EtherBox

Self-powered Ethernet Controller / Transceiver

==========

External Reference Specifications

==========

Sep 21, 1983

3Com Corporation

1. INTRODUCTION

The EtherBox is a very low cost, self-powered, outboard Ethernet interface for workstations, based on a VLSI Ethernet Data Link Controller IC (SEEQ DQ8001 EDLC), and a proven 3Com transceiver. It conforms to the Ethernet Specification, Version 1.0, 30 September 1980, as published by DEC, Intel and Xerox. With minor exceptions, it also implements Version 2.0, and completely implements Levels 2 AND 1 of the OSI specification, including:

Level 1 Functions

- Coax/station electrical isolation.
- Bit transmission/reception.
- Carrier sense.
- Transmit collision detection.
- Encoding/decoding.
- Preamble generation/removal

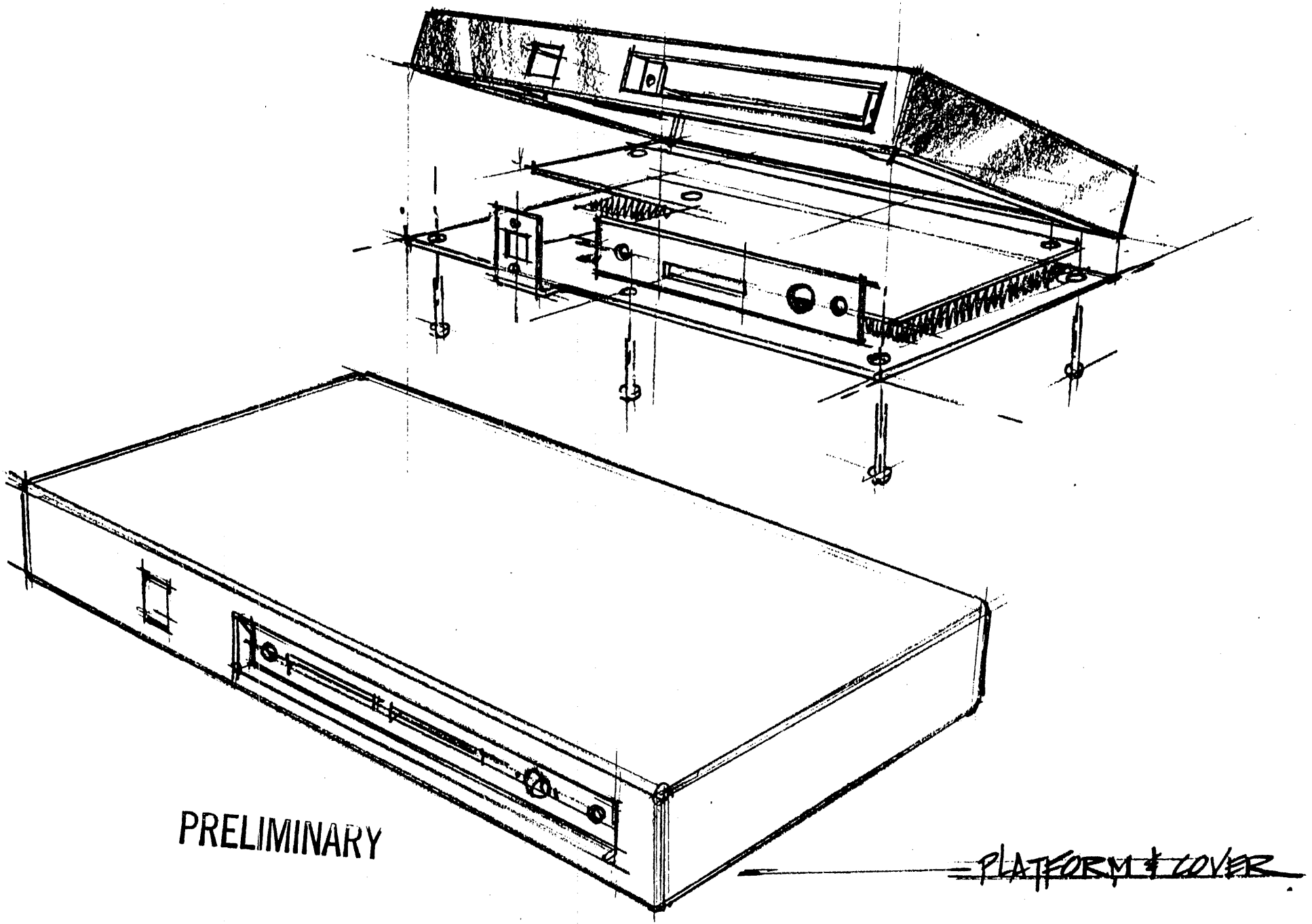
Level 2 Functions

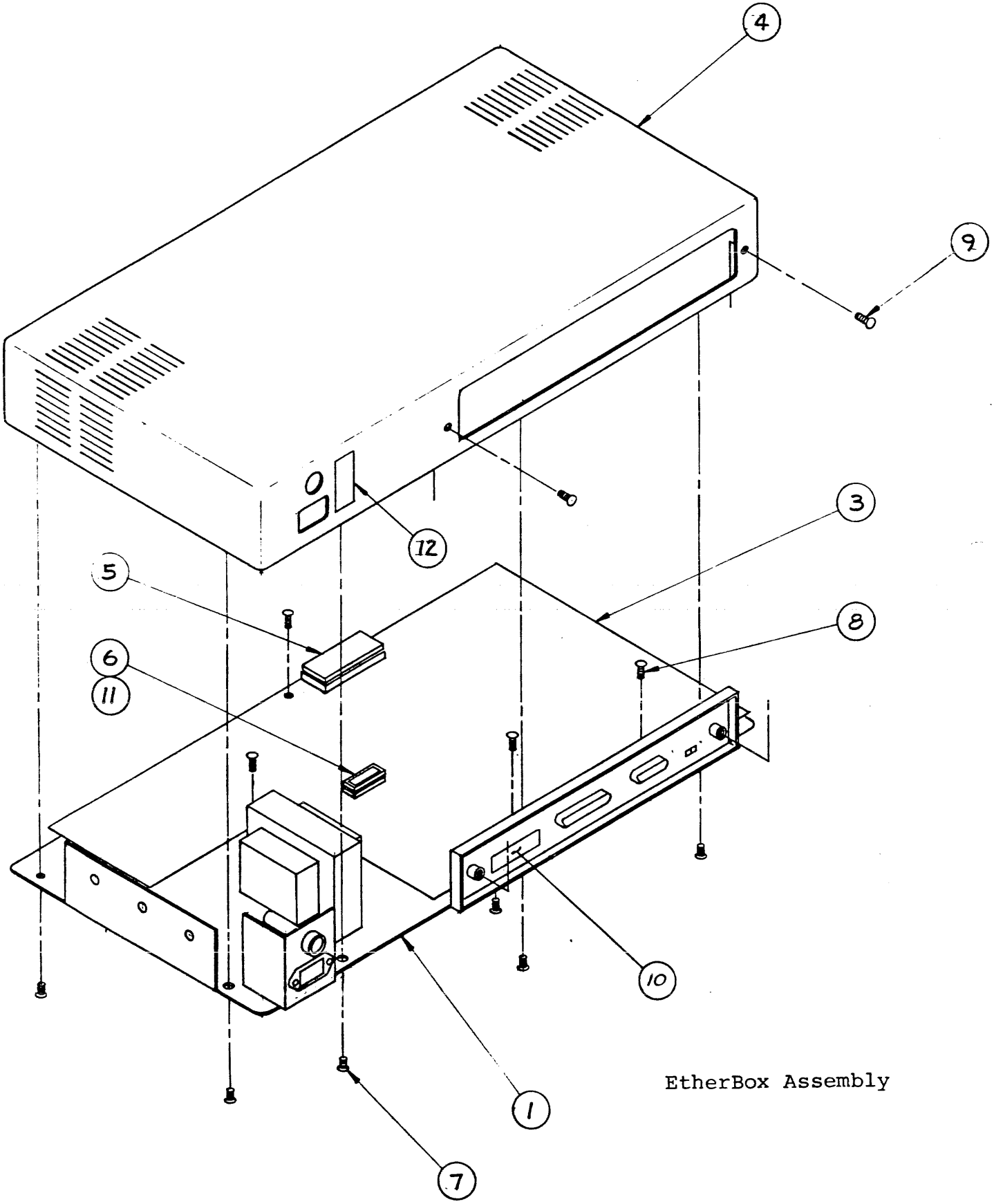
- CRC generation/checking.
- Carrier deference.
- Transmit collision enforcement.
- Collision fragment (runt) filtering.
- Bad packet filtering.
- Address recognition.

