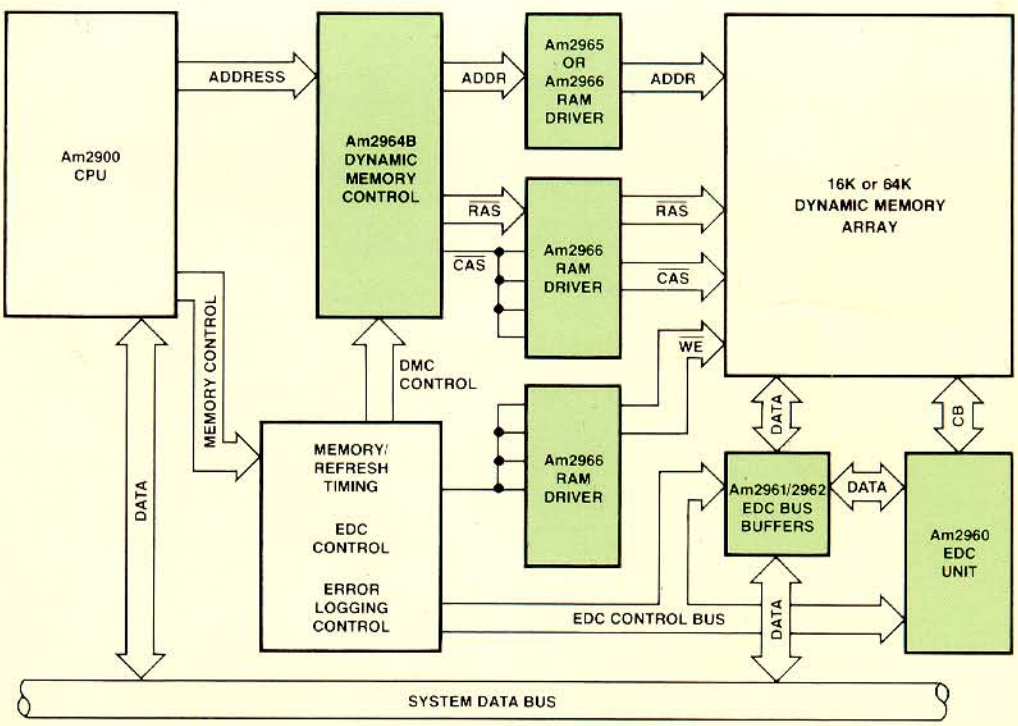
3 Jan 83
ARD

Advanced Micro Devices

Am2960 Series Dynamic Memory Support Handbook

- Error Detection/Correction
- Refresh and Control
- Data and Address Interface

HIGH PERFORMANCE COMPUTER MEMORY



Am2960
Fast Error Detection and Correction for Memories

Corrects All Single-Bit Errors

Corrects all single bit errors. Detects all double and some triple bit errors.

Expandable

One Am2960 provides Error Detection and Correction for 16-bits. Two Am2960s handle 32 bits; four Am2960s handle 64 bits.

Fast

Worst case 32 nanoseconds for error detect and 65 nanoseconds for error correct (16 bits).

Latches Built-In

Check Bit, Data, and Diagnostic latches are built-in to save MSI.

Flexible

Can be used with Am2900-based designs, the AmZ8000 or other processors.

Diagnostics Built-In

Logic on-chip for device test and software-controlled diagnostics.

Increases Memory Reliability

And can significantly reduce field maintenance costs.

A Must for 64K RAMs

Alpha error rates are several times higher for 64K RAMs than 16Ks.

Also available as the AmZ8160 for AmZ8000 Systems

DISTINCTIVE CHAR

- **Boosts Memory F** Corrects all single triple-bit errors. R creased more than
- **Very High Speed** Perfect for MOS r frame systems.
  - Data in to error
  - Data in to corre

  High performance check-only mode t
- **Replaces 25 to 50** All necessary featu ing diagnostics, da
- **Handles Data Wo** The Am2960 EDC bits, 4 for 64 bits.
- **Easy Byte Operat** Separate byte enab and cuts the time r
- **Diagnostics Built-** The processor ma software control to

TABLE OF CONTENT

Also available as the AmZ8160 for AmZ8000 Systems

Am2960 EDC

Cascadable 16-Bit Error Detection and Correction Unit

DISTINCTIVE CHARACTERISTICS

- **Boosts Memory Reliability**
  Corrects all single-bit errors. Detects all double and some triple-bit errors. Reliability of dynamic RAM systems is increased more than 60-fold.
- **Very High Speed**
  Perfect for MOS microprocessor, minicomputer, and mainframe systems.
  - Data in to error detect: 32ns worst case.
  - Data in to corrected data out: 65ns worst case.

  High performance systems can use the Am2960 EDC in check-only mode to avoid memory system slowdown.
- **Replaces 25 to 50 MSI chips**
  All necessary features are built-in to the Am2960 EDC, including diagnostics, data in, data out, and check bit latches.
- **Handles Data Words From 8 to 64 Bits**
  The Am2960 EDC cascades: 1 EDC for 8 or 16 bits, 2 for 32 bits, 4 for 64 bits.
- **Easy Byte Operations**
  Separate byte enables on the data out latch simplify the steps and cuts the time required for byte writes.
- **Diagnostics Built-In**
  The processor may completely exercise the EDC under software control to check for proper operation of the EDC.

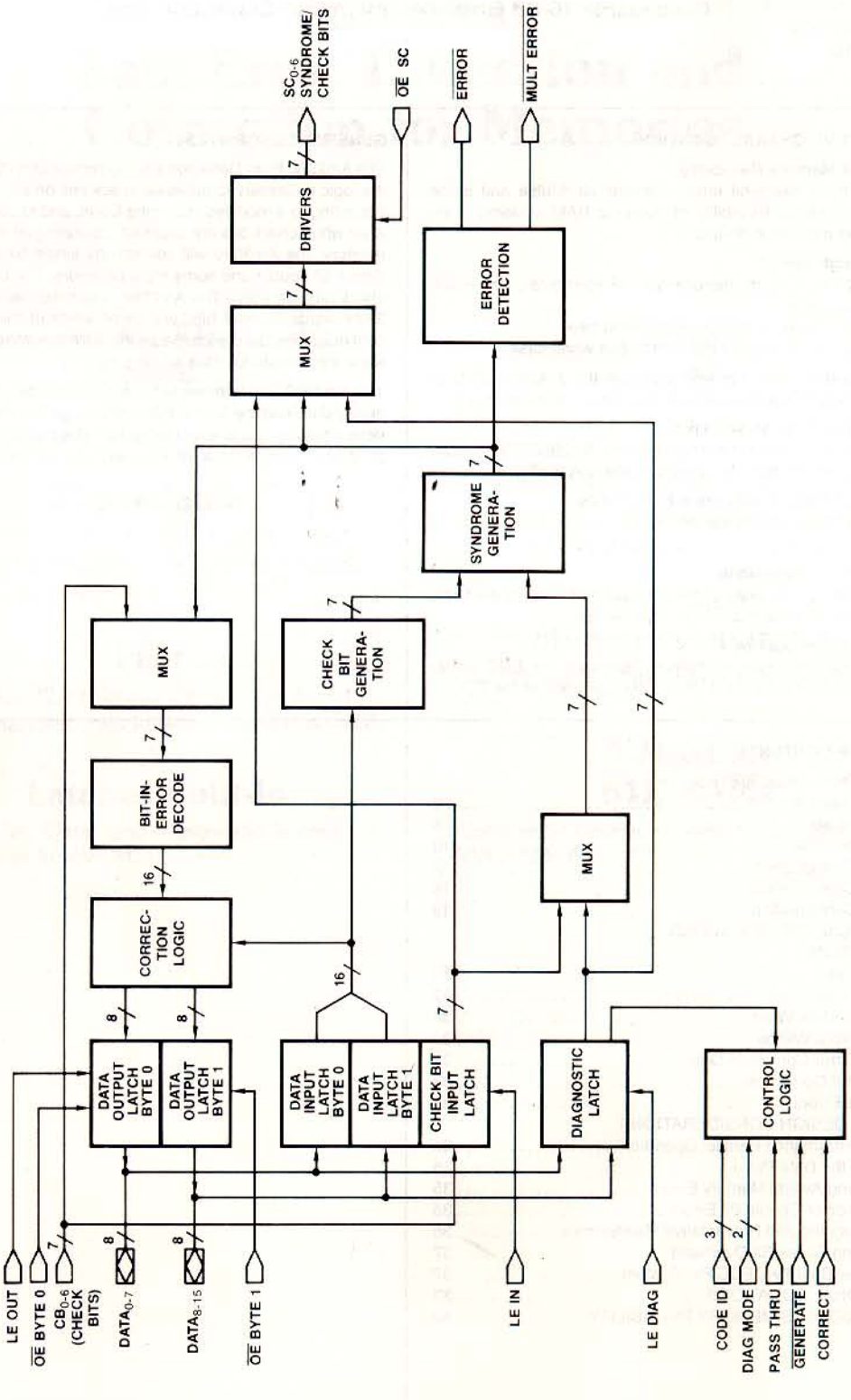
GENERAL DESCRIPTION

The Am2960 Error Detection and Correction Unit (EDC) contains the logic necessary to generate check bits on a 16-bit data field according to a modified Hamming Code, and to correct the data word when check bits are supplied. Operating on data read from memory, the Am2960 will correct any single bit error and will detect all double and some triple bit errors. For 16-bit words, 6 check bits are used. The Am2960 is expandable to operate on 32-bit words (7 check bits) and 64-bit words (8 check bits). In all configurations, the device makes the error syndrome available on separate outputs for data logging.

The Am2960 also features two diagnostic modes, in which diagnostic data can be forced into portions of the chip to simplify device testing and to execute system diagnostic functions. The product is supplied in a 48 lead hermetic DIP package.

TABLE OF CONTENTS

BLOCK DIAGRAM



MPR-729

EDC Architecture

The EDC Unit is a
check bit generation
nostics.

As shown in the bl
following:

- Data Input Latc
- Check Bit Input
- Check Bit Gene
- Syndrome Gen
- Error Detection
- Error Correctio
- Data Output La
- Diagnostic Latc
- Control Logic

Data Input Latch

16 bits of data are
under control of the L
control mode the inp
tion or error detectio

Check Bit Input La

Seven check bits ar
are used in the Erro

Check Bit Generati

This block generates
data in the Data Inp
cording to a modifie

Syndrome Generat

In both Error Detect
block compares the
newly generated set
from memory. If both
errors. If there is a
check bits is in erro

The syndrome bits a
sets of check bits.

EDC Architecture

The EDC Unit is a powerful 16-bit cascadable slice used for check bit generation, error detection, error correction and diagnostics.

As shown in the block diagram, the device consists of the following:

- Data Input Latch
- Check Bit Input Latch
- Check Bit Generation Logic
- Syndrome Generation Logic
- Error Detection Logic
- Error Correction Logic
- Data Output Latch
- Diagnostic Latch
- Control Logic

Data Input Latch

16 bits of data are loaded from the bidirectional DATA lines under control of the Latch Enable input, LE IN. Depending on the control mode the input data is either used for check bit generation or error detection/correction.

Check Bit Input Latch

Seven check bits are loaded under control of LE IN. Check bits are used in the Error Detection and Error Correction modes.

Check Bit Generation Logic

This block generates the appropriate check bits for the 16 bits of data in the Data Input Latch. The check bits are generated according to a modified Hamming code.

Syndrome Generation Logic

In both Error Detection and Error Correction modes, this logic block compares the check bits read from memory against a newly generated set of check bits produced for the data read in from memory. If both sets of check bits match, then there are no errors. If there is a mismatch, then one or more of the data or check bits is in error.

The syndrome bits are produced by an exclusive-OR of the two sets of check bits. If the two sets of check bits are identical (meaning there are no errors) the syndrome bits will be all zeroes. If there are errors, the syndrome bits can be decoded to determine the number of errors and the bit-in-error.

Error Detection Logic

This section decodes the syndrome bits generated by the Syndrome Generation Logic. If there are no errors in either the input data or check bits, the $\overline{\text{ERROR}}$ and $\overline{\text{MULT ERROR}}$ outputs remain HIGH. If one or more errors are detected, $\overline{\text{ERROR}}$ goes LOW. If two or more errors are detected, both $\overline{\text{ERROR}}$ and $\overline{\text{MULT ERROR}}$ go LOW.

Error Correction Logic

For single errors, the Error Correction Logic complements (corrects) the single data bit in error. This corrected data is loadable into the Data Output Latch, which can then be read onto the bidirectional data lines. If the single error is one of the check bits, the correction logic does not place corrected check bits on the syndrome/check bit outputs. If the corrected check bits are needed the EDC must be switched to Generate Mode.

Data Output Latch

The Data Output Latch is used for storing the result of an error correction operation. The latch is loaded from the correction logic under control of the Data Output Latch Enable, LE OUT. The Data Output Latch may also be loaded directly from the Data Input Latch under control of the PASS THRU control input.

The Data Output Latch is split into two 8-bit (byte) latches which may be enabled independently for reading onto the bidirectional data lines.

Diagnostic Latch

This is a 16-bit latch loadable from the bidirectional data lines under control of the Diagnostic Latch Enable, LE DIAG. The Diagnostic Latch contains check bit information in one byte and control information in the other byte. The Diagnostic Latch is used for driving the device when in Internal Control Mode, or for supplying check bits when in one of the Diagnostic Modes.

Control Logic

The control logic determines the specific mode the device operates in. Normally the control logic is driven by external control inputs. However, in Internal Control Mode, the control signals are instead read from the Diagnostic Latch.

PIN DEFINITIONS

$DATA_{0-15}$ 16 bidirectional data lines. They provide input to the Data Input Latch and Diagnostic Latch, and receive output from the Data Output Latch. $DATA_0$ is the least significant bit; $DATA_{15}$ the most significant.

CB_{0-6} Seven Check Bit input lines. The check bit lines are used to input check bits for error detection. Also used to input syndrome bits for error correction in 32 and 64-bit configurations.

LE IN Latch Enable – Data Input Latch. Controls latching of the input data. When HIGH the Data Input Latch and Check Bit Input Latch follow the input data and input check bits. When LOW, the Data Input Latch and Check Bit Input Latch are latched to their previous state.

$\overline{\text{GENERATE}}$ Generate Check Bits input. When this input is LOW the EDC is in the Check Bit Generate Mode. When HIGH the EDC is in the Detect Mode or Correct Mode.

In the Generate Mode the circuit generates the check bits or partial check bits specific to the data in the Data Input Latch. The generated check bits are placed on the SC outputs.

In the Detect or Correct Modes the EDC detects single and multiple errors, and generates syndrome bits based upon the contents of the Data Input Latch and Check Bit Input Latch. In Correct Mode, single-bit errors are also automatically corrected – corrected data is placed at the inputs of the Data Output Latch. The syndrome result is placed on the SC outputs and indicates in a coded form the number of errors and the bit-in-error.

SC_{0-6} Syndrome/Check Bit outputs. These seven lines hold the check/partial-check bits when the EDC is in Generate Mode, and will hold the syndrome/partial syndrome bits when the device is in Detect or Correct Modes. These are 3-state outputs.

$\overline{\text{OE SC}}$ Output Enable – Syndrome/Check Bits. When LOW, the 3-state output lines SC_{0-6} are enabled. When HIGH, the SC outputs are in the high impedance state.

$\overline{\text{ERROR}}$ Error Detected output. When the EDC is in Detect or Correct Mode, this output will go LOW if one or more syndrome bits are asserted, meaning there are one or more bit errors in the data or check bits. If no syndrome bits are asserted, there are no errors detected and the output will be HIGH. In Generate Mode, $\overline{\text{ERROR}}$ is forced HIGH. (In a 64-bit configuration, $\overline{\text{ERROR}}$ must be externally implemented.)

$\overline{\text{MULT ERROR}}$ Multiple Errors Detected output. When the EDC is in Detect or Correct Mode, this output if LOW indicates that there are two or more bit errors that have been detected. If HIGH this indicates that either one or no errors have been detected. In Generate mode, $\overline{\text{MULT ERROR}}$ is forced HIGH. (In a 64-bit configuration, $\overline{\text{MULT ERROR}}$ must be externally implemented.)

CORRECT Correct input. When HIGH this signal allows the correction network to correct any single-bit error in the Data Input Latch (by complementing the bit-in-error) before putting it onto the Data Output Latch. When LOW the EDC will drive data directly from the Data Input Latch to the Data Output Latch without correction.