So in addition to a traditonal Ethernet controller which implements Level 2 (and a small part of Level 1), it contains an Ethernet transceiver to provide complete Level 1 and 2 functionality in one self-powered outboard box. The EtherBox contains three 2KByte packet buffers and is connected to the host system via a Centronics-style, bidirectional, 8-bit parallel interface. It also has provisions for external loopback.

Because of the integral transceiver, the EtherBox provides both a standard (DA15 connectored) Ethernet outlet and a means for direct connection to an Ethernet coax. The direct connection is in the form of a single BNC connector designed for an RG-58 "ether" attached to the EtherBox by a BNC "T". (The RG-58 coax can be compatibly attached to a standard, yellow Ethernet coax simply through an BNC/N-series adapter.)

[In the following, the term "controller" is used to refer to the Level 2 functions only OR to refer to the EtherBox as a whole, i.e., including the transceiver. The particular sense should be clear from the context. Also "system" refers to the parallel interface and/or the host computer.]


PRELIMINARY
PLATFORM & COVER



EtherBox Assembly

2. ARCHITECTURE

The EtherBox is fully buffered with three 2KByte packet buffers that are permanently assigned; two are assigned to receive and one to transmit. Once a packet is transferred to the transmit buffer for transmission and a transmit initiated, all Ethernet management is handled automatically by the EtherBox. Operation is similarly automatic for receive, with selective enabling of several address recognition modes. When enabled for multicast recognition, address screening of multicast packets must be performed by the system. Two receive buffers permit reception of back-to-back packets.

Connection to the EtherBox is effected entirely through 16 byte-wide registers. (Register selection and register data are multiplexed over the single data/command path of the parallel interface.) These registers are used to write commands, read status information, and to read/write packet data. The packet buffers are accessible through three of these registers. The accessible location for the single transmit buffer is specified by a separate register pair, the Transmit Buffer Pointer. For the two receive buffers, the Bus Pointer specifies the buffer location.

The packet buffers are large enough for any legal size Ethernet packet. Buffer access is given to either the system, meaning the parallel port, or the EtherBox Controller. A buffer switch exists for each buffer. If the switch is reset, the system has read/ write access to the buffer. Once the buffer switch is set, then only the controller has read/write access to the buffer. The controller will automatically reset the buffer switch after a sucessful transmission or reception. Transmit packets are end-aligned in the buffer, receive packets are front-aligned. Buffer reads and writes automatically increment the appropriate buffer pointer permitting easy sequential access from the system. Writing the Transmit Buffer Pointer register, then reading or writing the buffer data register permits random access for the transmit buffer. The Bus Pointer can only be cleared by the system, thus random access of the receive buffers is not possible.

Reception of each type of packet, and enablement of the associated interrupt, or enablement of the detection of specific transmit or receive error conditions and the enablement of the associated interrupt, is independent of other types, or errors. However, only one interrupt is issued to the system, requiring a check of the Auxiliary Status Register for the cause of an interrupt. Interrupt requests can be disabled if desired, except for the the Power-On Interrupt. It is not possible to separate the specification of packets of interest from the enablement of the associated interrupt.

3. SYSTEM INTERFACE

Software controls the EtherBox by accessing the 16 internal registers. Since the parallel port provides only one "address" bit (CMD*), a method is provided for selecting which register is intended for subsequent "data" reads and writes.

For this purpose, the EtherBox contains a SELECTION Register. When the CMD signal in the interface is asserted (CMD*=low), the SELECTION Register can be written or read by the parallel port. The lower order 4 bits of the SELECTION Register hold the 4 address bits needed for internal register selection. When the CMD is not asserted (CMD*=high), the register in the EtherBox addressed by the low order 4 bits of the SELECTION Register can be written or read by the parallel port.

So in general, to read or write an EtherBox register through the parallel port, the SELECTION Register must first be set to the address of the desired register, before the desired register can be accessed. See, also, Appendix A.

There are three separate buffer pointers in the EtherBox controller. Two of them are accessible thru the EtherBox registers. The Transmit Buffer Pointer has full read/write capability from the system. The Bus Buffer Pointer can only be reset from the system. The third pointer the Receive Buffer Pointer is not accessible from the system. Both the Transmit Buffer Pointer and the Receive Buffer Pointer are used for loopback operation.

Transmit packets are end-aligned within the 2KB transmit buffer. The Transmit Buffer Pointer is used to fill the buffer from the system, then reset by the system to the packet beginning. Once the Transmit Buffer has been switched from the system to the EtherBox (set), the pointer is then used by the EtherBox to access the data for transmission.

Receive packets are front-aligned in the two receive buffers. After the EtherBox receives a packet, the EtherBox receive status plus the contents of the Receive Buffer Pointer are found in the first two bytes of the buffer memory. The packet data follows in the buffer memory. If the packet length exceeds the legal maximum, the first 2046 bytes will be saved in the buffer. After the 2046th byte, the buffer control locks up preventing any buffer overwrite. A packet length of 2047, corresponding to a buffer pointer value of 001H indicates a packet of more than 2046 bytes.

The EtherBox will support external loopback. A maximum length packet can be looped back. Loopback requires the EtherBox to be connected to an Ethernet by the onboard or external transceiver, or fitted with special BNC or DA-15 loopback plug by the user.

Note that any broadcast packet transmitted by the EtherBox will always be looped back to itself, unless the receiver is disabled during the broadcast.

The EtherBox's suggested Ethernet station address is stored in a PROM whose contents are accessed through another register. It is addressed with the Bus Buffer Pointer, just as the Bus Buffer Pointer accesses the two receive buffers. Reading the Ethernet address PROM will automatically increment the pointer. The Ethernet address is available in PROM locations 0 thru 5 with station address byte 0 at address 0.

Ethernet Address Prom Map

0	Ethernet Addr 0
1	Ethernet Addr 1
2	Ethernet Addr 2
3	Ethernet Addr 3
4	Ethernet Addr 4
5	Ethernet Addr 5
6	Month of Manufacture
7	Year of Manufacture
8	Top Assembly Dash #
9	Top Assembly Revision Level
A-1E	Unused (Zeroes) or ASCII text
1F	One's complement of sum of locations 00 through 1E Hex

3.1 CONTROLLER REGISTER MAP

	READ	WRITE
0		Actual Station Addr 0
1		Actual Station Addr 1
2		Actual Station Addr 2
3		Actual Station Addr 3
4		Actual Station Addr 4
5		Actual Station Addr 5
6		Receive Command
7	Transmit Status	Transmit Command
8	XMT Buffer Pointer [MSB]	XMT Buffer Pointer [MSB]
9	XMT Buffer Pointer [LSB]	XMT Buffer Pointer [LSB]
A	Ethernet Address PROM	BUS Buffer Pointer Clear
B	Auxiliary Command/Status	Auxiliary Command
C	Collision Counter	Collision Counter
D	TRANSMIT Buffer	TRANSMIT Buffer
E	RECEIVE Buffer A	RECEIVE Buffer A
F	RECEIVE Buffer B	RECEIVE Buffer B

NOTE:

1. Certain registers have different interpretations depending on whether they're being read from or written to.

2. The Collision Counter must be preset to 0 by software before each transmit. On Collision 16 the associated bit is set in the Transmit Status Register, while the Collision Counter overflows and its value is indeterminate. The Collision Counter is incremented by Tx Underflow.

3.2 TRANSMIT COMMAND REGISTER

[7:4] Undefined

[3] Enable Interrupt on End of Frame

[2] " " " Collision 16

[1] " " " Collision

[0] " " " Underflow

ALL BITS ARE ONES ASSERTED

3.3 TRANSMIT STATUS REGISTER

[7:4] Undefined

[3] Ready for new frame -- A packet has been sent without errors.