LE OUT Latch Enable – Data Output Latch. Controls the latching of the Data Output Latch. When LOW the Data Output Latch is latched to its previous state. When HIGH the Data Output Latch follows the output of the Data Input Latch as modified by the correction logic network. In Correct Mode, single-bit errors are corrected by the network before loading into the Data Output Latch. In Detect Mode, the contents of the Data Input Latch are passed through the correction network unchanged into the Data Output Latch. The inputs to the Data Output Latch are unspecified if the EDC is in Generate Mode.

$\overline{\text{OE}}$ BYTE 0, $\overline{\text{OE}}$ BYTE 1 Output Enable – Bytes 0 and 1, Data Output Latch. These lines control the 3-state outputs for each of the two bytes of the Data Output Latch. When LOW these lines enable the Data Output Latch and when HIGH these lines force the Data Output Latch into the high impedance state. The two enable lines can be separately activated to enable only one byte of the Data Output Latch at a time.

PASS THRU Pass Thru input. This line when HIGH forces the contents of the Check Bit Input Latch onto the Syndrome/Check Bit outputs (SC_{0-6}) and the unmodified contents of the Data Input Latch onto the inputs of the Data Output Latch.

DIAG $MODE_{0-1}$ Diagnostic Mode Select. These two lines control the initialization and diagnostic operation of the EDC.

CODE ID_{0-2} Code Identification inputs. These three bits identify the size of the total data word to be processed and which 16-bit slice of larger data words a particular EDC is processing. The three allowable data word sizes are 16, 32 and 64 bits and their respective modified Hamming codes are designated 16/22, 32/39 and 64/72. Special CODE ID input 001 (ID_2, ID_1, ID_0) is also used to instruct the EDC that the signals CODE ID_{0-2}, DIAG $MODE_{0-1}$, CORRECT and PASS THRU are to be taken from the Diagnostic Latch, rather than from the input control lines.

LE DIAG Latch Enable – Diagnostic Latch. When HIGH the Diagnostic Latch follows the 16-bit data on the input lines. When LOW the outputs of the Diagnostic Latch are latched to their previous states. The Diagnostic Latch holds diagnostic check bits, and internal control signals for CODE ID_{0-2}, DIAG $MODE_{0-1}$, CORRECT and PASS THRU.

FUNCTIONAL DESCRIPTION

The EDC contains the logic necessary to generate check bits on a 16-bit data field according to a modified Hamming code. Operating on data read from memory, the EDC will correct any single-bit error, and will detect all double and some triple-bit errors. The Am2960 may be configured to operate on 16-bit data words (with 6 check bits), 32-bit data words (with 7 check bits) and 64-bit data words (with 8 check bits). In fact the EDC can be configured to work on data words from 8 to 64 bits. In all configurations, the device makes the error syndrome bits available on separate outputs for error data logging.

Code and Byte Specification

The EDC may be configured in several different ways and operates differently in each configuration. It is necessary to indicate to the device what size data word is involved and which bytes of the data word it is processing. This is done with input lines CODE ID_{0-2}, as shown in Table I. The three modified Hamming codes referred to in Table I are:

- 16/22 – 16 data bits
  – 6 check bits
  – 22 bits in total.
- 32/39 code – 32 data bits
  – 7 check bits
  – 39 bits in total.
- 64/72 code – 64 data bits
  – 8 check bits
  – 72 bits in total.

CODE ID input 001 (ID_2, ID_1, $!D_0$) is a special code used to operate the device in Internal Control Mode (described later in this section).

Control Mode Selection

The device control lines are $\overline{\text{GENERATE}}$, CORRECT, PASS THRU, DIAG $MODE_{0-1}$ and CODE ID_{0-2}. Table III indicates the operating modes selected by various combinations of the control line inputs.

Diagnostics

Table II shows specifically how DIAG $MODE_{0-1}$ select between normal operation, initialization and one of two diagnostic modes.

The Diagnostic Modes allow the user to operate the EDC under software control in order to verify proper functioning of the device.

Check and Syndrome Bit Labeling

The check bits generated in the EDC are designated as follows:

- 16-bit configuration – CX C0, C1, C2, C4, C8;
- 32-bit configuration – CX, C0, C1, C2, C4, C8, C16;
- 64-bit configuration – CX, C0, C1, C2, C4, C8, C16, C32.

Syndrome bits are similarly labeled SX through S32. There are only 6 syndrome bits in the 16-bit configuration, 7 for 32 bits and 8 syndrome bits in the 64-bit configuration.

Initialize Mode

The inputs of the Data Output Latch are forced to zeroes. The check bit outputs (SC) are generated to correspond to the all-zero data. $\overline{\text{ERROR}}$ and $\overline{\text{MULT ERROR}}$ are forced HIGH in the Initialize Mode.

Initialize Mode is useful after power up when RAM contents are random. The EDC may be placed in initialize mode and its' outputs written in to all memory locations by the processor.

TABLE I. HAMMING CODE AND SLICE IDENTIFICATION.

CODE ID_2	CODE ID_1	CODE ID_0	Hamming Code and Slice Selected
0	0	0	Code 16/22
0	0	1	Internal Control Mode
0	1	0	Code 32/39, Bytes 0 and 1
0	1	1	Code 32/39, Bytes 2 and 3
1	0	0	Code 64/72, Bytes 0 and 1
1	0	1	Code 64/72, Bytes 2 and 3
1	1	0	Code 64/72, Bytes 4 and 5
1	1	1	Code 64/72, Bytes 6 and 7

TABLE II. DIAGNOSTIC MODE CONTROL.

DIAG $MODE_1$	DIAG $MODE_0$	Diagnostic Mode Selected
0	0	**Non-diagnostic mode.** The EDC functions normally in all modes.
0	1	**Diagnostic Generate.** The contents of the Diagnostic Latch are substituted for the normally generated check bits when in the Generate Mode. The EDC functions normally in the Detect or Correct modes.
1	0	**Diagnostic Detect/Correct.** In the Detect or Correct Mode, the contents of the Diagnostic Latch are substituted for the check bits normally read from the Check Bit Input Latch. The EDC functions normally in the Generate Mode.
1	1	**Initialize.** The outputs of the Data Input Latch are forced to zeroes (and latched upon removal of the Initialize Mode) and the check bits generated correspond to the all-zero data.

HAMMING CODE SELECTION

The Am2960 EDC uses a modified Hamming Code that allows 1) the EDC to be cascaded, 2) all double errors to be detected, 3) the gross error conditions of all 0s or 1s to be detected.

The error correction code can be selected independent of the processor with the exception of diagnostics software.

Diagnostic software run by a processor to checkout the EDC system must know specifically which code is being used. This is only a problem when the EDC replaces an existing MSI implementation on an existing computer. In this case, the computer's software must first determine which of two codes (the old one used by the MSI implementation or the new one used by the EDC) is used by the computer's memory system.

This is easily determined by writing a test data word into memory and then examining whether the generated check bits are typical of the old or the new code. From then on the software runs only the diagnostic appropriate for the code used on that particular computer's memory system.

TABLE III. Am2960 OPERATING MODES

Operating Mode	Diagnostic Mode**		$\overline{\text{GENERATE}}$	
	DM_1	DM_0	0	1
Normal	0	0	Generate	Correct*
Diagnostic Generate	0	1	Diagnostic Generate	Correct*
Diagnostic Correct	1	0	Generate	Diagnostic Correct*
Initialize	1	1	Initialize	Initialize
Pass Thru	When PASS THRU is asserted the Operating Mode is defaulted to the Pass Thru Mode.			

*Correct if the CORRECT Input is HIGH, Detect if the CORRECT Input is LOW.
**In Code $ID_{2\text{-}0}$ 001 (ID_2, ID_1, ID_0) DM_1 and DM_0 are taken from the Diagnostic Latch.

FUNCTIONAL DESCRIPTION – 16-BIT DATA WORD CONFIGURATION

The 16-bit format consists of 16 data bits, 6 check bits and is referred to as 16/22 code (see Figure 5.)

The 16-bit configuration is shown in Figure 6.

Generate Mode

In this mode check bits will be generated that correspond to the contents of the Data Input Latch. The check bits generated are placed on the outputs SC_{0-5} (SC_6 is a logical one, or high).

Check bits are generated according to a modified Hamming code. Details of the code for check bit generation are contained in Table IV. Each check bit is generated as either an XOR or XNOR of eight of the 16 data bits as indicated in the table. The XOR function results in an even parity check bit, the XNOR is an odd parity check bit.

Figure 1 shows the data flow in the Generate Mode.

Detect Mode

In this mode the device examines the contents of the Data Input Latch against the Check Bit Input Latch, and will detect all single-bit errors, all double-bit errors and some triple-bit errors. If one or more errors are detected, $\overline{\text{ERROR}}$ goes LOW. If two or more errors are detected, $\overline{\text{MULT ERROR}}$ goes LOW. Both error indicators are HIGH if there are no errors.

Also available on device outputs SC_{0-5} are the syndrome bits generated by the error detection step. The syndrome bits may be decoded to determine if a bit error was detected and, for single-bit errors, which of the data or check bits is in error. Table V gives the chart for decoding the syndrome bits generated by the 16-bit configuration (as an example, if the syndrome bits SX/S0/S1/S2/S4/S8 were 101001 this would be decoded to indicate that there is a single-bit error at data bit 9). If no error is detected the syndrome bits will all be zeroes.

In Detect Mode, the contents of the Data Input Latch are driven directly to the inputs of the Data Output Latch without correction.

Correct Mode

In this mode, the EDC functions the same as in Detect Mode except that the correction network is allowed to correct (complement) any single-bit error of the Data Input Latch before putting it onto the inputs of the Data Output Latch. (see Figure 2.) If multiple errors are detected, the output of the correction network is unspecified. If the single-bit error is a check bit there is no automatic correction. If check bit correction is desired, this can be done by placing the device in Generate Mode to produce a correct check bit sequence for the data in the Data Input Latch.

Pass Thru Mode

In this mode, the unmodified contents of the Data Input Latch are placed on the inputs of the Data Output Latch and the contents of the Check Bit Input Latch are placed on outputs SC_{0-5}. $\overline{\text{ERROR}}$ and $\overline{\text{MULT ERROR}}$ are forced HIGH in this mode.

Diagnostic Latch

The Diagnostic Latch serves both for diagnostic uses and internal control uses. It is loaded from the DATA lines under the control of LE DIAG. Table VI shows the loading definitions for the DATA lines.

Diagnostic Generate
Diagnostic Detect
Diagnostic Correct

These are special diagnostic modes selected by DIAG $MODE_{0-1}$ where either normal check bit inputs or outputs are substituted for by check bits loaded into the Diagnostic Latch. See Table III for details. Figures 3 and 4 illustrate the flow of data during the two diagnostic modes.

Internal Control Mode

This mode is selected by CODE ID_{0-2} input 001 (ID_2, ID_1, ID_0).

When in Internal Control Mode, the EDC takes the CODE ID_{0-2}, DIAG $MODE_{0-1}$, CORRECT and PASS THRU control signals from the internal Diagnostic Latch rather than from the external input lines.

Table VI gives the format for loading the Diagnostic Latch.

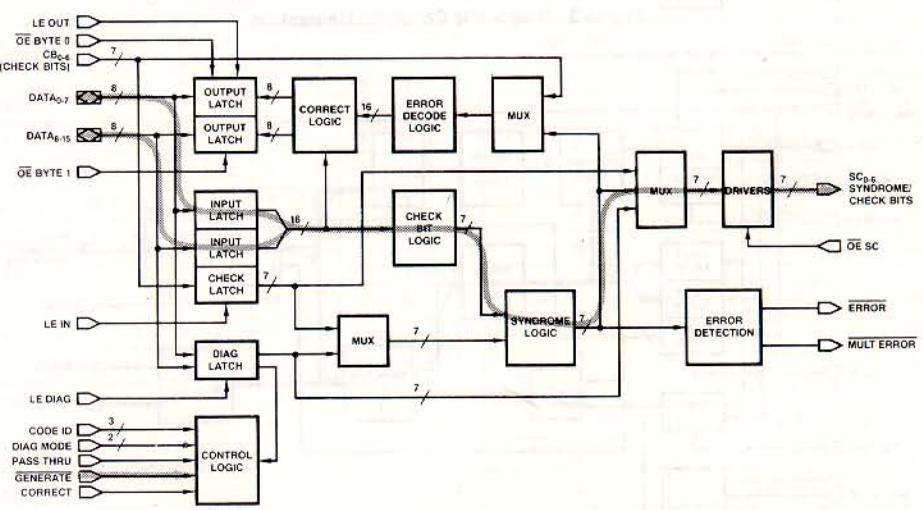


Figure 1. Check Bit Generation

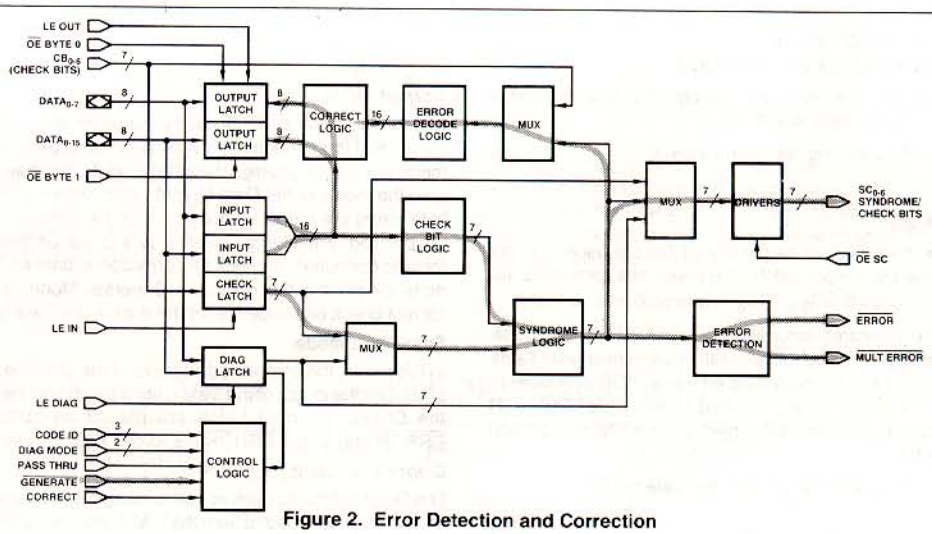


Figure 2. Error Detection and Correction

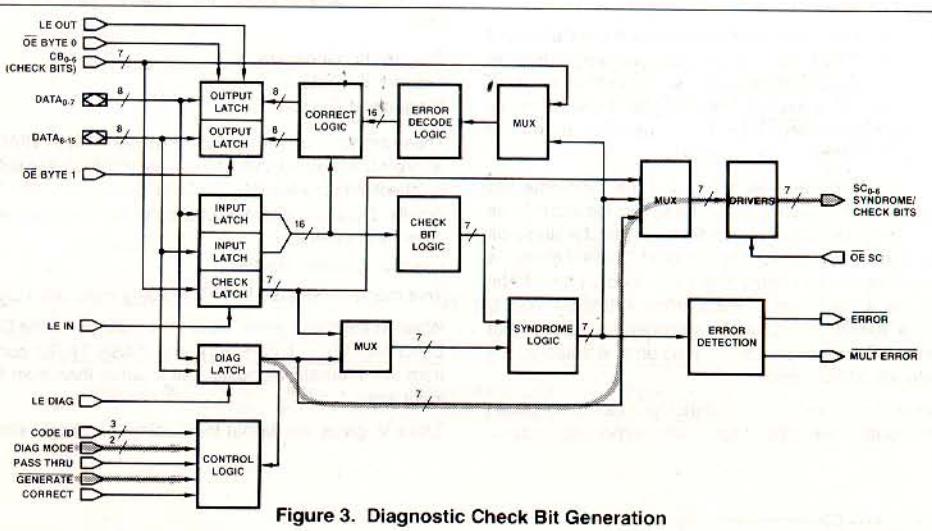


Figure 3. Diagnostic Check Bit Generation

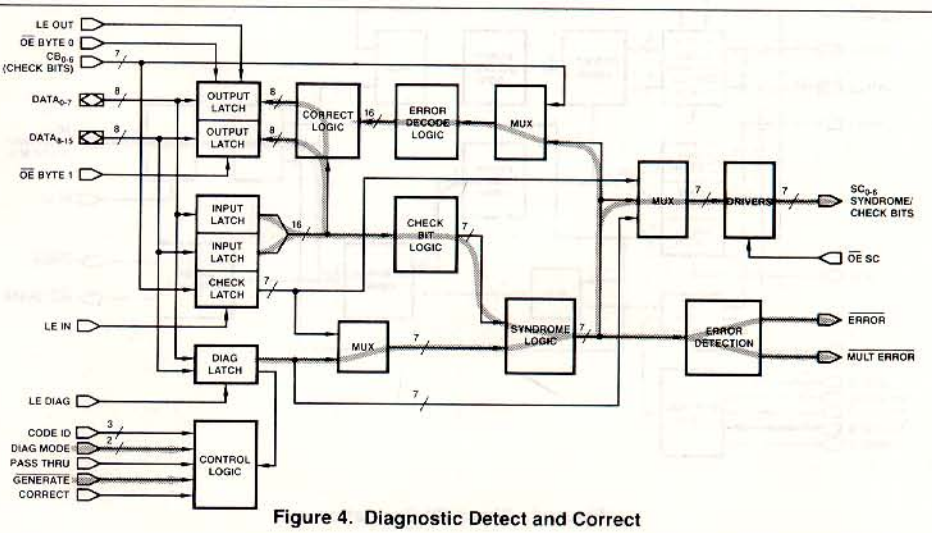
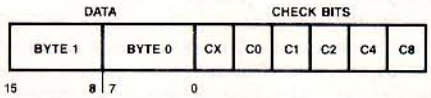


Figure 4. Diagnostic Detect and Correct



Uses Modified Hamming Code 16/22
- 16 data bits
- 6 check bits
- 22 bits in total

Figure 5. 16-Bit Data Format

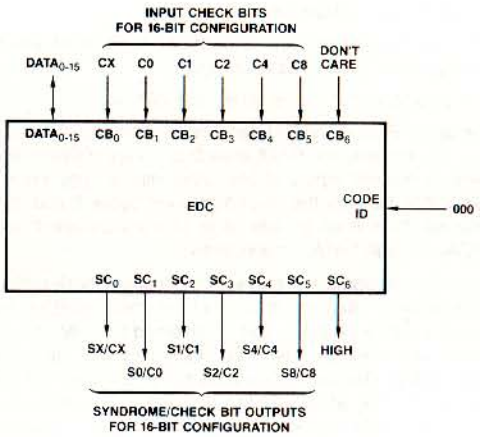


Figure 6. 16-Bit Configuration

TABLE IV. 16-BIT MODIFIED HAMMING CODE – CHECK BIT ENCODE CHART.

Generated Check Bits	Parity	Participating Data Bits															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CX	Even (XOR)		X	X	X		X			X	X		X			X	
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X

The check bit is generated as either an XOR or XNOR of the eight data bits noted by an "X" in the table.

TABLE V. SYNDROME DECODE TO BIT-IN-ERROR.

Syndrome Bits										
			S8	0	1	0	1	0	1	0	1
			S4	0	0	1	1	0	0	1	1
			S2	0	0	0	0	1	1	1	1
SX	S0	S1									
0	0	0		*	C8	C4	T	C2	T	T	M
0	0	1		C1	T	T	15	T	13	7	T
0	1	0		C0	T	T	M	T	12	6	T
0	1	1		T	10	4	T	0	T	T	M
1	0	0		CX	T	T	14	T	11	5	T
1	0	1		T	9	3	T	M	T	T	M
1	1	0		T	8	2	T	1	T	T	M
1	1	1		M	T	T	M	T	M	M	T

* – no errors detected
Number – the location of the single bit-in-error
T – two errors detected
M – three or more errors detected

TABLE VI. DIAGNOSTIC LATCH LOADING – 16-BIT FORMAT.

Data Bit	Internal Function
0	Diagnostic Check Bit X
1	Diagnostic Check Bit 0
2	Diagnostic Check Bit 1
3	Diagnostic Check Bit 2
4	Diagnostic Check Bit 4
5	Diagnostic Check Bit 8
6, 7	Don't Care
8	CODE ID 0
9	CODE ID 1
10	CODE ID 2
11	DIAG MODE 0
12	DIAG MODE 1
13	CORRECT
14	PASS THRU
15	Don't Care

FUNCTIONAL DESCRIPTION – 32-BIT DATA WORD CONFIGURATION

The 32-bit format consists of 32 data bits, 7 check bits and is referred to as 32/39 code (see Figure 7).

The 32-bit configuration is shown in Figure 8.

The upper EDC (Slice 0/1) handles the least significant bytes 0 and 1 – the external DATA lines 0 to 15 are connected to the same numbered inputs of the upper device. The lower EDC (Slice 2/3) handles the most significant bytes 2 and 3 – the external DATA lines for bits 16 to 31 are connected to inputs $DATA_0$ through $DATA_{15}$ respectively.

The valid syndrome and check bit outputs are those of Slice 2/3 as shown in the diagram. In Correct Mode these must be read into Slice 0/1 via the CB inputs and are selected by the MUX as inputs to the bit-in-error decoder (see block diagram), thus requiring external buffering and output enabling of the check bit lines as shown. The $\overline{OE}$ SC signal can be used to control enabling of check bit inputs – when syndrome outputs are enabled, the external check bit inputs will be disabled.

The valid $\overline{ERROR}$ and $\overline{MULT\ ERROR}$ outputs are those of the Slice 2/3. The $\overline{ERROR}$ and $\overline{MULT\ ERROR}$ outputs of Slice 0/1 are unspecified. All of the latch enables and control signals must be input to both of the devices.

Generate Mode

In this mode check bits will be generated that correspond to the contents of the Data Input Latch. The check bits generated are placed on the outputs $SC_{0\text{-}6}$ of Slice 2/3.

Check bits are generated according to a modified Hamming code. Details of the code for check bit generation are contained in Table X. Check bits are generated as either an XOR or XNOR of 16 of the 32 data bits as indicated in the table. The XOR function results in an even parity check bit, the XNOR in an odd parity check bit.

Detect Mode

In this mode the device examines the contents of the Data Input Latch against the Check Bit Input Latch, and will detect all single-bit errors, all double-bit errors and some triple-bit errors. If one or more errors are detected, $\overline{ERROR}$ goes LOW. If two or more errors are detected, $\overline{MULT\ ERROR}$ goes LOW. Both error indicators are HIGH if there are no errors. The valid $\overline{ERROR}$ and $\overline{MULT\ ERROR}$ signals are those of Slice 2/3 – those of Slice 0/1 are undefined.

Also available on Slice 2/3 outputs $SC_{0\text{-}6}$ are the syndrome bits generated by the error detection step. The syndrome bits may be decoded to determine if a bit error was detected and, for single-bit errors, which of the data or check bits is in error. Table VII gives the chart for decoding the syndrome bits generated for the 32-bit configuration (as an example, if the syndrome bits SX/S0/S1/S2/S4/S8/S16 were 0010011 this would be decoded to indicate that there is a single-bit error at data bit 25). If no error is detected the syndrome bits will be all zeroes.

In Detect Mode, the contents of the Data Input Latch are driven directly to the inputs of the Data Output Latch without corrections.

Correct Mode

In this mode, the EDC functions the same as in Detect Mode except that the correction network is allowed to correct (complement) any single-bit error of the Data Input Latch before putting it onto the inputs of the Data Output Latch. If multiple errors are detected, the output of the correction network is unspecified. If the single-bit error is a check bit there is no automatic correction – if desired this would be done by placing the device in Generate Mode to produce a correct check bit sequence for the data in the Data Input Latch.

For data correction, both Slices 0/1 and 2/3 require access to the syndrome bits on Slice 2/3's outputs $SC_{0\text{-}6}$. Slice 2/3 has access to these syndrome bits through internal data paths, but for Slice 0/1 they must be read through the inputs $CB_{0\text{-}6}$. The device connections for this are shown in Figure 8. When in Correct Mode the SC outputs must be enabled so that they are available for reading in through the CB inputs.

Pass Thru Mode

In this mode, the unmodified contents of the Data Input Latch are placed on the inputs of the Data Output Latch and the contents of the Check Bit Input Latch are placed on outputs $SC_{0\text{-}6}$ of Slice 2/3. $\overline{ERROR}$ and $\overline{MULT\ ERROR}$ are forced HIGH in this mode.

TABLE VII. SYNDROME DECODE TO BIT-IN-ERROR.

Syndrome Bits				S16	0	1	0	1	0	1	0	1
				S8	0	0	1	1	0	0	1	1
				S4	0	0	0	0	1	1	1	1
SX	S0	S1	S2									
0	0	0	0		*	C16	C8	T	C4	T	T	30
0	0	0	1		C2	T	T	27	T	5	M	T
0	0	1	0		C1	T	T	25	T	3	15	T
0	0	1	1		T	M	13	T	23	T	T	M
0	1	0	0		C0	T	T	24	T	2	M	T
0	1	0	1		T	1	12	T	22	T	T	M
0	1	1	0		T	M	10	T	20	T	T	M
0	1	1	1		16	T	T	M	T	M	M	T
1	0	0	0		CX	T	T	M	T	M	14	T
1	0	0	1		T	M	11	T	21	T	T	M
1	0	1	0		T	M	9	T	19	T	T	31
1	0	1	1		M	T	T	29	T	7	M	T
1	1	0	0		T	M	8	T	18	T	T	M
1	1	0	1		17	T	T	28	T	6	M	T
1	1	1	0		M	T	T	26	T	4	M	T
1	1	1	1		T	0	M	T	M	T	T	M

* – no errors detected
Numbers – number of the single bit-in-error
T – two errors detected
M – three or more errors detected

Uses Modified Hamming Code 32/39
- 32 data bits
- 7 check bits
- 39 bits in total

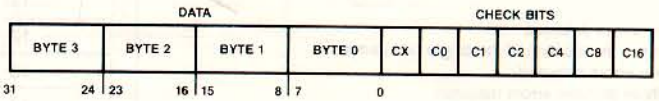
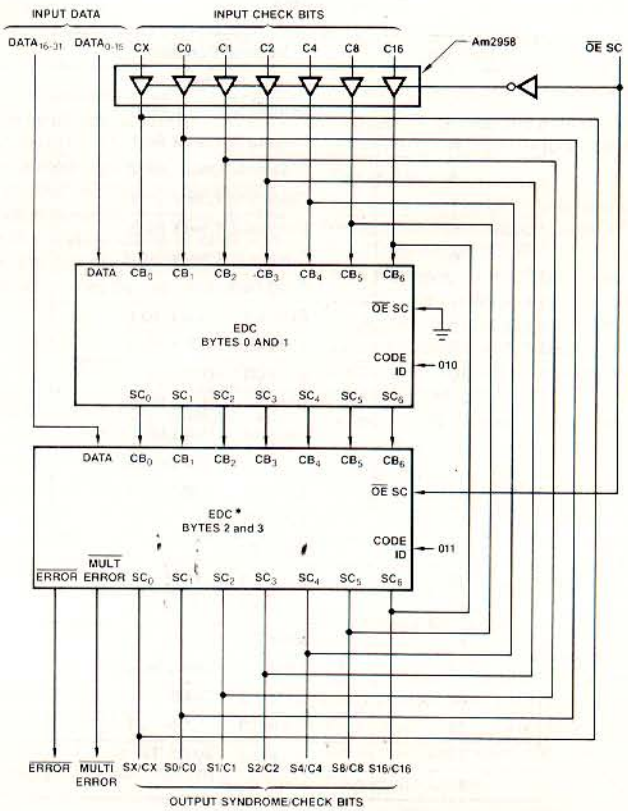


Figure 7. 32-Bit Data Format

MPR-732



*Check Bit Latch is Forced Transparent in this Code ID Combination for this Slice.

Figure 8. 32-Bit Configuration

TABLE VIII. KEY AC CALCULATIONS FOR THE 32-BIT CONFIGURATION

32-Bit Propagation Delay		Component Delay from Am2960 AC Specifications, Table C
From	To	
DATA	Check Bits Out	(DATA to SC) + (CB to SC, CODE ID 011)
DATA In	Corrected DATA Out	(DATA to SC) + (CB to SC, CODE ID 011) + (CB to DATA, CODE ID 010)
DATA	Syndromes Out	(DATA to SC) + (CB to SC, CODE ID 011)
DATA	$\overline{\text{ERROR}}$ for 32 Bits	(DATA to SC) + (CB to $\overline{\text{ERROR}}$, CODE ID 011)
DATA	$\overline{\text{MULT ERROR}}$ for 32 Bits	(DATA to SC) + (CB to $\overline{\text{MULT ERROR}}$, CODE ID 011)

35+28 = 63

35+28+45 = 108

35+28 = 63

35+29 = 64

35+47 = 92

TABLE IX. DIAGNOSTIC LATCH LOADING – 32-BIT FORMAT.

Data Bit	Internal Function
0	Diagnostic Check Bit X
1	Diagnostic Check Bit 0
2	Diagnostic Check Bit 1
3	Diagnostic Check Bit 2
4	Diagnostic Check Bit 4
5	Diagnostic Check Bit 8
6	Diagnostic Check Bit 16
7	Don't Care
8	Slice 0/1 – CODE ID 0
9	Slice 0/1 – CODE ID 1
10	Slice 0/1 – CODE ID 2
11	Slice 0/1 – DIAG MODE 0
12	Slice 0/1 – DIAG MODE 1
13	Slice 0/1 – CORRECT
14	Slice 0/1 – PASS THRU
15	Don't Care
16-23	Don't Care
24	Slice 2/3 – CODE ID 0
25	Slice 2/3 – CODE ID 1
26	Slice 2/3 – CODE ID 2
27	Slice 2/3 – DIAG MODE 0
28	Slice 2/3 – DIAG MODE 1
29	Slice 2/3 – CORRECT
30	Slice 2/3 – PASS THRU
31	Don't Care

TABLE X. 32-BIT MODIFIED HAMMING CODE – CHECK BIT ENCODE CHART.

Generated Check Bits	Parity	Participating Data Bits															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CX	Even (XOR)	X				X		X	X	X	X		X			X	
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)	X	X	X	X	X	X	X	X								

Generated Check Bits	Parity	Participating Data Bits															
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CX	Even (XOR)		X	X	X		X					X		X	X		X
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)									X	X	X	X	X	X	X	X

The check bit is generated as either an XOR or XNOR of the sixteen data bits noted by an "X" in the table.

FUNCTIONAL DESC
64-BIT DATA WORD

The 64-bit format co
referred to as 64/72 c

The configuration to p
this configuration a po
detection is implemen
correction the syndron
four EDCs through the
buffering shown in th
$\overline{OE}$ SC signal can co
drome bit outputs are
be disabled so that th
inputs.

The error detection sig
the 16 and 32-bit confi
same: it is LOW if one
errors are detected. T
only if a double-bit err
the $\overline{MULT\ ERROR}$ ou
$\overline{ERROR}$ is LOW for al
ERROR combinatior
Equations section.) It
detected.

This is a different m
configurations.

Generate Mode

In this mode check bit
contents of the Data I
pear at the outputs of

Check bits are generat
Details of the code for
XII. Check bits are ge
of the 64 bits as indi
sults in an even pari
check bit.

Detect Mode

In this mode the devic
Latch against the Ch
single-bit errors, all do
one or more errors ar
two errors are detecte
more errors are detect
$\overline{ERROR}$ output of any

Available as XOR gat
(see Figure 10). The
mine if a bit error was
of the data or check
for encoding the synd
uration (as an examp

BYTE 7

63

FUNCTIONAL DESCRIPTION – 64-BIT DATA WORD CONFIGURATION

The 64-bit format consists of 64 data bits, 8 check bits and is referred to as 64/72 code (see Figure 9.).

The configuration to process 64-bit format is shown in Figure 6. In this configuration a portion of the syndrome generation and error detection is implemented externally of the EDCs in MSI. For error correction the syndrome bits generated must be read back into all four EDCs through the CB inputs. This necessitates the check bit buffering shown in the connection diagram of Figure 10. The $\overline{OE}$ SC signal can control the check bit enabling – when syndrome bit outputs are enabled the external check bit lines will be disabled so that the syndrome bits may be read onto the CB inputs.

The error detection signals for the 64-bit configuration differ from the 16 and 32-bit configurations. The $\overline{ERROR}$ signal functions the same: it is LOW if one or more errors are detected, and HIGH if no errors are detected. The DOUBLE ERROR signal is HIGH if and only if a double-bit error is detected – it is LOW otherwise. All of the $\overline{MULT\ ERROR}$ outputs of the four devices are valid. $\overline{MULT\ ERROR}$ is LOW for all three ERROR cases and some DOUBLE ERROR combinations. (See TOME definition in Functional Equations section.) It is HIGH if either zero or one errors are detected.

This is a different meaning for $\overline{MULT\ ERROR}$ than in other configurations.

Generate Mode

In this mode check bits will be generated that correspond to the contents of the Data Input Latch. The check bits generated appear at the outputs of the XOR gates as indicated in Figure 10.

Check bits are generated according to a modified Hamming code. Details of the code for check bit generation are contained in Table XII. Check bits are generated as either an XOR or XNOR of 32 of the 64 bits as indicated in the table. The XOR function results in an even parity check bit, the XNOR in an odd parity check bit.