[2] Collision 16 -- Sixteen successive transmissions have been attempted, but each resulted in a collision. The EtherBox will stop attempting to transmit and will return the Transmit Buffer to the System.

[1] Collision -- One or more collisions occurred. The actual number of collisions is in the Collision Counter. This bit remains set for any number of collisions, including the last one which turns on the Collision 16 bit.

[0] Underflow -- The EtherBox has not been able to supply bytes to the Ethernet fast enough. This occurs when the XMT EOF DISABLE bit is set in the Auxilary Control Register, to force a transmit underflow. This is used by diagnostics to transmit a packet with an invalid CRC. This condition also increments the collision counter.

NOTES:

1. ALL BITS ARE ONES ASSERTED

2. The Transmit Status is valid only after the transmission of a packet. Reads of the Transmit Status at any other time may result in status of a previous transmission.

3. If the Transmit Status read results in a status of all ones, (xxxx1111 binary) re-read the status. If Transmit Status is read by the System at the same time the EtherBox loads the Receive Status into the packet buffer, a bus lockout will occur, yielding this all ones condition.

4. The Transmit Buffer is switched back to the system for all the above events except Collision.

3.4 RECEIVE COMMAND REGISTER

[7:6] Address Match Mode

0 - Receiver Disabled

3 - Match station address, and multicast/broadcast

2 - Match station address and broadcast (including own broadcasts)

1 - Promiscuous on all addresses

[5] Enable Detection of Good Frame * 2,3

[4] " " " Any Frame * 3

[3] " " " Short Frame * 3,5

[2] Enable Detection of Dribble error * 3

[1] " " " CRC error * 3

[0] " " " Overflow error * 3,4

NOTES:

1. ALL BITS ARE ONES ASSERTED
2. Good Frame constitutes one without CRC, Short Frame or Dribble errors.
3. A successful detection will also generate an interrupt to the System.
4. The Overflow condition is indicative of hardware failure in the controller.
5. The initial SEEQ DQ8001 EDLC refuses to make a "runt" or Short Frame, or one with a CRC error be a frame of interest (receivable).

3.5 RECEIVE STATUS

NOTE: THE RECEIVE STATUS IS READ IN THE FIRST BYTE OF THE RECEIVE BUFFER MEMORY (RECEIVE BUFFER HEADER). IT IS NOT POSSIBLE TO READ RECEIVE STATUS FROM THE EDLC.

RECEIVE BUFFER HEADER

BYTE 0 :

[7] Stale Status -- This status is forced by controller reset, and indicates that the buffer does not hold a valid packet, but instead a discarded packet.

[6] Short Frame -- This is a packet whose length is less than the Ethernet minimum, 60 bytes (excluding the FCS and preamble). This would include collision fragments (runts).

[5] Dribble error -- This packet did not end on a byte boundary. Ethernet's "Alignment Error" is equivalent to sensing both Dribble and CRC Errors.

[4] CRC error -- The received FCS does not match the calculated one.

[3] Overflow error -- indicates controller hardware error.

[2:0] Most significant three bits of Receive Buffer Pointer

BYTE 1 :

[7:0] Least signifcant eight bits of Receive Buffer Pointer

ALL BITS ARE ONES ASSERTED

3.6 AUXILIARY COMMAND REGISTER

Bit	Name	Description
[7]	EDLC RESET	Resets the EDLC chip without affecting the rest of the controller's logic. Toggle this bit (on then off).
[6]	SYSTEM INTERRUPT ENABLE	Enables the EtherBox to assert Interrupt Request (BSY) on the parallel interface, for events other than Power-Up.
[5]	RECEIVE BUFFER B SWITCH	When set, allows EDLC access to Receive Buffer B for packet reception. It is reset by the controller upon reception of any packet-of-interest, permitting the System read/ write access to the buffer.
[4]	RECEIVE BUFFER A SWITCH	When set, allows EDLC access to Receive Buffer A for packet reception. It is reset by the controller upon reception of any packet-of-interest, permitting the System read/write access to the buffer.
[3]	TRANSMIT BUFFER SWITCH	When set, allows EDLC access to the Transmit Buffer for packet transmission. It is reset by the controller for every transmission event, except Collisions 1-15. When reset, the System has read/write access.
[2]	UNDEFINED	
[1]	XMT EOF DISABLE	Transmit End of File Disable. Normally, the EDLC gets an EOF (end of frame) bit, with the last byte of a transmit packet. When "XMT EOF DISABLE" is asserted, the EOF is not passed to the EDLC with the last byte for transmission. This will cause the EDLC input FIFO to underflow and the packet is transmitted without a CRC. This bit is asserted only for testing.
[0]	Reset Power-On Interrupt	Upon powerup or reset of the EtherBox power supplies a non-maskable interrupt will be generated. Asserting this bit clears that interrupt.

NOTES:
1. ALL BITS ARE ONES ASSERTED
2. ONLY ONES CAN BE WRITTEN INTO BITS [5:3] AND BIT [0] FROM THE SYSTEM INTERFACE.

3.7 AUXILIARY STATUS REGISTER

Bit	Name	Description
[7]	XCVR SWITCH	If set, the integral transceiver is enabled. If reset, an external Ethernet transceiver must be connected to the 15-pin DIX connector. NOTE: Do NOT connect BOTH an external transceiver to the DIX connector AND an Ethernet cable to the BNC connector at the same time.
[6:3]	<same as Auxiliary Command Register>	
[2]	RECEIVE B BEFORE A (RBBA)	If RECEIVE BUFFER B SWITCH and RECEIVE BUFFER A SWITCH are BOTH reset, then this bit is VALID, unless the EtherBox was JUST Reset or PoweredUp. If RBBA is zero then the packet in Buffer A was received before the packet in Buffer B. If RBBA is one, the packet in Buffer B arrived first. Thus software may maintain packet order.
[1]	<same as Auxiliary Command Register>	
[0]	Power-On Interrupt	The Power-On condition exist in the controller. It must be cleared by the software, before any other interrupt activity is to proceed.

4. EtherBox PROGRAMMING

The EtherBox can be operated in polling or interrupt modes. When using the interrupt mode, SYSTEM INTERRUPT ENABLE must be asserted; otherwise it may be cleared. The occurrence of an interrupt indicates the need to poll the Auxiliary Status Register. After setting the TRANSMIT BUFFER SWITCH to initiate a transmission, the EtherBox will reset the buffer switch upon successful transmission. For packet reception, both buffer switches are set and then monitored for either RECEIVE BUFFER B SWITCH or RECEIVE BUFFER A SWITCH being cleared.

For collisions on transmit, the EtherBox will automatically reload the starting buffer location to the XMT Buffer Pointer. However, the EDLC will still generate an interrupt if the enables are set in the EDLC Transmit Command byte for Collision or Collision 16 as appropriate.

If the EtherBox is successfull at transmitting on it's first attempt, as would usually be the case, then it will switch the Transmit Buffer Switch back to 0 (the system side vs the Ether side) and the Transmit Status will be 8 hexadecimal indicating successful transmission, and the Collision counter will be left at the value 0 it was initialized to by software just before the first transmission, indicating one Transmission was made.

If the EtherBox is not successful at transmitting without collisions, as would be the case with a broken or severly overloaded network, then it will eventually give up trying, and switch the Transmit Buffer Switch back to 0 and the Transmit Status will be 6 hexadecimal indicating "Collision 16" (4 hex) and "Collision" (2 Hex), and the Collision counter will be indeterminate, indicating that there were 16 transmissions, 16 collsions, 15 retries. The signifigant item to note is the "Collision16" bit in the Transmit Status Reg, and believe that rather than the Collision Counter, in that case. Failure to initialize the Collision counter will result in a reduced number of retries under some circumstances. If the Collision counter is initialized to a count higher than 0, then the EtherBox will transmit fewer times before giving up.