Detect Mode

In this mode the device examines the contents of the Data Input Latch against the Check Bit Input Latch, and will detect all single-bit errors, all double-bit errors and some triple-bit errors. If one or more errors are detected, $\overline{ERROR}$ goes LOW. If exactly two errors are detected, DOUBLE ERROR goes HIGH. If three or more errors are detected, $\overline{MULT\ ERROR}$ goes LOW – the $\overline{MULT\ ERROR}$ output of any of the four EDCs may be used.

Available as XOR gate outputs are the generated syndrome bits (see Figure 10). The syndrome bits may be decoded to determine if a bit error was detected and, for single-bit errors, which of the data or check bits is in error. Table XIII gives the chart for encoding the syndrome bits generated for the 64-bit configuration (as an example, if the syndrome bits SX/S1/S2/S4/S8/S16/S32 were 00100101 this would be decoded to indicate that there is a single-bit error at data bit 41). If no error is detected the syndrome bits will all be zeroes.

In Detect Mode the contents of the Data Input Latch are driven directly to the inputs of the Data Output Latch without corrections.

Correct Mode

In this mode, the EDC functions the same as in Detect Mode except that the correction network is allowed to correct (complement) any single-bit error of the Data Input Latch before putting it onto the inputs of the Data Output Latch. If multiple errors are detected, the output of the correction network is unspecified. If the single bit error is a check bit there is no automatic correction. Check bit correction can be done by placing the device in generate mode to produce a correct check bit sequence for the data in the Data Input Latch.

To perform the correction step, all four slices require access to the syndrome bits which are generated externally of the devices. This access is provided by reading the syndrome bits in through the CB inputs where they are selected as inputs to the bit-in-error decoder by the multiplexer (see block diagram). The device connections for this are shown in Figure 10. When in Correct Mode the SC outputs must be enabled so that the syndrome bits are available at the CB inputs.

Pass Thru Mode

In this mode, the unmodified contents of the Data Input Latch are placed on the inputs of the Data Output Latch, and the contents of the Check Bit Input Latch are passed through the external XOR network and appear inverted at the XOR gate outputs labeled CX to C32 (see Figure 10).

Diagnostic Latch

The Diagnostic Latch serves both for diagnostic uses and internal control uses. It is loaded from the DATA lines under the control of LE DIAG. Table XIV shows the loading definitions for the DATA lines.

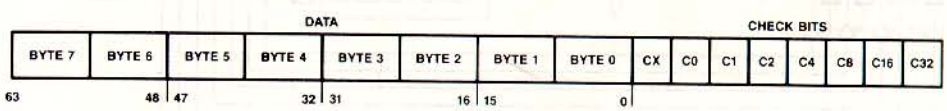
Diagnostic Generate
Diagnostic Detect
Diagnostic Correct

These are special diagnostic modes selected by DIAG $MODE_{0\text{-}1}$ where either normal check bit inputs or outputs are substituted for by check bits from the Diagnostic Latch. See Table II for details.

Internal Control Mode

This mode is selected by CODE $ID_{0\text{-}2}$, input 001 (ID_2, ID_1, ID_0).

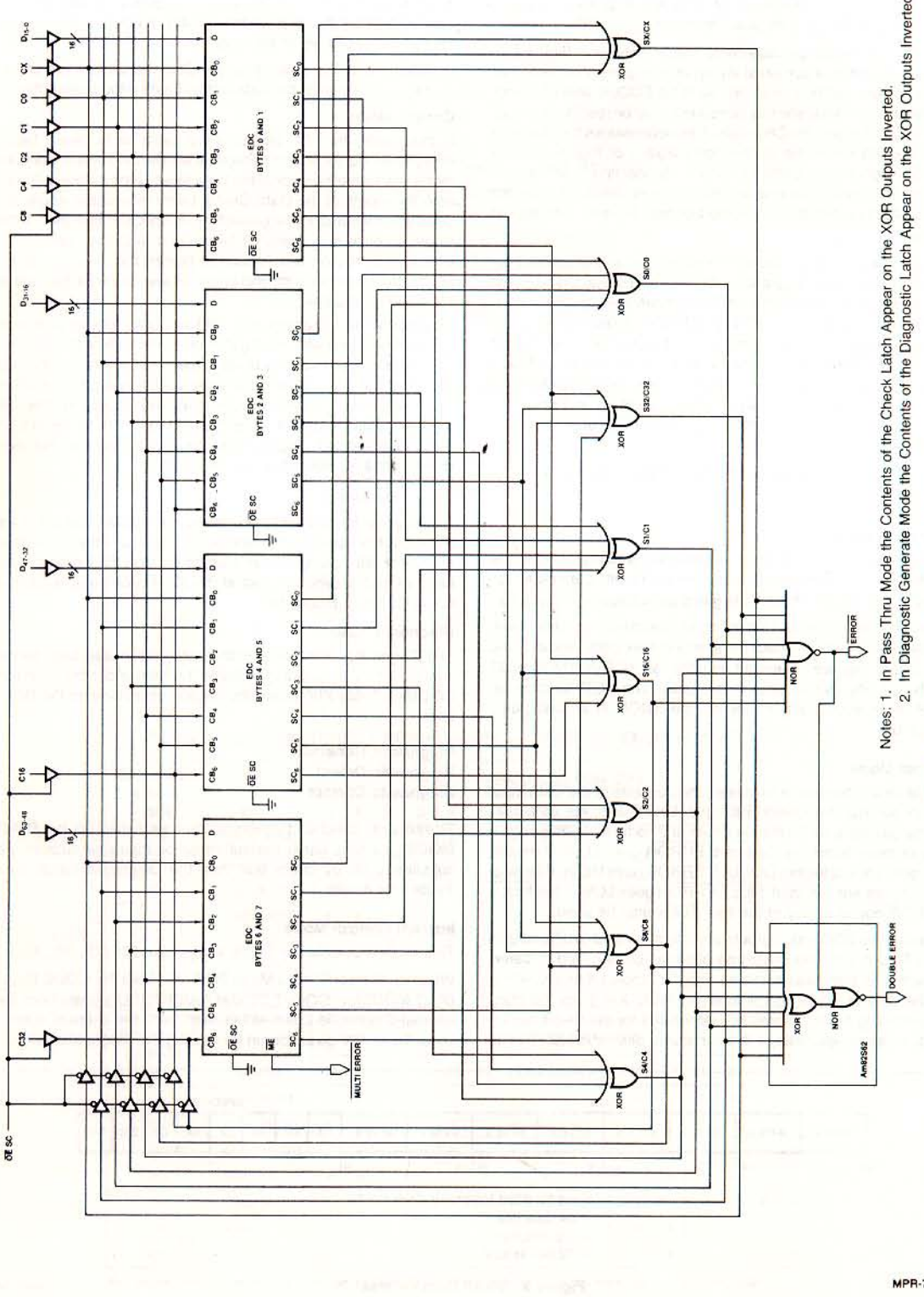
When in Internal Control Mode the EDC takes the CODE $ID_{0\text{-}2}$, DIAG $MODE_{0\text{-}1}$, CORRECT and PASS THRU signals from the internal Diagnostic Latch rather than from the external control lines. Table XIV gives format for loading the Diagnostic Latch.



Uses Modified Hamming Code 64/72
- 64 data bits
- 8 check bits
- 72 bits in total

Figure 9. 64-Bit Data Format

MPR-734



Notes: 1. In Pass Thru Mode the Contents of the Check Latch Appear on the XOR Outputs Inverted.
2. In Diagnostic Generate Mode the Contents of the Diagnostic Latch Appear on the XOR Outputs Inverted.

Figure 10. Am2960 – 64-Bit Data Configuration

TABLE

	Pro
From	
DATA	C
DATA In	C
DATA	S
DATA	$\overline{E}$
DATA	$\overline{M}$
DATA	D

TABLE XI. KEY AC CALCULATIONS FOR THE 64-BIT CONFIGURATION

64-Bit Propagation Delay		Component Delays from Am2960 AC Specifications, Table C (plus MSI)
From	To	
DATA	Check Bits Out	(DATA to SC) + (XOR Delay)
DATA In	Corrected DATA Out	(DATA to SC) + (XOR Delay) + (Buffer Delay) + (CB to DATA, CODE ID 1xx)
DATA	Syndromes	(DATA to SC) + (XOR Delay)
DATA	$\overline{\text{ERROR}}$ for 64 Bits	(DATA to SC) + (XOR Delay) + (NOR Delay)
DATA	$\overline{\text{MULT ERROR}}$ for 64 Bits	(DATA to SC) + (XOR Delay) + (Buffer Delay) + (CB to $\overline{\text{MULT ERROR}}$, CODE ID 1xx)
DATA	DOUBLE ERROR for 64 Bits	(DATA to SC) + (XOR Delay) + (XOR/NOR Delay)

Figure 10. Am2960 — 64-Bit Data Configuration

TABLE XII. 64-BIT MODIFIED HAMMING CODE – CHECK BIT ENCODE

Generated Check Bits	Parity	Participating Data Bits															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CX	Even (XOR)		X	X	X		X			X	X		X			X	
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)	X	X	X	X	X	X	X	X								
C32	Even (XOR)	X	X	X	X	X	X	X	X								

Generated Check Bits	Parity	Participating Data Bits															
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CX	Even (XOR)		X	X	X		X			X	X		X			X	
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)									X	X	X	X	X	X	X	X
C32	Even (XOR)									X	X	X	X	X	X	X	X

Generated Check Bits	Parity	Participating Data Bits															
		32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
CX	Even (XOR)	X				X		X	X			X		X	X		X
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)	X	X	X	X	X	X	X	X								
C32	Even (XOR)									X	X	X	X	X	X	X	X

Generated Check Bits	Parity	Participating Data Bits															
		48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CX	Even (XOR)	X				X		X	X			X		X	X		X
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)									X	X	X	X	X	X	X	X
C32	Even (XOR)	X	X	X	X	X	X	X	X								

The check bit is generated as either an XOR or XNOR of the 32 data bits noted by an "X" in the table.

Syndro Bits		
SX	S0	S
0	0	
0	0	
0	0	
0	0	
0	1	
0	1	
0	1	
0	1	
1	0	
1	0	
1	0	
1	0	
1	1	
1	1	
1	1	
1	1	

∗ – no errors
Number – the

Data Bit
0
1
2
3
4
5
6, 7
8
9
10
11
12
13
14
15
16-23
24
25
26
27
28
29
30

TABLE XIII. SYNDROME DECODE TO BIT-IN-ERROR.

Syndrome Bits				S32	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
				S16	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				S8	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				S4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
SX	S0	S1	S2																	
0	0	0	0		*	C32	C16	T	C8	T	T	M	C4	T	T	M	T	46	62	T
0	0	0	1		C2	T	T	M	T	43	59	T	T	53	37	T	M	T	T	M
0	0	1	0		C1	T	T	M	T	41	57	T	T	51	35	T	15	T	T	31
0	0	1	1		T	M	M	T	13	T	T	29	23	T	T	7	T	M	M	T
0	1	0	0		C0	T	T	M	T	40	56	T	T	50	34	T	M	T	T	M
0	1	0	1		T	49	33	T	12	T	T	28	22	T	T	6	T	M	M	T
0	1	1	0		T	M	M	T	10	T	T	26	20	T	T	4	T	M	M	T
0	1	1	1		16	T	T	0	T	M	M	T	T	M	M	T	M	T	T	M
1	0	0	0		CX	T	T	M	T	M	M	T	T	M	M	T	14	T	T	30
1	0	0	1		T	M	M	T	11	T	T	27	21	T	T	5	T	M	M	T
1	0	1	0		T	M	M	T	9	T	T	25	19	T	T	3	T	47	63	T
1	0	1	1		M	T	T	M	T	45	61	T	T	55	39	T	M	T	T	M
1	1	0	0		T	M	M	T	8	T	T	24	18	T	T	2	T	M	M	T
1	1	0	1		17	T	T	1	T	44	60	T	T	54	38	T	M	T	T	M
1	1	1	0		M	T	T	M	T	42	58	T	T	52	36	T	M	T	T	M
1	1	1	1		T	48	32	T	M	T	T	M	M	T	T	M	T	M	M	T

* – no errors detected
Number – the number of the single bit-in-error
T – two errors detected
M – more than two errors detected

TABLE XIV. DIAGNOSTIC LATCH LOADING – 64-BIT FORMAT.

Data Bit	Internal Function
0	Diagnostic Check Bit X
1	Diagnostic Check Bit 0
2	Diagnostic Check Bit 1
3	Diagnostic Check Bit 2
4	Diagnostic Check Bit 4
5	Diagnostic Check Bit 8
6, 7	Don't Care
8	Slice 0/1 – CODE ID 0
9	Slice 0/1 – CODE ID 1
10	Slice 0/1 – CODE ID 2
11	Slice 0/1 – DIAG MODE 0
12	Slice 0/1 – DIAG MODE 1
13	Slice 0/1 – CORRECT
14	Slice 0/1 – PASS THRU
15	Don't Care
16-23	Don't Care
24	Slice 2/3 – CODE ID 0
25	Slice 2/3 – CODE ID 1
26	Slice 2/3 – CODE ID 2
27	Slice 2/3 – DIAG MODE 0
28	Slice 2/3 – DIAG MODE 1
29	Slice 2/3 – CORRECT
30	Slice 2/3 – PASS THRU

Data Bit	Internal Function
31	Don't Care
32-37	Don't Care
38	Diagnostic Check Bit 16
39	Don't Care
40	Slice 4/5 – CODE ID 0
41	Slice 4/5 – CODE ID 1
42	Slice 4/5 – CODE ID 2
43	Slice 4/5 – DIAG MODE 0
44	Slice 4/5 – DIAG MODE 1
45	Slice 4/5 – CORRECT
46	Slice 4/5 – PASS THRU
47	Don't Care
48-54	Don't Care
55	Diagnostic Check Bit 32
56	Slice 6/7 – CODE ID 0
57	Slice 6/7 – CODE ID 1
58	Slice 6/7 – CODE ID 2
59	Slice 6/7 – DIAG MODE 0
60	Slice 6/7 – DIAG MODE 1
61	Slice 6/7 – CORRECT
62	Slice 6/7 – PASS THRU
63	Don't Care

MAXIMUM RATINGS (above which the useful life may be impaired)

Storage Temperature	−65 to +150°C
Temperature (Case) Under Bias	−55 to +125°C
Supply Voltage to Ground Potential	−0.5V to +7.0V
DC Voltage Applied to Outputs for high Output State	−0.5V to V_{CC} max.
DC Input Voltage	−0.5 to +5.5V
DC Output Current, Into Outputs	30mA
DC Input Current	−30 to +5.0 mA

OPERATING RANGE

P/N	Range	Temperature	V_{CC}	
Am2960DC, XC	COM'L	T_A = 0 to +70°C	V_{CC} = 5.0V ±5%	(MIN = 4.75V, MAX = 5.25V)
Am2960DM, FM, XM	MIL	T_C = −55 to +125°C	V_{CC} = 5.0V ± 10%	(MIN = 4.50V, MAX = 5.50V)

DC CHARACTERISTICS

Parameters	Description	Test Conditions (Note 1)			Min	Typ (Note 2)	Max	Units
V_{OH}	Output HIGH Voltage	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL}	I_{OH} = −0.8mA	COM'L	2.7			Volts
				MIL	2.4			
V_{OL}	Output LOW Voltage	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL}	I_{OL} = 8mA				0.5	Volts
V_{IH}	Input HIGH Voltage	Guaranteed Input Logical HIGH Voltage for all Inputs (Note 7)			2.0			Volts
V_{IL}	Input LOW Voltage	Guaranteed Input Logical LOW Voltage for all Inputs (Note 7)					0.8	Volts
V_I	Input Clamp Voltage	V_{CC} = MIN, I_{IN} = −18mA					−1.5	Volts
I_{IL}	Input LOW Current	V_{CC} = MAX, V_{IN} = 0.5V	$DATA_{0-15}$				−410	μA
			All Other Inputs				−360	
I_{IH}	Input HIGH Current	V_{CC} = MAX, V_{IN} = 2.7V	$DATA_{0-15}$				70	μA
			All Other Inputs				50	
I_I	Input HIGH Current	V_{CC} = MAX, V_{IN} = 5.5V					1.0	mA
I_{OZH} I_{OZL}	Off State (High Impedance) Output Current	V_{CC} MAX	$DATA_{0-15}$	V_O = 2.4			70	μA
				V_O = 0.5			−410	
			SC_{0-6}	V_O = 2.4			50	
				V_O = 0.5			−50	
I_{OS}	Output Short Circuit Current (Note 3)	V_{CC} = V_{CC} MAX +0.5V, V_O = 0.5V			−25		−85	mA
I_{CC}	Power Supply Current (Note 6)	V_{CC} = MAX		T_A = 25°C		275	390	mA
			COM'L	T_A = 0 to +70°C			400	
				T_A = +70°C			365	
			MIL	T_C = −55 to +125°C			400	
				T_C = +125°C			345	

Notes: 1. For conditions shown as MIN or MAX, use the appropriate value specified under Electrical Characteristics for the applicable device type.
2. Typical limits are at V_{CC} = 5.0V, 25°C ambient and maximum loading.
3. Not more than one output should be shorted at a time. Duration of the short circuit test should not exceed one second.
4. These are three-state outputs internally connected to TTL inputs. Input Characteristics are measured with output enables HIGH.
5. "MIL" = Am2960XM, DM, FM. "COM'L" = Am2960XC, DC.
6. Worst case I_{CC} is at minimum temperature.
7. These input levels provide zero noise immunity and should only be tested in a static, noise-free environment.