In the case where the EtherBox succeeds at transmission after some number of collisions, then when the Transmit Buffer switch is cleared by the EtherBox to indicate completion, the Transmit Status will be 0A hex, indicating some number of collisions greater than 0 and less than 16. The Collision counter will be left at some value indicating the number of Collisions , provided that the Collision counter was properly initialized to the value 0 just before the first transmission attempt. While interrupts may be generated for the first and 16th collision, they are not generated for intermediate collisions.

In the following descriptions, only specific bit changes are shown. Of course, this interface does not permit individual bit access so, with each register access, the programmer must be sure to set the other bits to their "current" values. For example, when operating in interrupt mode, the SYSTEM INTERRUPT ENABLE bit must be present in every access to the auxiliary command register.

4.1 TO TRANSMIT A PACKET

1. Load the packet Buffer

   Load the Transmit Buffer Pointer with 800hex minus the packet length. Write the packet to the Transmit Buffer.

2. Reload the XMT Buffer Pointer:

   Tx Buffer Pointer <--

   (2048-packet Length (in Bytes))

3. Set up the EDLC:

   EDLC Transmit Command <--

   Underflow, Collision, Collision 16 and/or
   End of Frame, as desired

4. Enable the Start of transmission:

   Auxiliary Command Register <--

   (TRANSMIT BUFFER SWITCH = 1)

5. Wait for completion, by interrupt or polling. For polling, wait for the TRANSMIT BUFFER SWITCH = 0. This indicates that the packet has been successfully transmitted or that the controller has attempted 16 transmissions and has encountered 16 collisions.

6. Read the Transmit Status Register to determine the status of the transmission.

   (Note, the Collision status signal will be asserted upon the first collision; it will stay set for the duration of processing of this packet. The Transmit Buffer is switched back to the System only after Collision 16.)

7. The EtherBox will automatically retransmit on collision. An interrupt on collision can also be obtained if the proper enable bits are set. The EtherBox discontinues retransmission after the 16th collision. If the software wishes to continue transmissions, it must explicitly re-load the Transmit Buffer Pointer and again set the Transmit Buffer switch.

8. If the System software requires the counting of the number of collisions, the Collision Counter may be read after the Transmit Status Register is read, to see how many Collisions occurred before the controller gave-up or succeeded. A hexadecimal number between 0 and F will appear in the least significant nybble of the register. The Transmit Status bit indicating collision 16 supersedes the collision counter value, indicating 16 attempts, 16 collisions, and cessation of transmissions. The collision counter must be explicitly set to the value of 0 by software, before each transmit.

4.2 TO RECEIVE A PACKET

1. Enable EDLC:

   EDLC RECEIVE COMMAND <--

   Match Mode & Enables, as desired (not 0)

2. Give Buffer(s) to EDLC:

   AUXILIARY COMMAND REGISTER <--

   (RECEIVE BUFFER A SWITCH = 1) and/or
   (RECEIVE BUFFER B SWITCH = 1)

3. Wait for completion by interrupt or polling. For polling, test for either RECEIVE BUFFER A SWITCH = 0 or RECEIVE BUFFER B SWITCH = 0. If both switches are zero then use RBBA to determine which packet arrived first.

   If interrupts are used , an interrupt will be generated if a packet of interest is received. Poll the Aux Status Register to determine which buffer contains the received packet.

4. Clear the Bus Buffer Pointer.

5. Read the EDLC RECEIVE STATUS in the upper five bits of byte zero of the receive buffer. The last three bits plus the eight bits of byte one form an offset to point to the first free byte in that receive buffer.

6. Read the packet data, which begins in byte two, and continues for a byte count of 2 less than the offset value. Note that an offset of 0 is ambiguous between having received a packet with errors (0 length) and having received a packet of 2046 bytes. This ambiguity is solved by looking at the receive status for errors. An offset value of 1 indicates a packet of greater than 2046 bytes.

5. ETHERNET INTERFACE

The EtherBox provides two options for connection to an Ethernet, selectable by an external switch. The first is the standard, DA-15 DIX outlet, which uses fully compatible signalling, including halfstep signalling per Version 2.0 of the Ethernet spec. This outlet attaches to a standard transceiver cable, which in turn is connected to any Ethernet transceiver. This would presumably be the usage when an Ethernet was pre-installed. See Appendix B which contains the concise Ethernet Specification.

The other Ethernet interface uses the onboard transceiver and is designed to be used with low-cost (50 ohm) RG-58 coax as the ether. The integral transceiver is attached to this ether via a single BNC connector on the box, to be mated with a BNC "T" pre-installed on the RG-58 coax. The station can be coupled and uncoupled without affecting network operation. The transceiver input impedance is 50K ohms versus the 100K ohm Version 2.0 spec. The transceiver does not have 802.3 heartbeat or jabber control.

The RG-58 Ethernet is electrically compatible with the yellow Ethernet coax. In fact, the RG-58 Ethernet can be attached to a yellow Ethernet by simply coupling them with an N-series/BNC adapter, and EtherBoxes can communicate with any other station on the RG-58 or yellow coax. One drawback of the RG-58 Ethernet is that the distance limitation is more severe: approximately 330 meters of an RG-58-only segment.

The integral transceiver provides complete electrical isolation and is in every other way Ethernet compatible with one exception: it has no RECEIVER-based collision detection. Transmit collison detection is, of course, fully implemented.

[Receiver-based collision detection permits a transceiver to detect collisions even when it is not participating in them. Without this, a transceiver might erroneously sense no carrier when, by chance, the colliding stations' signals saturate the net to produce no bit transitions. If during this time, the EtherBox has a packet to transmit it will think the coax is free, try to transmit its packet, immediately detect a collision (since transmit collisions are always detected) and backoff. Theoretically, if other similarly-equipped stations are in a particular, staggered synchronization with this one and each other, this could lead to a self-perpetuating transmit/backoff cycle (a so-called "Ethernet tsunami") among these stations. The probability of this is extremely small.

6. PERFORMANCE

6.1 System Interface

Write Operations to EtherBox buffer memory	3Mbytes/sec	max
Read Operations from EtherBox buffer memory	2MBytes/sec	max

6.2 Ethernet Interface

Conforms fully to the Ethernet specification , Version 1.0 , published on 30 September, 1980, by DEC, Intel, and Xerox.

Bit Rate	10Mbits/sec
Packet spacing	9.6 microseconds min.

7. PACKAGING

The EtherBox is contained in a stand-alone metal box, of dimensions 3.25" x 8.75" x 18.75". It includes an integral power supply drawing less than 35W. A sketch of the enclosure is attached.

The box will interface to the external world through the following ports:

- Fused, power cord receptacle. Separate power cord. (No ON/OFF switch or pilot light.)
- Female, DB-25 connector for parallel port, which connects to a shielded cable as specified in Appendix A.
- Female, DA-15 connector with slide lock for connection to external transceiver.
- Female, BNC connector for low-cost RG-58 Ethernet.
- Slide switch to select between the DA-15 and BNC.

8. ENVIRONMENTAL

The EtherBox will be tested to verify adherence to the following environmental specifications:

8.1 Operating Temperature

0C to 60C, ambient max.

8.2 Transit Temperature

-40C to +65C, ambient max. for a period of 72 hours

8.3 Humidity

5% to 95% at 25C to 40C, non-condensing.

8.4 Line Voltage and Frequency (North American Version)

115v + 6%, -15% and with frequency variation of 2 cylces from 60Hz, less than 35 watts.

9. Regulatory Certification (North American Version)

9.1 Safety Standards

The EtherBox will be UL certified under UL 114 (Office Machines) and UL 478 (Data Processing Equipment).

The EtherBox will be CSA certified under CSA C22.2 No.154 (Data Processing Equipment).

9.2 Electromagnetic Compatibility

The EtherBox will be certified as a Class A device under FCC Part 15, Subpart J for radiated and conducted emissions. Testing will be performed in a simulated environment with a host processor (electromagnetically isolated from the EtherBox if required).

10. European Version

A separate version of the EtherBox will be manufactured for European markets. The only external difference will be the line voltage and frequency, along with different power cord(s).

To satisfy European safety standards, the EtherBox will comply with IEC 380 (Safety of Electrically Energized Office Machines, Safety Extra Low Voltage). A compliance report can be made available.