Notes on Testing

Incoming test procedu
ned, taking into accou
part. The following no

1. Insure the part is ac
changes in V_{CC} c
erroneous function
2. Do not leave inputs
to oscillate at high
3. Do not attempt to
Following an input
as much as 400mA

Notes on Testing

Incoming test procedures on this device should be carefully planned, taking into account the complexity and power levels of the part. The following notes may be useful.

1. Insure the part is adequately decoupled at the test head. Large changes in V_{CC} current as the device switches may cause erroneous function failures due to V_{CC} changes.
2. Do not leave inputs floating during any tests, as they may start to oscillate at high frequency.
3. Do not attempt to perform threshold tests at high speed. Following an input transition, ground current may change by as much as 400mA in 5-8ns. Inductance in the ground cable may allow the ground pin at the device to rise by 100's of millivolts momentarily.
4. Use extreme care in defining input levels for AC tests. Many inputs may be changed at once, so there will be significant noise at the device pins and they may not actually reach V_{IL} or V_{IH} until the noise has settled. AMD recommends using $V_{IL} \leq 0.4V$ and $V_{IH} \geq 2.4V$ for AC tests.
5. To simplify failure analysis, programs should be designed to perform DC, Function, and AC tests as three distinct groups of tests.
6. To assist in testing, AMD offers documentation on our test procedures and, in most cases, can provide Fairchild Sentry programs, under license.

1. Am2960 Guaranteed Commercial Range Performance

The tables below specify the guaranteed performance of the Am2960 over the commercial operating range of 0 to +70°C, with V_{CC} from 4.75V to 5.25V. All data are in ns, with inputs switching between 0V and 3V at 1V/ns and measurements made at 1.5V. All outputs have maximum DC load.

This data applies to the following part numbers: Am2960DC, XC.

A. Combinational Propagation Delays
C_L = 50pF

From Input \ To Output	SC_{0-6}	$DATA_{0-15}$	$\overline{ERROR}$	$\overline{MULT\ ERROR}$
$DATA_{0-15}$	32	65*	32	50
CB_{0-6} ($CODE\ ID_{2-0}$ 000, 011)	28	56	29	47
CB_{0-6} ($CODE\ ID_{2-0}$ 010, 100, 101, 110, 111)	28	45	29	34
$\overline{GENERATE}$	35	63	36	55
CORRECT (Not Internal Control Mode)	–	45	–	–
DIAG MODE (Not Internal Control Mode)	50	78	59	75
PASS THRU (Not Internal Control Mode)	36	44	29	46
$CODE\ ID_{2-0}$	61	90	60	80
LE IN (From latched to transparent)	39	72*	39	59
LE OUT (From latched to transparent)	–	31	–	–
LE DIAG (From latched to transparent; Not Internal Control Mode)	45	78	45	65
Internal Control Mode: LE DIAG (From latched to transparent)	67	96	66	86
Internal Control Mode: $DATA_{0-15}$ (Via Diagnostic Latch)	67	96	66	86

*Data In (or LE In) to Correct Data Out measurement requires timing as shown in Figure D opposite.

B. Set-up and H

From Input	(L U
$DATA_{0-15}$	L
CB_{0-6}	L
$DATA_{0-15}$	L
CB_{0-6} (CODE ID 000, 011)	L
CB_{0-6} (CODE ID 010, 100, 101, 110, 111)	L
$\overline{GENERATE}$	L
CORRECT	L
DIAG MODE	L
PASS THRU	L
$CODE\ ID_{2-0}$	L
LE IN	L
$DATA_{0-15}$	L

$DATA_{0-15}$

LE IN

$\overline{OE}$ BYTE 0 & 1

B. Set-up and Hold Times Relative to Latch Enables

From Input	To (Latching Up Data)	Set-up Time	Hold Time
$DATA_{0-15}$	LE IN	6	7
CB_{0-6}	LE IN	5	6
$DATA_{0-15}$	LE OUT	44	5
CB_{0-6} (CODE ID 000, 011)	LE OUT	35	0
CB_{0-6} (CODE ID 010, 100, 101, 110, 111)	LE OUT	27	0
$\overline{GENERATE}$	LE OUT	42	0
CORRECT	LE OUT	26	1
DIAG MODE	LE OUT	69	0
PASS THRU	LE OUT	26	0
CODE ID_{2-0}	LE OUT	81	0
LE IN	LE OUT	51	5
$DATA_{0-15}$	LE DIAG	6	8

C. Output Enable/Disable Times

Output disable tests performed with C_L = 5pF and measured to 0.5V change of output voltage level.

Input	Output	Enable	Disable
$\overline{OE}$ BYTE 0, $\overline{OE}$ BYTE 1	$DATA_{0-15}$	30	30
$\overline{OE}$ SC	SC_{0-6}	30	30

D. Minimum Pulse Widths

LE IN, LE OUT, LE DIAG	15

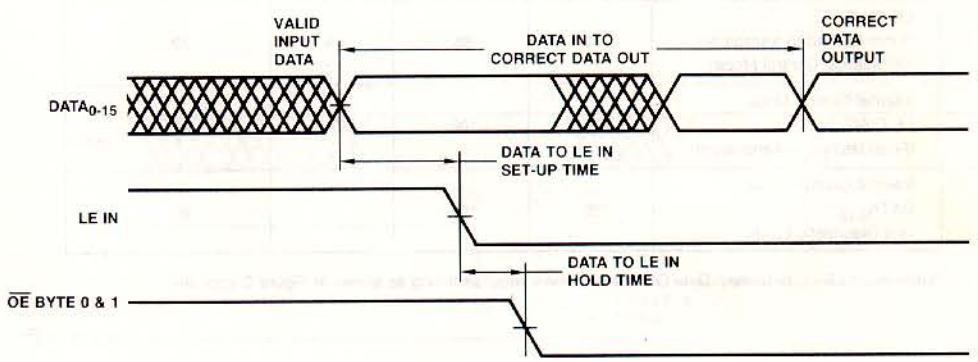


Figure D.

1. Am2960 Guaranteed Military Range Performance

The tables below specify the guaranteed performance of the Am2960 over the military operating range of −55 to +125°C case temperature, with V_{CC} from 4.5V to 5.5V. All data are in ns, with inputs switching between 0V and 3V at 1V/ns and measurements made at 1.5V. All outputs have maximum DC load.

This data applies to the following part numbers: Am2960DM, FM, XM.

A. Combinational Propagation Delays
C_L = 50pF

From Input \ To Output	SC_{0-6}	$DATA_{0-15}$	$\overline{ERROR}$	$\overline{MULT\ ERROR}$
$DATA_{0-15}$	35	73*	36	56
CB_{0-6} ($CODE\ ID_{2-0}$ 000, 011)	30	61	31	50
CB_{0-6} ($CODE\ ID_{2-0}$ 010, 100, 101, 110, 111)	30	50	31	37
$\overline{GENERATE}$	38	69	41	62
CORRECT (Not Internal Control Mode)	–	49	–	–
DIAG MODE (Not Internal Control Mode)	58	89	65	90
PASS THRU (Not Internal Control Mode)	39	51	34	54
$CODE\ ID_{2-0}$	69	100	68	90
LE IN (From latched to transparent)	44	82*	43	66
LE OUT (From latched to transparent)	–	33	–	–
LE DIAG (From latched to transparent; Not Internal Control Mode)	50	88	49	72
Internal Control Mode: LE DIAG (From latched to transparent)	75	106	74	96
Internal Control Mode: $DATA_{0-15}$ (Via Diagnostic Latch)	75	106	74	96

*Data In (or LE In) to Correct Data Out measurement requires timing as shown in Figure D opposite.

B. Set-up and Hol

From Input	(La Up
$DATA_{0-15}$	LE
CB_{0-6}	LE
$DATA_{0-15}$	LE
CB_{0-6} (CODE ID 000, 011)	LE
CB_{0-6} (CODE ID 010, 100, 101, 110, 111)	LE
$\overline{GENERATE}$	LE
CORRECT	LE
DIAG MODE	LE
PASS THRU	LE
$CODE\ ID_{2-0}$	LE
LE IN	LE
$DATA_{0-15}$	LE

$DATA_{0-15}$

LE IN

$\overline{OE}$ BYTE 0 & 1

B. Set-up and Hold Times Relative to Latch Enables

From Input	To (Latching Up Data)	Set-up Time	Hold Time
$DATA_{0-15}$	LE IN	7	7
CB_{0-6}	LE IN	5	7
$DATA_{0-15}$	LE OUT	50	5
CB_{0-6} (CODE ID 000, 011)	LE OUT	38	0
CB_{0-6} (CODE ID 010, 100, 101, 110, 111)	LE OUT	30	0
$\overline{GENERATE}$	LE OUT	46	0
CORRECT	LE OUT	28	1
DIAG MODE	LE OUT	84	0
PASS THRU	LE OUT	30	0
CODE ID_{2-0}	LE OUT	89	0
LE IN	LE OUT	59	5
$DATA_{0-15}$	LE DIAG	7	9

C. Output Enable/Disable Times

Output disable tests performed with C_L = 5pF and measured to 0.5V change of output voltage level.

Input	Output	Enable	Disable
$\overline{OE}$ BYTE 0, $\overline{OE}$ BYTE 1	$DATA_{0-15}$	35	35
$\overline{OE}$ SC	SC_{0-6}	35	35

D. Minimum Pulse Widths

LE IN, LE OUT, LE DIAG	15

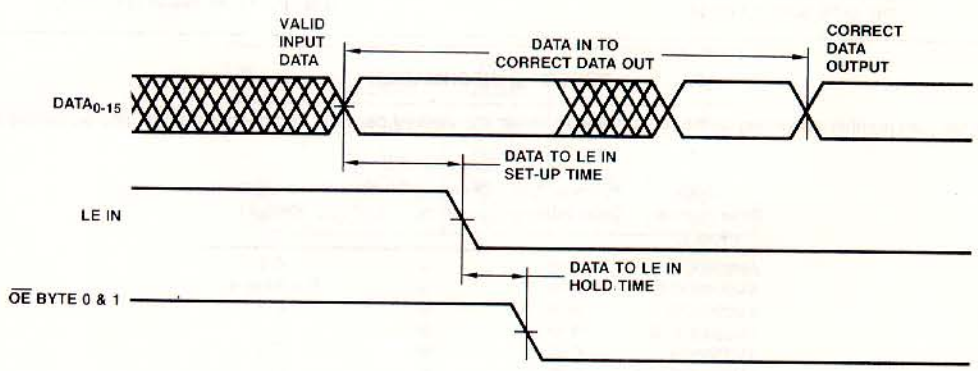
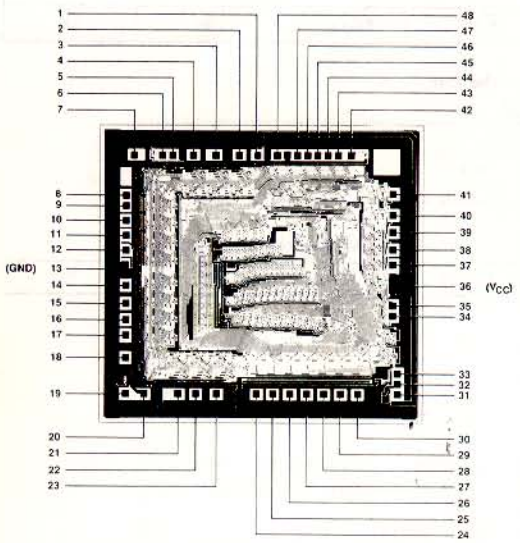


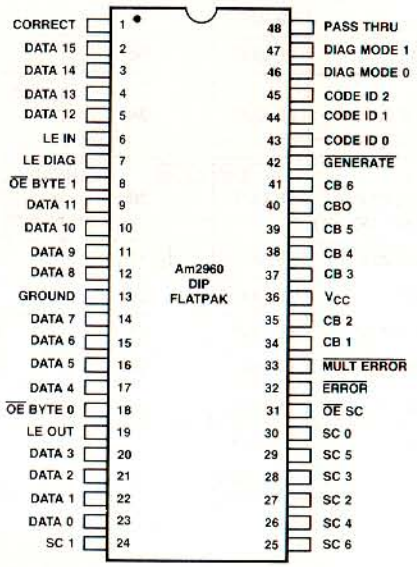
Figure D.

METALLIZATION AND PAD LAYOUT



DIE SIZE: 0.200" X 0.183"

CONNECTION DIAGRAM
Top View



Note: Pin 1 is marked for orientation.

ORDERING INFORMATION

Order the part number according to the table below to obtain the desired package, temperature range, and screening level.

Am2960 Order Number (Note 3)	Package Type Order Number	Operating Range (Note 1)	Screening Level (Note 2)
AM2960DC	D-48	C	C-1
AM2960DC-B	D-48	C	B-2 (Note 4)
AM2960DM	D-48	M	C-3
AM2960DM-B	D-48	M	B-3
AM2960FM	F-48	M	C-3
AM2960FM-B	F-48	M	B-3
AM2960XC	Dice	C	Visual inspection to MIL-STD-883 Method 2010B.
AM2960XM	Dice	M	

Notes: 1. P = Molded DIP, D = Hermetic DIP, F = Flat Pak. Number following letter is number of leads. See Appendix B for detailed outline. Where Appendix B contains several dash numbers, any of the variations of the package may be used unless otherwise specified.
2. C = 0 to +70°C, V_{CC} = 4.75V to 5.25V, M = −55 to +125°C, V_{CC} = 4.50V to 5.50V.
3. See Appendix A for details of screening. Levels C-1 and C-3 conform to MIL-STD-883, Class C. Level B-3 conforms to MIL-STD-883, Class B.
4. 96 hour burn-in.

TEST OUTPUT LOAD CONFIGURATION FOR Am2960

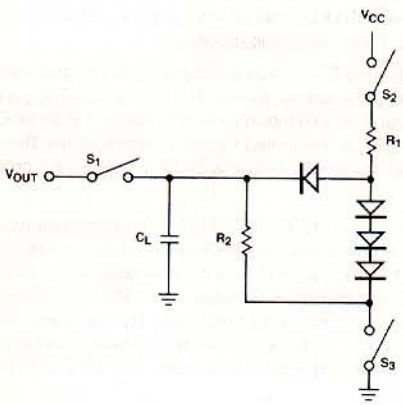


Figure 11. Three-State Outputs

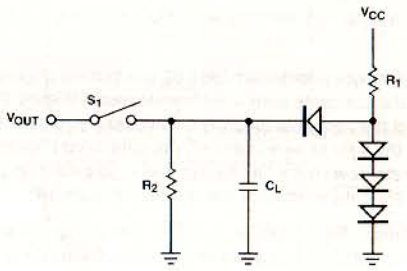


Figure 12. Normal Outputs

Notes:
1. C_L = 50pF includes scope probe, wiring and stray capacitances without device in test fixture.
2. S_1, S_2, S_3 are closed during function test and all A.C. tests, except output enable tests.
3. S_1 and S_3 are closed while S_2 is open for t_{PZH} test.
   S_1 and S_2 are closed while S_3 is open for t_{PZL} test.
4. R_2 = 1K for three-state output.
   R_2 is determined by the I_{OH} at V_{OH} = 2.4V for non-three-state outputs.
5. R_1 is determined by I_{OL} (MIL) with V_{CC} = 5.0V minus the current to ground through R_2.
6. C_L = 5.0 pF for output disable tests.

TEST OUTPUT LOADS

Pin #	Pin Label	Test Circuit	R_1	R_2
–	D_0-D_{15}	Fig. 11	430Ω	1kΩ
24-30	SC_0-SC_6	Fig. 11	430Ω	1kΩ
32	$\overline{\text{ERROR}}$	Fig. 12	470Ω	3kΩ
33	$\overline{\text{MULTERROR}}$	Fig. 12	470Ω	3kΩ

For additional information on testing, see section "Guidelines on Testing Am2900 Family Devices."