Because of its peripheral nature, the EtherBox cannot be certified on its own under the VDE standards. However, we intend to perform some tests to assure that the EtherBox will not contribute to overall system emissions.

Appendix A: "Centronics" Parallel Port Interface

The signal DIRection is:

<	From EtherBox to System
>	To EtherBox from System
<>	Bi-Directional

PIN	Wire Color	SIGNAL DIR	NAME	DESCRIPTION
7				Unused. No Connection.
	Pair 1.....			
14	Black		Common	
15	Red	>	PSTRB*	Processor Strobe--a data strobe from the computer signifying valid data. For writes from the computer to the Etherbox, the data should be stable for the duration. For reads into the computer from the EtherBox, the computer should latch on the rising edge.
	Pair 2.....			
20	Black		Common	
21	White	>	CRES*	Controller Reset--resets all logic in the controller. Min. 10 microseconds active.
	Pair 3.....			
2	Black		Common	
16	Green	<	BuSY	Busy--A High Level is an active INTERRUPT. Rcv. > 600 nsec. pulse. Tx. active until Tx. status read.
	Pair 4.....			
4	Black		Common	
18	Blue	<	PARITY*	Parity--A signal from the peripheral used to generate parity comparision on data received from parallel port. Odd parity.
	Pair 5.....			
19	Black	<	OCD	Open Cable Detect--used to determine if the peripheral is not connected. Sense of signal is positive indicates Open Cable. Ground indicates connected cable. Grounded in EtherBox.
25	Yellow			NOT CONNECTED.

PIN	Wire Color	SIGNAL DIR	NAME	DESCRIPTION
	Pair 6			
17	Black	>	CMD*	Command--a signal from the computer indicating whether the data bus is carrying "command/status" information or "data". When asserted, the data bus carries "command/status" information.
3	Brown	>	DRW	Data Bus Direction--a signal from the computer that controls the direction of the data transceivers at both ends of the cable. When asserted, this data path is toward the computer.
	Pair 7			
5	Black	<>	DD0	Data Bus Bit 0
6	Orange	<>	DD1	Data Bus Bit 1
	Pair 8			
8	Red	<>	DD2	Data Bus Bit 2
22	White	<>	DD3	Data Bus Bit 3
	Pair 9			
23	Red	<>	DD4	Data Bus Bit 4
11	Green	<>	DD5	Data Bus Bit 5
	Pair 10			
12	Red	<>	DD6	Data Bus Bit 6
13	Blue	<>	DD7	Data Bus Bit 7
	Cable Shield			
1,9,10,24, Shell			Common	These pins and the connector shell are all interconnected with 22AWG Bus Wire.

Note: signal names followed by an asterisk ("*") are ACTIVE LOW.

A pre-fabricated cable such as the Belden 9668, Style CB, from Belden ISO, (704)865-4513, may be used; or to construct you own: The wires in the cable should be either unpaired (non-twisted) or should be twisted pairs as described above. Each end of the cable has a male 25pin D connnector with a tin plated shell with spring indents. The recommended 10-pair 24AWG, overall foil + braid sheilded cable is Belden 9835 of maximum 5 foot length (1.5 meters). The 22AWG Bus Wire is Belden 8021.

The connectors are equivalent to the following AMP part numbers:

Male Shell	207464-2	Qty 2
Male Pins	205089-1	Qty 48
Hood Ass'y	745173-2	Qty 2
Split Ring	745508-8	Qty 2

Parallel Port DC Signal Characteristics

EtherBox Output Signals

logic "1"	2.0V min at	-15mA Max
logic "0"	0.5V max at	24mA Max

EtherBox Input Signals

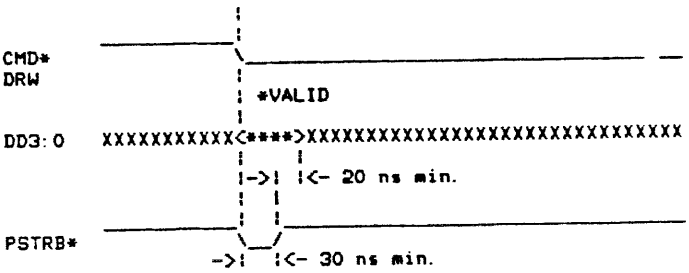
Data Lines

logic "1"	2.0V min	20 micro A at 2.7V
logic "0"	0.8V max	-0.2mA at 0.4V

Control Lines

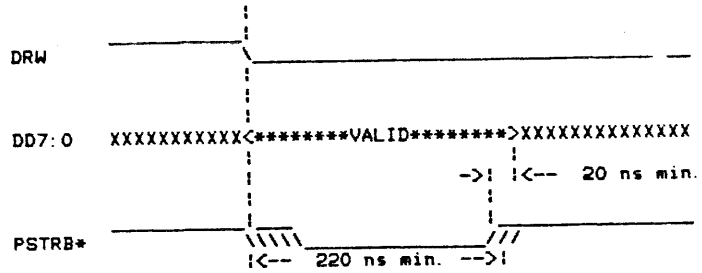
logic "1"	2.67V min	20 micro A Max
logic "0"	0 < V < 1.4V max	-15ma Max

REGISTER SELECT

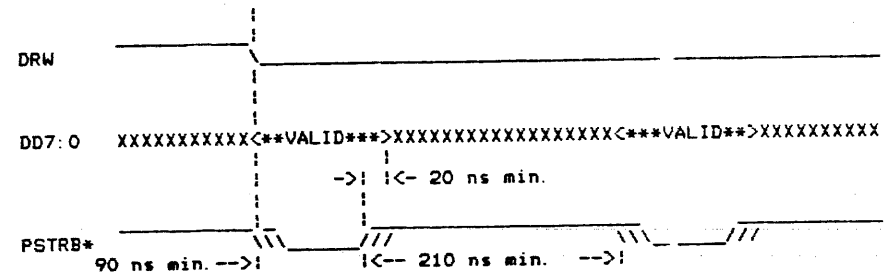


NOTE: THE "CMD" LINE SHOULD BE HELD INACTIVE FOR ALL THE FOLLOWING ACTIONS.

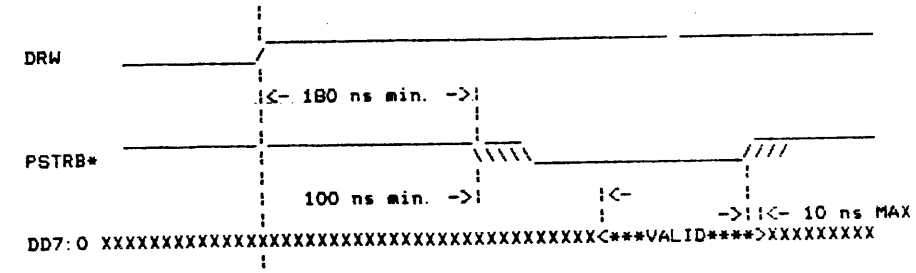
COMMAND WRITE (Receive, Transmit and Auxilary Command Registers only)



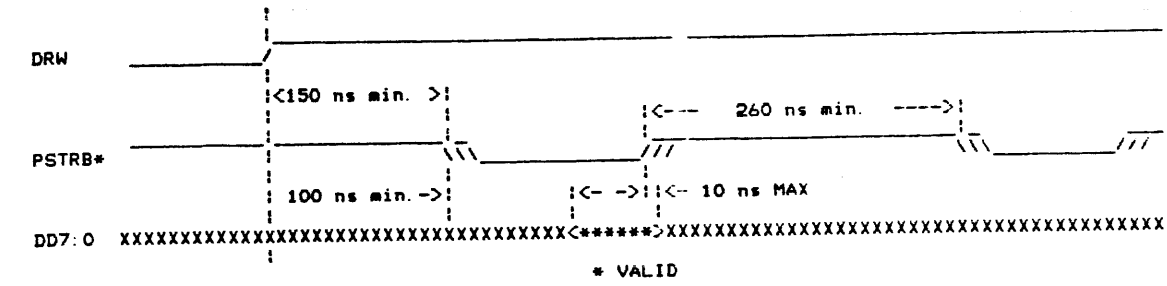
DATA WRITE (All registers except the Receive, Transmit and Auxilary Command Registers)



STATUS READ (Transmit and Auxilary Status Registers only)



DATA READ (All Register reads except the Transmit Status and Auxilary Status Registers)



To: Larry Gardner

From:

Date: 27-May-83

Subject: Interrupt problems with 3COM I/F

CC: Sunny Kirsten, Al Lem, Larry Birenbaum (c/o 3COM)
K. Okin, J. Davis, F. Jones, D. Offen, J. Weil, B. Fox

This note summarizes recent conversations between myself and 3COM engineering -- notably Sunny Kirsten and Al Lem -- about problems with to the E-Box vis-a-vis interrupts.

Prior to my involvement, Frank Jones had put together a list of interrupt related problems which he had seen in his testing of the E-Box. Since I was more concerned with the basic transmission aspects of the box, I had not tested the interrupt related hardware until recently. In conversations with Sunny prior to this week, it seemed that the problems could be handled with proper software. However, upon looking at the problem earlier this week, we (Sunny, Al and myself) feel that the problem must be addressed in the 3COM box. I suggested a solution over the phone yesterday which they are analyzing to see if that's the way they want to solve the problem.

The basic problem stems from the 6522 characteristics and the manner in which it must be used in our 2-port card. 3COM had been assuming that our board was capable of "latching" interrupt events on the 2-port card. Hence, they **do not** latch the **BSY** line (used for interrupt control). Whenever the A-port of the 6522 is accessed (as it must for data transfers), the interrupt flag gets cleared inside of the 6522; this, in turn, prevents the 6522 from generating an **IRQ** into the Lisa interrupt system.

Within the E-Box, the EDLC **INT** line is gated (via the E-Box's System Interrupt Enable) directly to the **BSY** line. The EDLC keeps its **INT** asserted until someone reads the corresponding status register. For transmits, the Host (Lisa) reads the Transmit Status Reg; hence, **INT** (**BSY**) will stay asserted. However, the E-Box always automatically reads receive status -- storing it into the receive buffer -- from the EDLC. Hence, the **INT** (**BSY**) line is asserted for only a brief instant -- on the order of 1-2 µsec. This is just barely long enough for the 6522 to sense the edge-triggered CA1 line (coming in on BSY) and "latch" the interrupt.

However, any activity occurring over the 2-port card (e.g., setting up a transmit or reading the other receive buffer), will cause the latched interrupt condition to be cleared. This causes the interrupt to be "lost". This has been noted by Sunny and is probably the source of problems noted by Frank.

My suggested solution is to include the **RBSA*** and **RBSB*** lines as terms in the input to the interrupt gate. Thus, the **BSY** line would be asserted whenever **SIE** is true **and** (the EDLC **INT** line is asserted **or** either receive buffer is available to the Host). This, in effect, latches the receive interrupts within the E-Box. **BSY** would go away when the Host (Lisa) reads the Transmit Status Reg (in case of a transmit complete) or gives the receive buffer(s) back to the E-Box (for receive interrupts).

(11:00, 27-May-83) The 3COM solution (as per conversations with Al) will be to actually latch the **INT** signal within the E-Box. This is the cleanest solution.

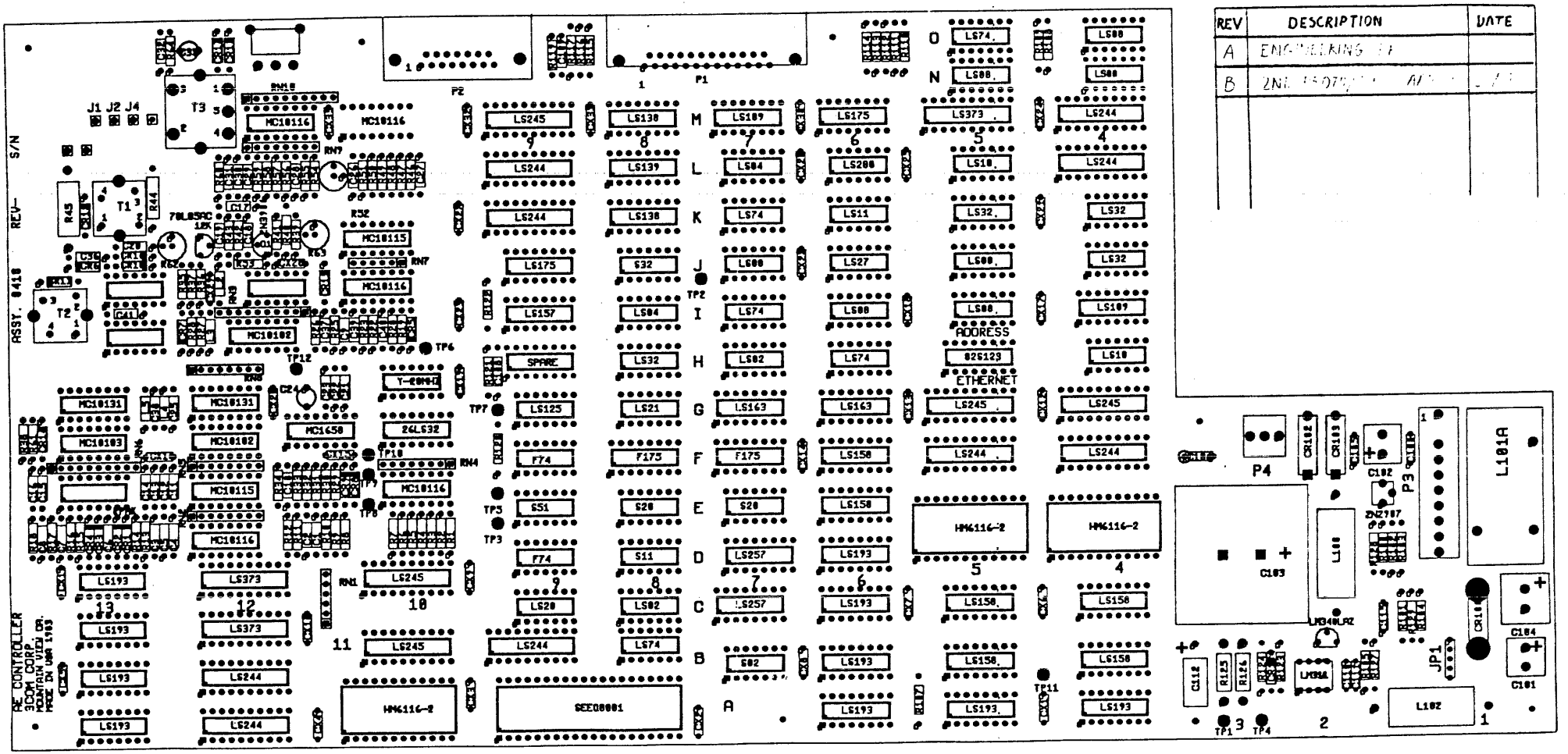
There is still a "minor" problem which must be resolved by software. The **BSY** interrupt is edge-triggered. Even with the suggested change, the leading edge of **BSY** could be "lost" due to the 6522 problem; this only happens when a "driver" is using the I/F to talk to the E-Box. However, the state of the **BSY** line is also readable via the **PB1** input. Since the interrupt is latched (assuming some change similar to that proposed above is made), the Host can sense if **BSY** is asserted before waiting for an interrupt. If it always does this after using the I/F, and "fakes" an interrupt, the interrupt will not get "lost" and all will be well.

I talked with Al and Larry Birenbaum this morning. The current assumption is that we can use the E-Box as currently implemented (using a software kludge described below), but that they will fix the problem in a future board rev. Thus, my feeling is that we can accept the delivery of the current design and assume that the next scheduled set will have the problem fixed in hardware (within the E-Box).