APPLICATIONS

Byte Write

Byte operations are increasingly common for 16 and 32-bit processors. These complicate memory operations because check bits are generated for a complete 16 or 32 or 64-bit memory word, not for a single byte.

To write a byte into memory with EDC requires the following steps:

- Latch the byte into the Am2961/62 bus buffers (Figure 13)
- Read the complete data word from memory (Figure 13)
- Correct the complete data word if necessary (Figure 13)
- Insert the byte to be written into the data word (Figure 14)
- Generate new check bits for the entire data word (Figure 14)
- Store the data word back into memory (Figure 14)

(In fact these steps must be taken for any piece of data being written into memory that is not as wide as a full memory word).

The Am2960 EDC is designed with the intent of keeping byte operations simple in EDC systems. The EDC has separate output enables for each byte in the Data Output Latch. As shown in figures 13 and 14, this allows the data word to be read from memory, the new byte to be inserted among the old, and new check bits to be generated using less time and less hardware than if separate byte enables were not available.

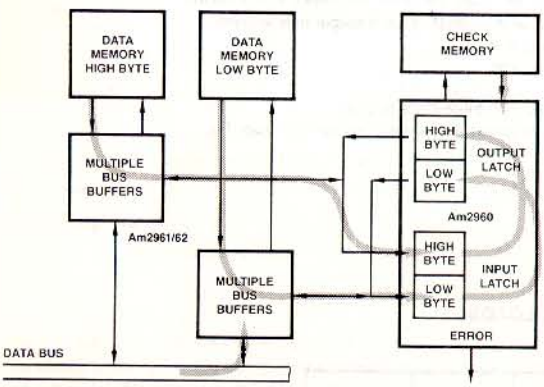


Figure 13. Byte Write, Phase 1: Read Out the Old Word and Correct

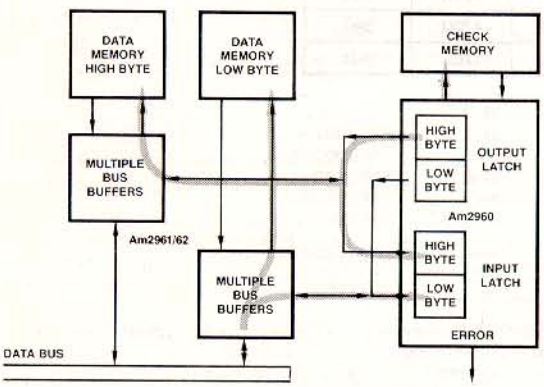


Figure 14. Byte Write, Phase 2: Insert the New Byte, Generate Checks and Write Into Memory

Diagnostics

EDC is used to boost the reliability of the overall system. It is necessary to also be able to check the operation of the EDC itself. For this reason the EDC has an internal control mode, a diagnostic latch, and two diagnostic modes.

To check that the EDC is functioning properly, the processor can put the EDC under software control by setting CODE ID_{2-0} to 001. This puts the EDC into Internal Control Mode. In Internal Control Mode the EDC is controlled by the contents of the Diagnostic Latch which is loaded from the DATA inputs under processor control.

The EDC is set into CORRECT Mode. The processor loads in a known set of check bits into the Diagnostic Latch, a known set of data bits into the Data In Latch, and forces data errors. The output of the EDC (syndromes, error flags, corrected data) is then compared against the expected responses. By exercising the EDC with a string of data/check combinations and comparing the output against the expected responses, the EDC can be fully checked out.

Eight Bit Data Word

Eight bit MOS microprocessors can use EDC too. Only five check bits are required. The EDC configuration for eight bits is shown in Figure 15. It operates as does the normal 16-bit configuration with the upper byte fixed at 0.

Check bit overhead for 8-bit data words can be reduced two ways. See the sections "Single Error Correction Only" and "Reducing Check Bit Overhead."

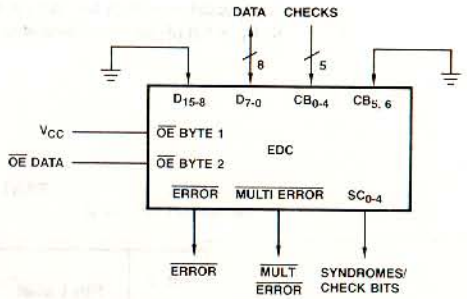


Figure 15. 8-Bit Configuration

Other Word Widths

EDC on data words other than 8, 16, 32, of 64 bits can be accomplished with the Am2960. In most cases the extra data bits can be forced to a constant, and EDC will procede as normal. For example a 24-bit data word is shown in Figure 16.

Single Error Correction Only

The EDC normally corrects all single bit errors *and* detects all double bit and some triple bit errors. To save one check bit per word the ability to detect double bit errors can be sacrificed – single errors are still detected and corrected.

Data Bits	Check Bits Required	
	Single Error Correction Only	Single Error Correct & Double Error Detect
8	4	5
16	5	6
32	6	7
64	7	8

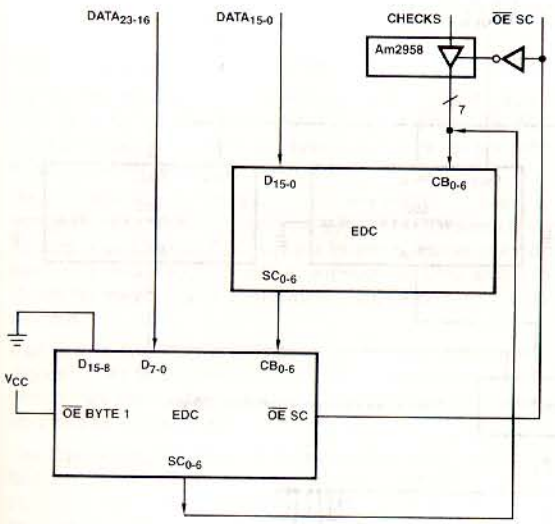


Figure 16. 24-Bit Configuration

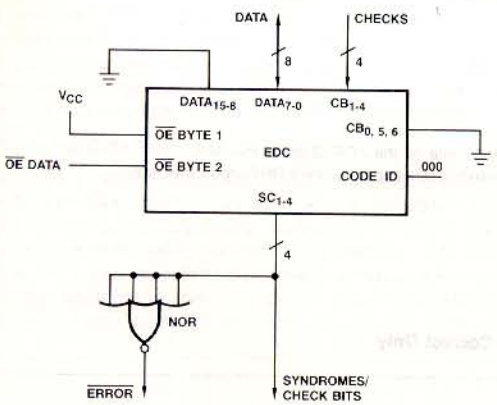


Figure 17. 8-Bit Single Error Correction Only

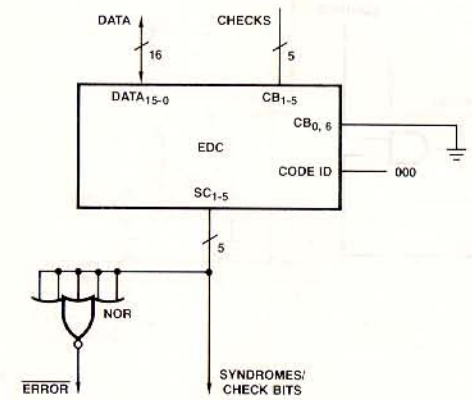
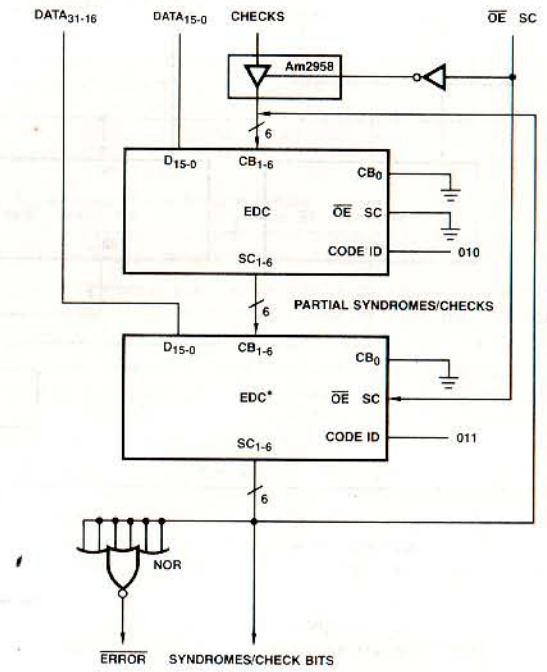


Figure 18. 16-Bit Single Error Correction Only



*The Code ID Combination for this Slice Forces the Check Bit Latch Transparent.

Figure 19. 32-Bit Single Correct Only

Figures 17, 18, 19, 20 show single error correction only configurations for 8, 16, 32, and 64-bit data words respectively.

Check Bit Correction

The EDC detects single bit errors whether the error is a data bit or a check bit. Data bit errors are automatically corrected by the EDC. To generate corrected check bits once a single check bit error is detected, the EDC need only be switched to GENERATE mode (data in the DATA INPUT LATCH is valid).

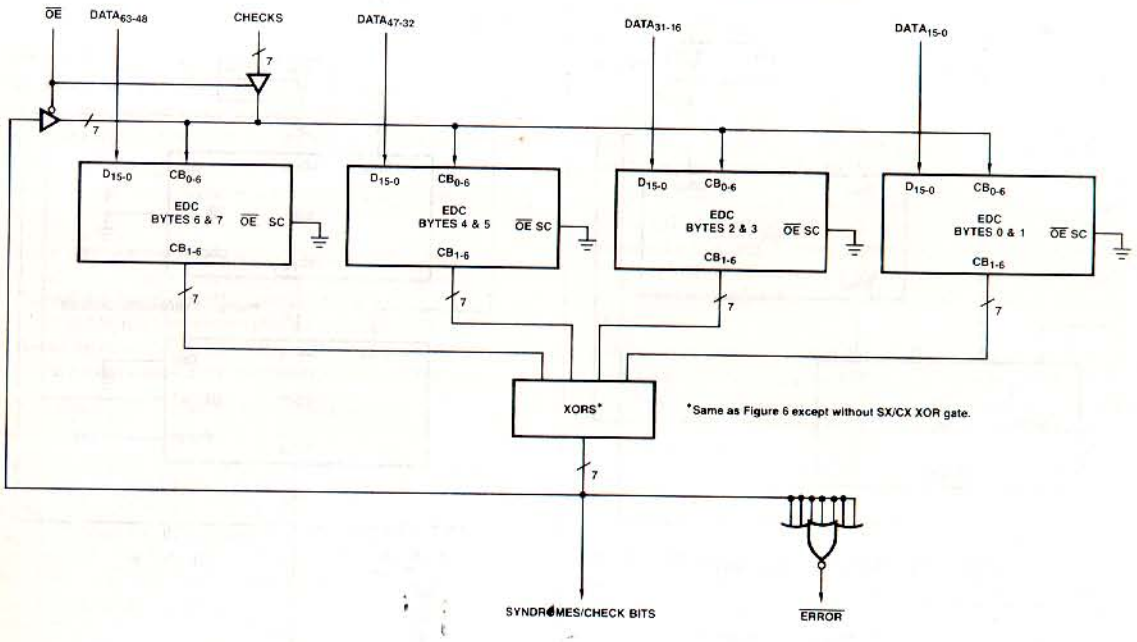
The syndromes generated by the EDC may be decoded to determine whether the single bit error is a check bit.

In many memory systems, a check bit error will be ignored on the memory read and corrected during a periodic "scrubbing" of memory (see section in System Design Considerations).

Multiple Errors

The bit-in-error decode logic uses syndrome bits S0 through S32 to correct errors, SX is only used in developing the multiple error signal. This means that some multiple errors will cause a data bit to be inverted.

For example, in the 16-bit mode if data bits 8 and 13 are in error the syndrome 111100 (SX, S0, S1, S2, S4, S8) is produced. This is flagged a double error by the error detection logic, but the bit-in-error decoder only receives syndrome 11100 (S0, S1, S2, S4, S8) which it decodes as a single error in data bit 0 and inverts that bit. If it is desired to inhibit this inversion, the multiple error output may be connected to the correct input as in Figure 21. This will inhibit correction when a multiple error occurs. Extra time delay may be introduced in the data to correct data path when this is done.



Notes: 1. In Pass Thru Mode the Contents of the Check Latch Appear on the XOR Outputs Inverted.
2. In Diagnostic Generate Mode the Contents of the Diagnostic Latch appear on the XOR Outputs Inverted.

Figure 20. 64-Bit Single Correct Only

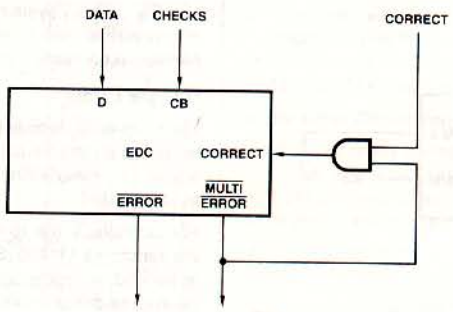


Figure 21. Inhibition of Data Modification

SYSTEM DESIGN CONSIDERATIONS

High Performance Parallel Operation

For maximum memory system performance the EDC should be used in the Check-Only configuration shown in Figure 22. With this configuration the memory system operates as fast with EDC as it would without.

On reads from memory, data is read out from the RAMs directly to the data bus (same as in a non-EDC system). At the same time, the data is read into the EDC to check for errors. If an error exists the EDC's error flags are used to interrupt the CPU and/or to stretch the memory cycle. If no error is detected, no slowdown is required.

If an error is detected, the EDC generates corrected data for the processor. At the designer's option the correct data may be written back into memory; error logging and diagnostic routines may also be run under processor control.

The Check-Only configuration allows data reads to proceed as fast with EDC as without. Only if an error is detected is there any slowdown. But even if the memory system had an error every hour this would mean only one error every 3-4 *billion* memory cycles. So even with a very high error rate, EDC in a Check-Only configuration has essentially zero impact on memory system speed.

On writes to memory, check bits must be generated before the full memory word can be written into memory. But using the Am2961/62 Data Bus Buffers allows the data word to be buffered on the memory board while check bits are generated. This makes the check bit generate time transparent to the processor.

EDC in the Data Path

The simplest configuration for EDC is to have the EDC directly in the data path as shown in Figure 23 (Correct-Always Configuration). In this configuration data read from memory is always corrected prior to putting the data on the data bus. The advantages are simpler operation and no need for mid-cycle interrupts. The disadvantage is that memory system speed is slowed by the amount of time it takes for error correction on ever cycle.

Usually the Correct-Always Configuration will be used with MOS microprocessors which have ample memory timing budgets. Most high performance processors will use the high performance parallel configuration shown in Figure 22. (Check-Only Configuration).

Scrubbing Avoids Double Errors

Single-bit errors are by far the most common in a memory system and are always correctable by the EDC.

Double bit memory errors are far less frequent than single bit (50 to 1, or 100 to 1) and are always detected by the EDC but not corrected.

In a memory system, soft errors occur only one at a time. A double bit error in a data word occurs when a single soft error is left uncorrected and is followed by another error in the data word hours, days, or weeks after the first.

"Scrubbing" memory periodically avoids almost all double-bit errors. In the scrubbing operation, every data word in memory is periodically checked by the EDC for single-bit errors. If one is found, it is corrected and the data word written back into memory. Errors are not allowed to pile up and so most double-bit errors are avoided.

The scrubbing operation is generally done as a background routine when the memory is not being used by the processor.

If memory is scrubbed frequently, errors that are detected and corrected during processor accesses need not be immediately written back into memory. Instead the error will be corrected in memory during scrubbing. This reduces the time delay involved in a processor access of an incorrect memory word.

Correction of Double-Bit Errors

In some cases, double-bit memory errors can be corrected! This is possible when one of the two bit errors is a hard error.

When a double bit error is detected the data word should be checked to determine if one of the errors is a hard error. If so the

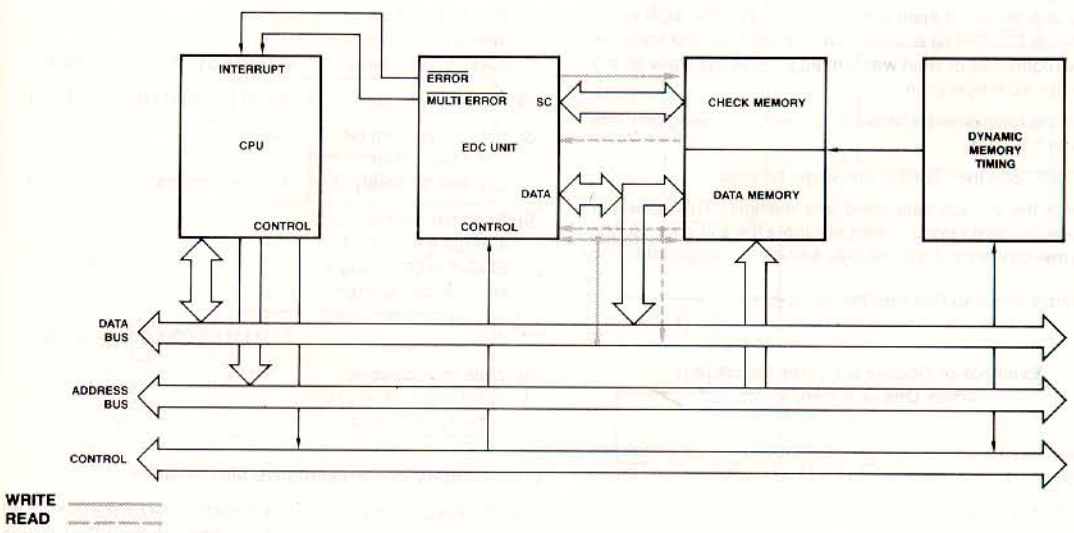


Figure 22. Check-Only Configuration

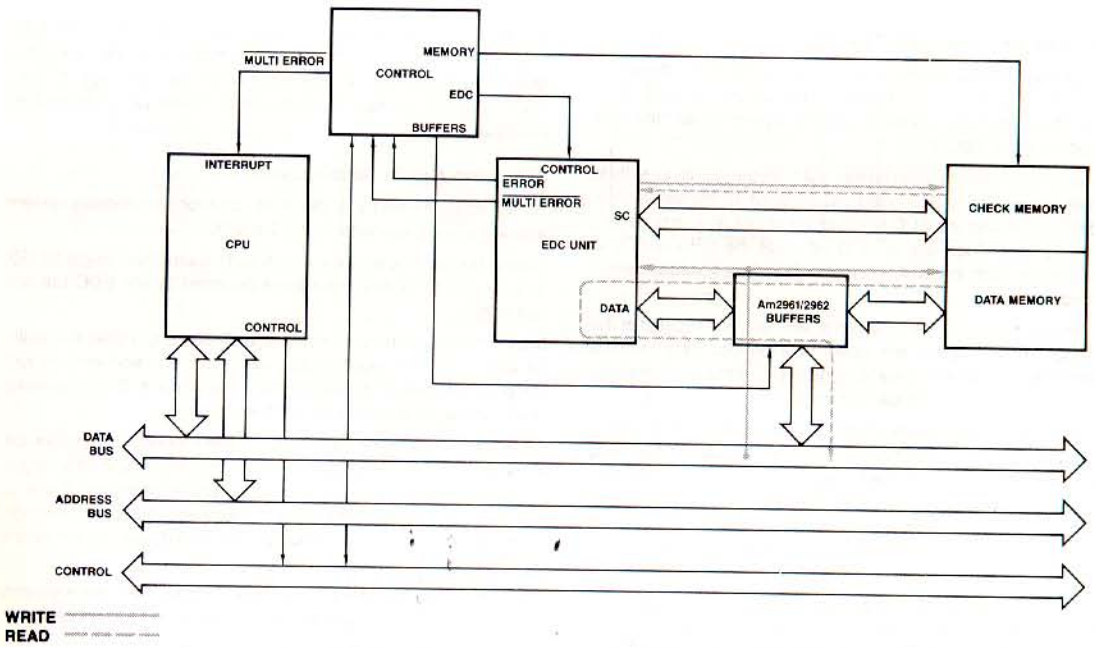


Figure 23. Correct-Always Configuration

hard error bit may be corrected by inverting it leaving only a single, correctable error. The time for this operation is negligible since it will occur infrequently.

The procedure after detection of a double error is as follows:

- Invert the data bits read from memory.
- Write the inverted data back into the same memory word.
- Re-read the memory location and XOR the newly read out value with the old. If there is no hard error then the XOR result will be all 1's. If there is a hard error, it will have the same bit value regardless of what was written in. So it will show as a 0 after the XOR operation
- Invert the hard error bit (this will "correct" it) leaving only one error in the data.
- The EDC can then correct the single bit error.
- Rewrite the correct data word into memory. This does not change the hard error but does eliminate the soft error. So the next memory access will find only a single-bit, correctable error.

An example helps to illustrate the procedure:

Example of Double Bit Error Correction When One is a Hard Error

1) Data Read from Memory (D_1)	16 data bits 1111111100000011	6 check bits 011010	
2) EDC detects a multiple error. Syndromes:		011000	
3) Syndrome decode indicates a double bit error.			
4) Invert the bits read from memory (D_1)	0000000011111100	100101	
5) Write D_1 back to the same memory location			
6) Read back the memory location (D_2)	0000000011111101	100101	
7) XOR D_1 and D_2	1111111111111110	111111	
8) So the last data bit is the hard error. Use this to modify D_1	1111111100000010	011010	
9) Pass the modified D_1 through the EDC. The EDC detects a single bit correctable error and outputs corrected data:	1111111100000000	011010	
10) Write the corrected data back to memory to fix the soft error.			

Error Logging and Preventative Maintenance

The effectiveness of preventative maintenance can be increased by logging information on errors detected by the EDC. This is called error logging.

The EDC provides sy
syndromes indicate wh
each individual RAM s
So the syndrome and c
in error.

Typically a permanent/
of time where the RAM
mittent, soft errors. Erro
increasing intermittent
replaced before a perr

Error logging also rec
RAMs. With EDC a ha
EDC always can correc
EDC can also correct d
soft (see "Correction of
continue operation des
need for emergency fi
can be instead replace
preventative maintena

Reducing Check Bit

Memory word widths ne
of the processor. The
overhead if wider mem

Memory	
# Data Bits	
8	
16	
32	
64	

Figur

The EDC provides syndromes when errors are detected. The syndromes indicate which bit is in error. In most memory systems, each individual RAM supplies only one bit of the memory word. So the syndrome and data word address specify which RAM was in error.

Typically a permanent/hard RAM failure is preceded by a period of time where the RAM displays an increasing frequency of intermittent, soft errors. Error logging statistic can be used to detect an increasing intermittent error frequency so that the RAM can be replaced before a permanent failure occurs.

Error logging also records the location of already hard failed RAMs. With EDC a hard failure will not halt system operation. EDC always can correct single bit errors even if it is a hard error. EDC can also correct double bit errors where one is hard and one soft (see "Correction of Double Bit Errors" Section). The ability to continue operation despite hard errors can greatly reduce the need for emergency field maintenance. The hard-failed RAMs can be instead replaced at low cost during a regularly scheduled preventative maintenance session.

Reducing Check Bit Overhead

Memory word widths need not be the same as the data word width of the processor. There is a substantial reduction in check bit overhead if wider memory words are used:

Memory Word		Check Bit Overhead
# Data Bits	# Check Bits	
8	5	38%
16	6	27%
32	7	14%
64	8	11%

This reduction in check bit overhead lowers cost and increases the amount of data that can be packed on to each board.

The trade off is that when writing data pieces into memory that are narrower than the memory word width, more steps are required. These steps are exactly the same as those described in Byte Write in the Applications section. No penalty exists for reads from memory.

EDC Per Board vs EDC Per System

The choice of an EDC per system or per board depends on the economics and the architecture of the system.

Certainly the cheaper approach is to have only one EDC per system and this is a viable solution if only one memory location is accessed at at time.

This solution does require that the system has both data and check bit lines (see Figure 25). This makes retrofitting a system difficult and creates complications if static or ROM memory, which do not require check bits, are mixed in with dynamic RAM.

If the system has an advanced architecture it is quite likely that it is necessary to simultaneously access memory locations on different memory boards (see Figure 24). Architectural features that require this are interleaved memory, cache memory, and DMA that is done simultaneously with processor memory accesses. EDC per board is a simpler system from a design standpoint.

The EDC is designed to work efficiently in either the per system or per board configurations.

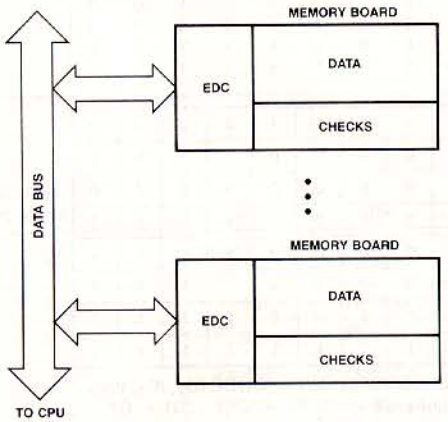


Figure 24. EDC Per Board

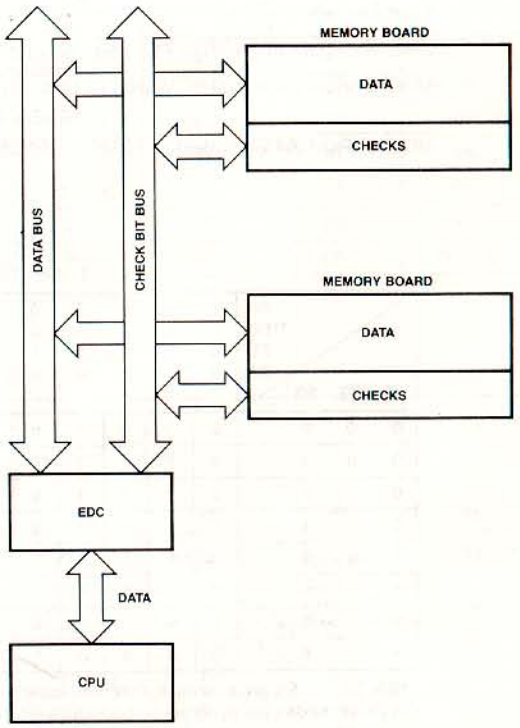


Figure 25. EDC Per System

FUNCTIONAL EQUATIONS

The following equations and tables describe in detail how the output values of the Am2960 EDC are determined as a function of the value of the inputs and the internal states. Be sure to carefully read the following definitions of symbols before examining the tables.

Definitions

D_i ← ($DATA_i$ if LE IN is HIGH or the output of bit i of the Data Input Latch if LE IN is LOW)
C_i ← (CB_i if LE IN is HIGH or the output of bit i of the Check Bit Latch if LE IN is LOW)
DL_i ← Output of bit i of the Diagnostic Latch
S_i ← Internally generated syndromes (same as outputs of SC_i if outputs enabled)

$PA \leftarrow D0 \oplus D1 \oplus D2 \oplus D4 \oplus D6 \oplus D8 \oplus D10 \oplus D12$
$PB \leftarrow D0 \oplus D1 \oplus D2 \oplus D3 \oplus D4 \oplus D5 \oplus D6 \oplus D7$
$PC \leftarrow D8 \oplus D9 \oplus D10 \oplus D11 \oplus D12 \oplus D13 \oplus D14 \oplus D15$
$PD \leftarrow D0 \oplus D3 \oplus D4 \oplus D7 \oplus D9 \oplus D10 \oplus D13 \oplus D15$
$PE \leftarrow D0 \oplus D1 \oplus D5 \oplus D6 \oplus D7 \oplus D11 \oplus D12 \oplus D13$
$PF \leftarrow D2 \oplus D3 \oplus D4 \oplus D5 \oplus D6 \oplus D7 \oplus D14 \oplus D15$
$PG_1 \leftarrow D0 \oplus D4 \oplus D6 \oplus D7$
$PG_2 \leftarrow D1 \oplus D2 \oplus D3 \oplus D5$
$PG_3 \leftarrow D8 \oplus D9 \oplus D11 \oplus D14$
$PG_4 \leftarrow D10 \oplus D12 \oplus D13 \oplus D15$

Error Signals

$$\overline{ERROR} \leftarrow \overline{(S6 \cdot (ID_1 + ID_2))} \cdot \overline{S5} \cdot \overline{S4} \cdot \overline{S3} \cdot \overline{S2} \cdot \overline{S1} \cdot \overline{S0} + GENERATE + INITIALIZE + PASSTHRU$$

$$\overline{MULT\ ERROR}\ (16\text{ and }32\text{-Bit Modes}) \leftarrow \overline{((\overline{S6 \cdot ID_1}) \oplus S5 \oplus S4 \oplus S3 \oplus S2 \oplus S1 \oplus S0)\,(\overline{ERROR})} + TOME + GENERATE + PASSTHRU + INITIALIZE$$

$$\overline{MULT\ ERROR}\ (64\text{-Bit Modes}) \leftarrow TOME + GENERATE + PASSTHRU + INITIALIZE$$

TOME (Three or More Errors)*

S1	S2	S3	S0	**S6	S5	S4
			0	0	0	0
			1	0	0	0
			0	1	0	0
			1	1	0	0
			0	0	1	0
			1	0	1	0
			0	1	1	0
			1	1	1	0
			0	0	0	1
			1	0	0	1
			0	1	0	1
			1	1	0	1
			0	0	1	1
			1	0	1	1
			0	1	1	1
			1	1	1	1
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	0	0	0	0
1	1	0	1	1	1	1
1	1	1	0	0	0	0

*S6, S5, . . . S0 are internal syndromes except in Modes 010, 100, 101, 110, 111 (CODE ID_2, ID_1, ID_0). In these modes the syndromes are input over the Check-Bit lines. S6 ← C6, S5 ← C5, . . . S1 ← C1, S0 ← C0.

**The S6 internal syndrome is always forced to 0 in CODE ID 000.

SC Outputs

Tables XV, XVI, XVII, generated in each con control mode not appli

GENERATE Mode (Check Bits)	CODE ID
SC_0 ←	
SC_1 ←	
SC_2 ←	
SC_3 ←	
SC_4 ←	
SC_5 ←	
SC_6 ←	

Detect and Correct Modes (Syndromes)	CODE ID
SC_0 ←	
SC_1 ←	
SC_2 ←	
SC_3 ←	
SC_4 ←	
SC_5 ←	
SC_6 ←	

*In CODE $ID_{2\text{-}0}$ 011 th

Diagnostic Read Mode	CODE ID
SC_0 ←	
SC_1 ←	
SC_2 ←	
SC_3 ←	
SC_4 ←	
SC_5 ←	
SC_6 ←	

*In CODE $ID_{2\text{-}0}$ 011 th

SC Outputs

Tables XV, XVI, XVII, XVIII, XIX show how outputs SC_{0-6} are generated in each control mode for various CODE IDs (internal control mode not applicable).

TABLE XV.

CODE ID_{2-0} / GENERATE Mode (Check Bits)	000	010	011	100	101	110	111
$SC_0 \leftarrow$	$PG_2 \oplus PG_3$	$PG_1 \oplus PG_3$	$PG_2 \oplus PG_4$	$PG_2 \oplus PG_3$	$PG_2 \oplus PG_3$	$PG_1 \oplus PG_4$	$PG_1 \oplus PG_4$
$SC_1 \leftarrow$	PA	PA	PA	PA	PA	PA	PA
$SC_2 \leftarrow$	$\overline{PD}$	$\overline{PD}$	PD	$\overline{PD}$	PD	PD	PD
$SC_3 \leftarrow$	$\overline{PE}$	$\overline{PE}$	PE	$\overline{PE}$	PE	PE	PE
$SC_4 \leftarrow$	PF	PF	PF	PF	PF	PF	PF
$SC_5 \leftarrow$	PC	PC	PC	PC	PC	PC	PC
$SC_6 \leftarrow$	1	PB	PC	PB	PB	PB	PB

TABLE XVI.

CODE ID_{2-0} / Detect and Correct Modes (Syndromes)	000	010	011*	100	101	110	111
$SC_0 \leftarrow$	$PG_2 \oplus PG_3 \oplus C0$	$PG_1 \oplus PG_3 \oplus C0$	$PG_2 \oplus PG_4 \oplus CB_0$	$PG_2 \oplus PG_3 \oplus C0$	$PG_2 \oplus PG_3$	$PG_1 \oplus PG_4$	$PG_1 \oplus PG_4$
$SC_1 \leftarrow$	$PA \oplus C1$	$PA \oplus C1$	$PA \oplus CB_1$	$PA \oplus C1$	PA	PA	PA
$SC_2 \leftarrow$	$\overline{PD} \oplus C2$	$\overline{PD} \oplus C2$	$PD \oplus CB_2$	$\overline{PD} \oplus C2$	PD	PD	PD
$SC_3 \leftarrow$	$\overline{PE} \oplus C3$	$\overline{PE} \oplus C3$	$PE \oplus CB_3$	$\overline{PE} \oplus C3$	PE	PE	PE
$SC_4 \leftarrow$	$PF \oplus C4$	$PF \oplus C4$	$PF \oplus CB_4$	$PF \oplus C4$	PF	PF	PF
$SC_5 \leftarrow$	$PC \oplus C5$	$PC \oplus C5$	$PC \oplus CB_5$	$PC \oplus C5$	PC	PC	PC
$SC_6 \leftarrow$	1	$PB \oplus C6$	$PC \oplus CB_6$	PB	PB	$PB \oplus C6$	$PB \oplus C6$

*In CODE ID_{2-0} 011 the Check-Bit Latch is forced transparent, the Data Latch operates normally.