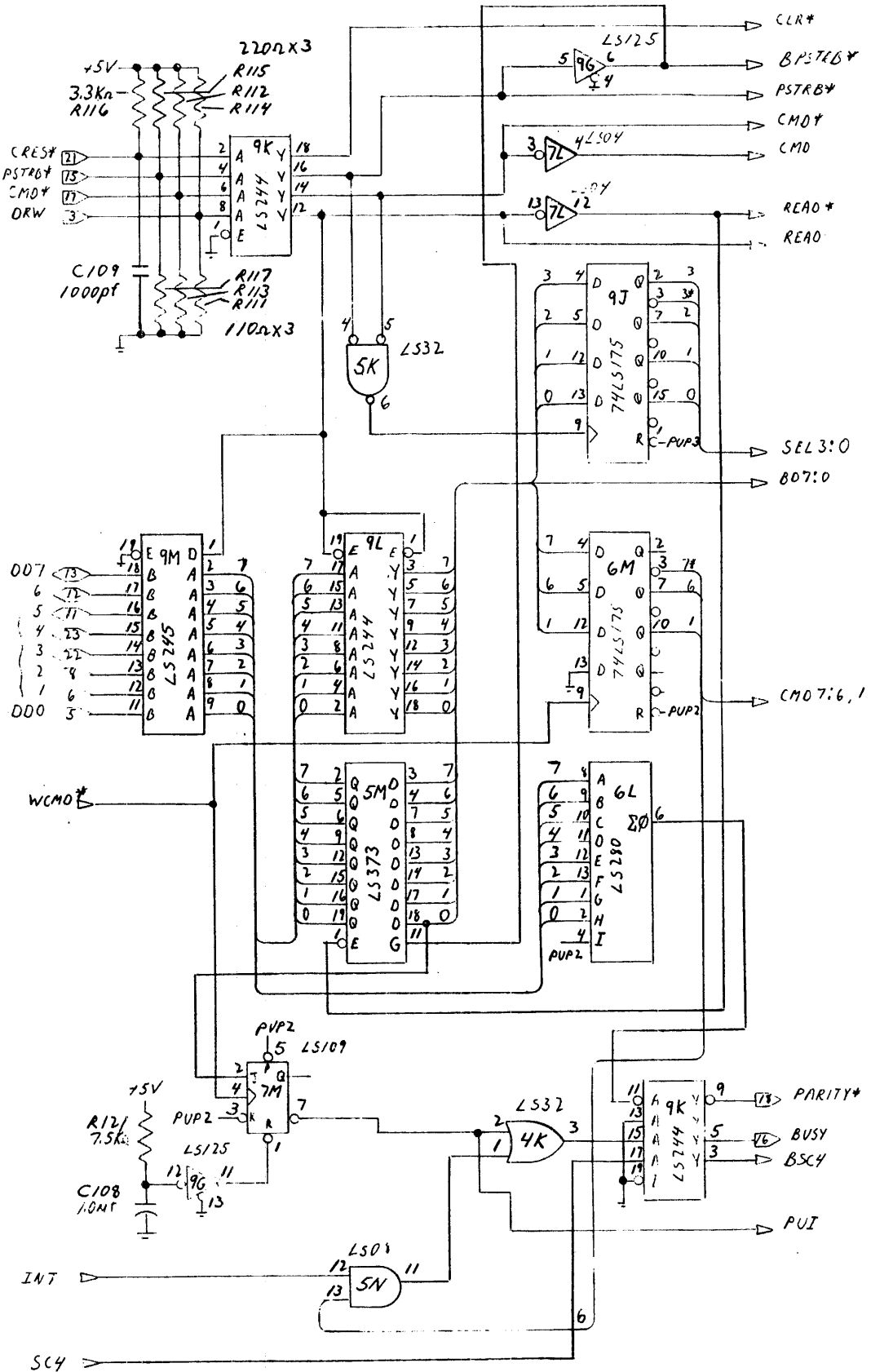
The software kludge for existing E-Boxes involves checking, in software, the conditions in my suggestion to a hardware fix. I.e., after playing with the I/F, the software must poll both **PB1** and **ASR**. If either **PB1** is asserted **or** either of the receive buffers has been given back to the Host (Lisa), then an interrupt must be faked. This is done by pushing a dummy interrupt status onto the stack and jumping to the entry of the interrupt handler. (I am enclosing a copy of my code to do this as a sample.)

As of this note (11:00, 27-May-83), the above software "kludge" seems to be effective. I have run my echo test utilizing this interrupt "faking" and it successfully runs without "dropping" interrupts.

Note, some problems noted by Frank may not be solved by this problem. This solution addresses only those of "lost" interrupts. That is, cases where the E-Box correctly generates status but where the associated interrupt is not being "caught". I suggest that Frank change his tests to include the kludge and test the boxes again. Any problems not directly related to the "interrupt problem" must be treated independently.



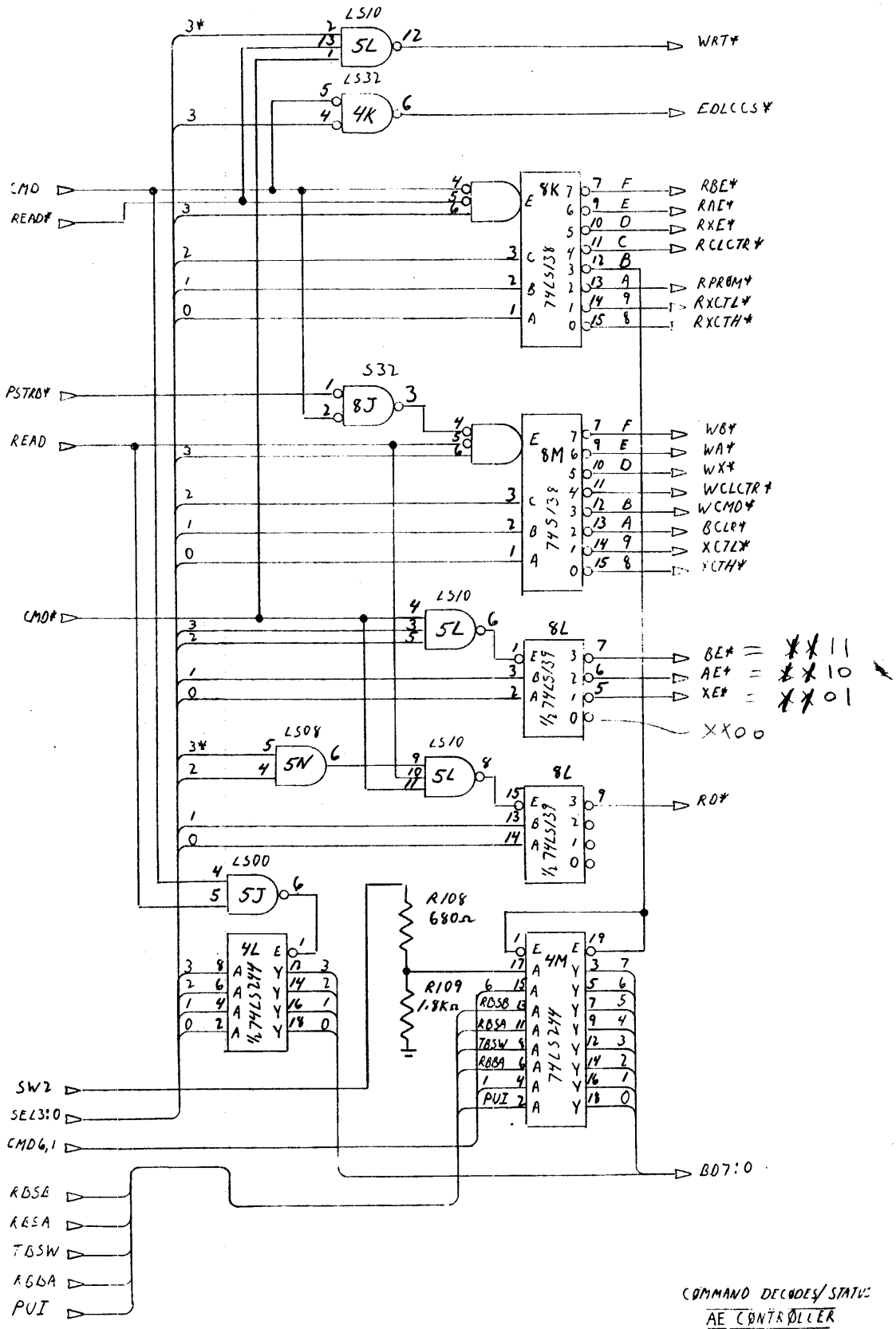
3Com Etherbox Ethernet Controller



PARALLEL PORT INTF.