TABLE XVII.

CODE ID_{2-0} / Diagnostic Read Mode	000	010	011*	100	101	110	111
$SC_0 \leftarrow$	$PG_2 \oplus PG_3 \oplus DL_0$	$PG_1 \oplus PG_3 \oplus DL_0$	$PG_2 \oplus PG_4 \oplus CB_0$	$PG_2 \oplus PG_3 \oplus DL_0$	$PG_2 \oplus PG_3$	$PG_1 \oplus PG_4$	$PG_1 \oplus PG_4$
$SC_1 \leftarrow$	$PA \oplus DL_1$	$PA \oplus DL_1$	$PA \oplus CB_1$	$PA \oplus DL_1$	PA	PA	PA
$SC_2 \leftarrow$	$\overline{PD} \oplus DL_2$	$\overline{PD} \oplus DL_2$	$PD \oplus CB_2$	$\overline{PD} \oplus DL_2$	PD	PD	PD
$SC_3 \leftarrow$	$\overline{PE} \oplus DL_3$	$\overline{PE} \oplus DL_3$	$PE \oplus CB_3$	$\overline{PE} \oplus DL_3$	PE	PE	PE
$SC_4 \leftarrow$	$PF \oplus DL_4$	$PF \oplus DL_4$	$PF \oplus CB_4$	$PF \oplus DL_4$	PF	PF	PF
$SC_5 \leftarrow$	$PC \oplus DL_5$	$PC \oplus DL_5$	$PC \oplus CB_5$	$PC \oplus DL_5$	PC	PC	PC
$SC_6 \leftarrow$	1	$PB \oplus DL_6$	$PC \oplus CB_6$	PB	PB	$PB \oplus DL_6$	$PB \oplus DL_7$

*In CODE ID_{2-0} 011 the Check-Bit Latch is forced transparent, the Data Latch operates normally.

TABLE XVIII.

Diagnostic Write Mode \ CODE ID_{2-0}	000	010	011*	100	101	110	111
$SC_0 \leftarrow$	DL_0	DL_0	CB_0	DL_0	1	1	1
$SC_1 \leftarrow$	DL_1	DL_1	CB_1	DL_1	1	1	1
$SC_2 \leftarrow$	DL_2	DL_2	CB_2	DL_2	1	1	1
$SC_3 \leftarrow$	DL_3	DL_3	CB_3	DL_3	1	1	1
$SC_4 \leftarrow$	DL_4	DL_4	CB_4	DL_4	1	1	1
$SC_5 \leftarrow$	DL_5	DL_5	CB_5	DL_5	1	1	1
$SC_6 \leftarrow$	1	DL_6	CB_6	1	1	DL_6	DL_7

*In CODE ID_{2-0} 011 the Check-Bit Latch is forced transparent; the Data Input Latch operates normally.

TABLE XIX.

PASS THRU Mode \ CODE ID_{2-0}	000	010	011*	100	101	110	111
$SC_0 \leftarrow$	C0	C0	CB_0	C0	1	1	1
$SC_1 \leftarrow$	C1	C1	CB_1	C1	1	1	1
$SC_2 \leftarrow$	C2	C2	CB_2	C2	1	1	1
$SC_3 \leftarrow$	C3	C3	CB_3	C_3	1	1	1
$SC_4 \leftarrow$	C4	C4	CB_4	C4	1	1	1
$SC_5 \leftarrow$	C5	C5	CB_5	C5	1	1	1
$SC_6 \leftarrow$	1	C6	CB_6	1	1	C6	C6

*In CODE ID_{2-0} 011 the Check-Bit Latch is forced transparent; the Data Input Latch operates normally.

Data Correction

Tables XX to XXVI shows which data output bits are corrected (inverted) depending upon the syndromes and the CODE ID position. Note that the syndromes that determine data correction are in some cases syndromes input externally via the CB inputs and in some cases syndromes generated internally by that EDC (S_i are the internal syndromes and are the same as the value of the SC_i output of that EDC if enabled).

The tables show the number of data bit inverted (corrected) if any for the CODE ID and syndrome combination.

TABLE XX. CODE ID_{2-0} = 000*

S2	S1 \ S5 S4 S3	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0	0	–	–	–	5	–	11	14	–
0	1	–	1	2	6	8	12	–	–
1	0	–	–	3	7	9	13	15	–
1	1	–	0	4	–	10	–	–	–

*Unlisted S combinations are no correction.

TABLE XXI. CODE ID_{2-0} = 010*

CB_2	CB_1 \ CB_6 CB_5 CB_4 CB_3	0 1 0 0	0 1 1 0	0 1 1 1	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1
0	0	–	11	14	–	–	–	–	5
0	1	8	12	–	–	–	1	2	6
1	0	9	13	15	–	–	–	3	7
1	1	10	–	–	–	–	0	4	–

*Unlisted CB combinations are no correction.

TABLE XXII. CODE ID_{2-0} = 011*

S6	0	0	0	0	1	1	1	1
S5	0	0	0	0	1	1	1	1
S4	0	0	1	1	0	0	1	1
S3	0	1	0	1	0	1	0	1
S2 S1								
0 0	–	–	–	5	–	11	14	–
0 1	–	1	2	6	8	12	–	–
1 0	–	–	3	7	9	13	15	–
1 1	–	0	4	–	10	–	–	–

*Unlisted S combinations are no correction.

TABLE XXIII. CODE ID_{2-0} = 100*

CB_0	0	0	0	0	1	1	1	1
CB_6	0	0	0	0	1	1	1	1
CB_5	1	1	1	1	0	0	0	0
CB_4	0	0	1	1	0	0	1	1
CB_3	0	1	0	1	0	1	0	1
CB_2 CB_1								
0 0	–	11	14	–	–	–	–	5
0 1	8	12	–	–	–	1	2	6
1 0	9	13	15	–	–	–	3	7
1 1	10	–	–	–	–	0	4	–

*Unlisted CB combinations are no correction

TABLE XXIV. CODE ID_{2-0} = 101*

CB_0	0	0	0	0	1	1	1	1
CB_6	0	0	0	0	1	1	1	1
CB_5	0	0	0	0	1	1	1	1
CB_4	0	0	1	1	0	0	1	1
CB_3	0	1	0	1	0	1	0	1
CB_2 CB_1								
0 0	–	–	–	5	–	11	14	–
0 1	–	1	2	6	8	12	–	–
1 0	–	–	3	7	9	13	15	–
1 1	–	0	4	–	10	–	–	–

*Unlisted CB combinations are no correction.

TABLE XXV. CODE ID_{2-0} = 110*

CB_0	0	0	0	0	1	1	1	1
CB_6	1	1	1	1	0	0	0	0
CB_5	0	0	0	0	1	1	1	1
CB_4	0	0	1	1	0	0	1	1
CB_3	0	1	0	1	0	1	0	1
CB_2 CB_1								
0 0	–	–	–	5	–	11	14	–
0 1	–	1	2	6	8	12	–	–
1 0	–	–	3	7	9	13	15	–
1 1	–	0	4	–	10	–	–	–

*Unlisted CB combinations are no correction.

TABLE XXVI. CODE ID_{2-0} = 111*

CB_0	0	0	0	0	1	1	1	1
CB_6	1	1	1	1	0	0	0	0
CB_5	1	1	1	1	0	0	0	0
CB_4	0	0	1	1	0	0	1	1
CB_3	0	1	0	1	0	1	0	1
CB_2 CB_1								
0 0	–	11	14	–	–	–	–	5
0 1	8	12	–	–	–	1	2	6
1 0	9	13	15	–	–	–	3	7
1 1	10	–	–	–	–	0	4	–

*Unlisted CB combinations are no correction.

ADVANCED MICRO DEVICES

TECHNICAL REPORT

Am2960 BOOSTS MEMORY RELIABILITY

ABSTRACT

Memory error frequency will increase due to the use of larger memory systems and the use of 16K and 64K RAMs, which are more susceptible to soft errors because of their smaller memory cell geometry.

At the same time, the need for reliability is increasing, both for the user and the system manufacturer. EDC (Error Detection and Correction) can reduce system downtime, can reduce field maintenance expenses and can provide manufacturers a marketing advantage due to increased reliability.

The Am2960 implements EDC using a modified Hamming code, and so boosts memory reliability by a factor of 60 or better. It slashes package count and adds initialization, byte-write and diagnostic features. It is fast and flexible enough to handle word widths from 8 to 64 bits.

The Am2960 is one of a series of Memory Support devices designed for use with dynamic MOS RAM memory systems.

Prepared by: Advanced Micro Devices, Bipolar Microprocessor.

Am2960 BOOSTS MEMORY RELIABILITY

The Am2960 is a 16-bit, expandable Error Detection and Correction (EDC) unit. It is used in conjunction with system main memories to boost memory reliability.

The Am2960 can correct *all* single-bit memory errors in a data word. It detects all double-bit errors and even some triple-bit errors. The gross error conditions of all 0s or all 1s are always detected.

Memory error and detection using the Am2960 boosts system reliability by a factor of 60 or better. System crashes will occur far less frequently and maintenance costs can be slashed.

MEMORY ERRORS

Memory Error Frequency

Memory errors are becoming *more* frequent due to two general trends:

1. Total system memory size is growing, and,
2. The geometry of individual memory cells in dynamic RAMs is shrinking, making them more susceptible to "soft" errors.

There are two basic types of memory errors. Hard errors are permanent physical failures of either the whole RAM, a row, a column, or a single bit. Hard errors are caused by power shorts, open leads, and various other factors. Initial testing and burn-in will reduce but not eliminate hard error failures in RAMs during system operation.

Soft errors are non-repeating, single-bit errors where there is no permanent damage. A soft error occurs when the charge state of a bit incorrectly shifts from 0 to 1 or from 1 to 0. This can be caused by system noise, pattern sensitivity, power surges[6] or alpha particles. The new 16K and 64K dynamic RAMs, with their smaller memory cell geometries are especially susceptible to soft errors induced by alpha particles. (The smaller the memory cell geometry, the less energy is required to cause the cell to change state.)

A paper given at Wescon, 1979[1] presented these failure rates for dynamic RAMs of increasing size (see Table 1). This table reflects only soft errors due to alpha particles.

Undetected Memory Failures are Expensive

Memory failures will occur in a system. When they do they will result either in a system crash or in loss of data integrity, unless memory error detection schemes are used. Either situation is expensive and inconvenient for the system users. Either situation can result in maintenance calls to the system manufacturer.

If the memory error occurs in an instruction word and the instruction is executed without being corrected, then a system crash will almost certainly occur. For example, an "Add" instruction could be changed to a "Jump" instruction with only a one-bit change – if the error is undetected, the jump would take place to essentially a random location.

If the error occurs in a word that is used for storing data, then data integrity is lost. In typical applications this could mean that bank account balances would be altered, blood diagnosis would be incorrect, or cooling water valves could be closed instead of opened.

System failures of any kind will often result in unscheduled maintenance requests to the system manufacturers. Maintenance calls are expensive for the system manufacturer and are to be avoided by preventative means if at all possible.

Strategies for Memory Errors

For reliability and maintenance cost reduction, memory errors must be dealt with by the system designer.

A common scheme is to use parity. But parity schemes cannot correct errors and can detect only single-bit errors. If a double-bit error occurs in a word, the parity is unchanged and so the error goes undetected. Parity cannot correct errors.

Error detection and correction (EDC) is implemented by the Am2960 using a modified Hamming code. The Hamming code scheme involves generating several check bits that contain enough redundant information to correct *all* single-bit errors and to detect all double-bit errors and some triple-bit errors. Also, the EDC modified Hamming code detects the gross error conditions of all 0s and all 1s.

Table 2 demonstrates that EDC is the superior strategy for both the system user and the system manufacturer.

TABLE 1. ERRORS ARE INCREASING.

Density Bits/Chip	Typical Error Rate (% per 1,000 Hours)	
	Soft*	Hard**
1K	.001	.0001
4K	.02	.002
16K	.10	.011
64K	.5***	.016

*Reflects alpha particles only. Does not include errors due to noise, power, patterns.
**After infant mortality.
***Based on initial customer evaluation.
Note: 0.1% per 1000 hours equals 1 failure in 10^6 hours.

TABLE 2. COMPARISON OF ERROR STRATEGIES.

Error Type	No Checking	Parity	EDC Using Am2960
Single-Bit Error	System crash.	System halt.	Correctable. System runs.
Double-Bit Error	System crash.	System crash.	System halt.
Entire RAM Failure	System crash.	System halt.	Correctable. System runs.

With EDC, the incidence of maintenance calls is significantly reduced. Even the failure of an entire RAM chip will not necessarily result in a system failure. Double-bit errors are not corrected but are detected so that the system may be halted and the user informed of a memory error and the exact location of it. A controlled system halt is far more desirable than an uncontrolled system crash.

EDC Improves MTTF

Error detection and correction as implemented on the Am2960 significantly improves the MTTF (mean time to failure) of memory systems.

A paper presented at Wescon, 1979[1] used the dynamic RAM error rates shown previously to calculate the following MTTFs for a 16 Megabyte system using 64K RAMs (see Table 3).

TABLE 3.

Error Type	MTTF*
Correctable Soft Error (Single-Bit)	13 days
Correctable Hard Error (Single-Bit)	110 days
Non-Correctable Soft Error (Double-Bit)	864 days
Non-Correctable Hard Error (Double-Bit)	7,021 days

*Based on 64K RAM alpha error rate of 0.13% per 1000 hours.

The MTTF improves by a factor of at least 60 with EDC. This improvement factor has been noted by others[2].

Another paper [3,4] calculated that with EDC, RAM errors would become a small factor in memory based failures relative to failures of other board components such as MSI, capacitors and resistors. The same paper[4] discusses how frequently preventative maintenance should be done so that a hard-failed RAM is replaced prior to a second RAM experiencing a hard-failure. The Am2960 has features that allow easy logging of data errors – this aids the maintenance engineer in quickly pinpointing hard-failed RAMs and RAMs displaying excessive soft error rates.

Memory Reliability is a Competitive Edge

EDC boosts memory reliability and gives you two competitive advantages:

1. Your system is more reliable.
2. Your field maintenance costs are reduced.

The demand for reliable system performance is increasing steadily. Reliability is a must for applications in aerospace, medical, banking, process control and on-line systems. Applications such as word-processors, small business systems and telecommunications also need memory reliability, as their users do not have the technical staff to handle system failures and are willing to pay for the convenience of smooth, error-free operation.

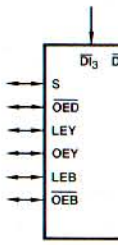
REFERENCES

1. Eric C. Westerfield, Four-Phase Systems, "Memory System Strategies for Soft and Hard Errors," Wescon, September, 1979.
2. "As Memory Density Quadruples Again, Designers Focus on Reliability," Electronic Design, January 18, 1979.
3. Robert Koppel, "RAM Reliability in Large Memory Systems – Significance of Predicting MTBF," Computer Design, February, 1979.
4. Robert Koppel, "RAM Reliability in Large Memory Systems – Improving MTBF with ECC," Computer Design, March, 1979.
5. A.V. Ferris-Prabhu, IBM General Technology Division, "Improving Memory Reliability Through Error Correction," Computer Design, July, 1979.
6. "Power-Line Disturbances Scorecard," Electronic Design, February 15, 1979.
7. "Alphas Stymie Statics," Electronics, March 15, 1979.
8. "Analyzing Computer Technology Costs," Computer Design, October, 1978.
9. "Alphas Cause Rift at ECC," Electronic Engineering Times, May 28, 1979.
10. Ernst L. Wall, ITT, "Applying the Hamming Code to Microprocessor-Based Systems," Electronics, November 22, 1979.

These devices are
also characterized
as:
AmZ8161
AmZ8162

DISTINCTIVE CHAR

- Quad high-speed
- Provides complete
  Error Detection a
  and dynamic RAM
- Three-state 24mA
- Three-state data o
- Inverting data bus
- Data bus latches a
- Space saving 24-p
- 100% MIL-STD-88



B-Bus

CO

S
OEY
$\overline{DO_0}$
$(B_0)\ \overline{B_0}$
Y_0
$\overline{DI_0}$
$\overline{DI_1}$
Y_1
$(B_1)\ \overline{B_1}$
$\overline{DO_1}$
LEY
GND

Note: F