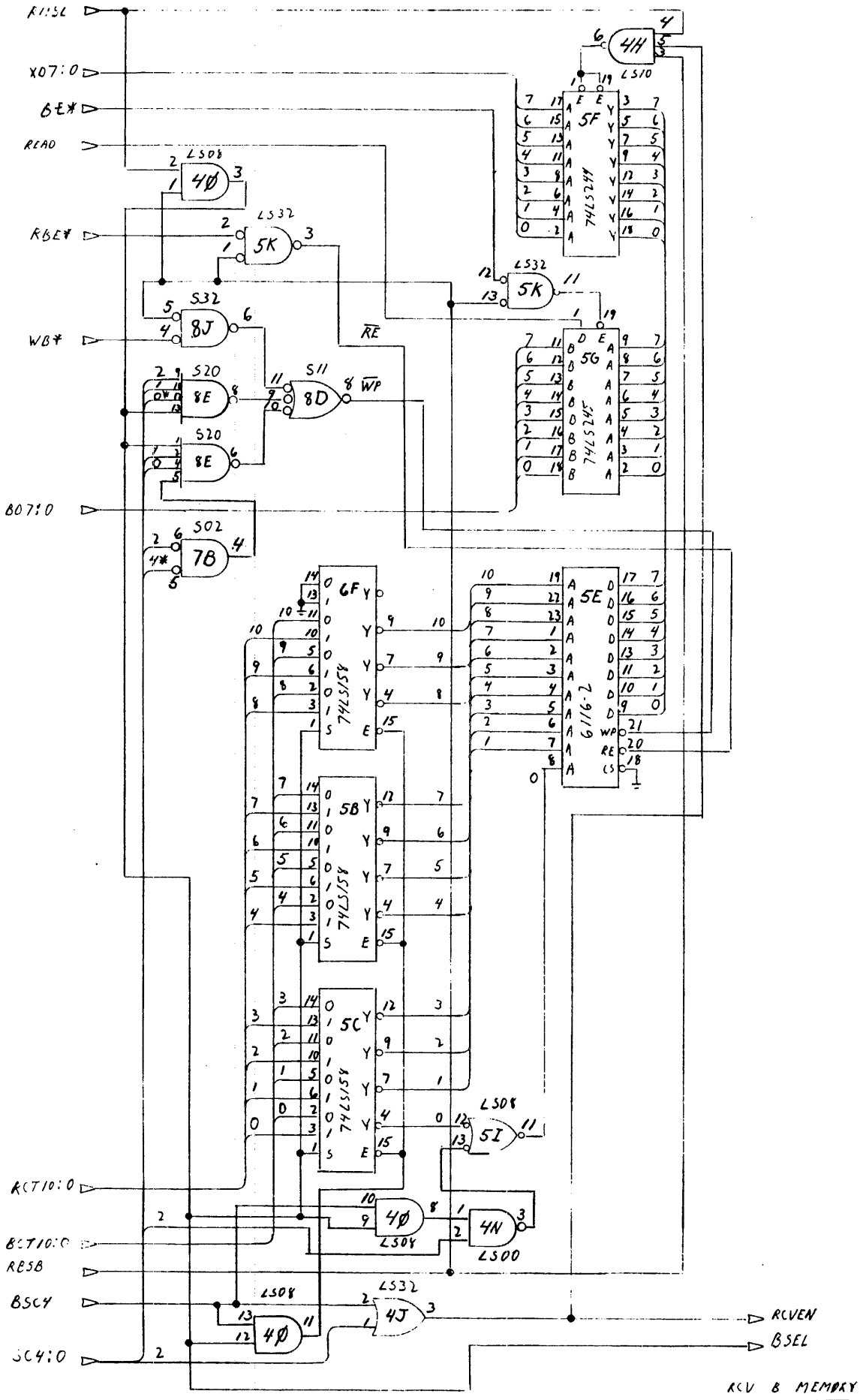
AE CONTROLLER

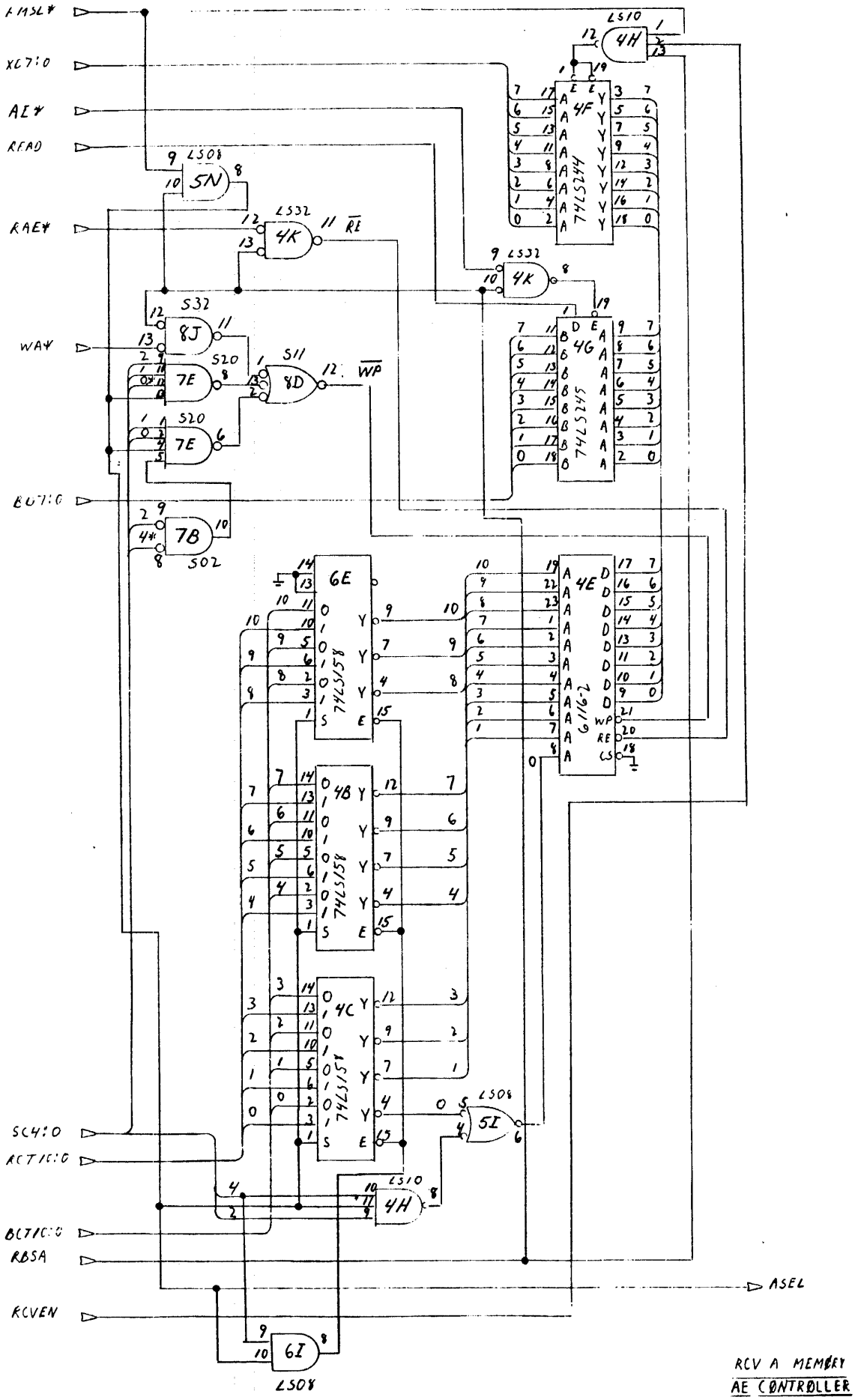
SH.1



COMMAND DECODES/STATUS
AE CONTROLLER

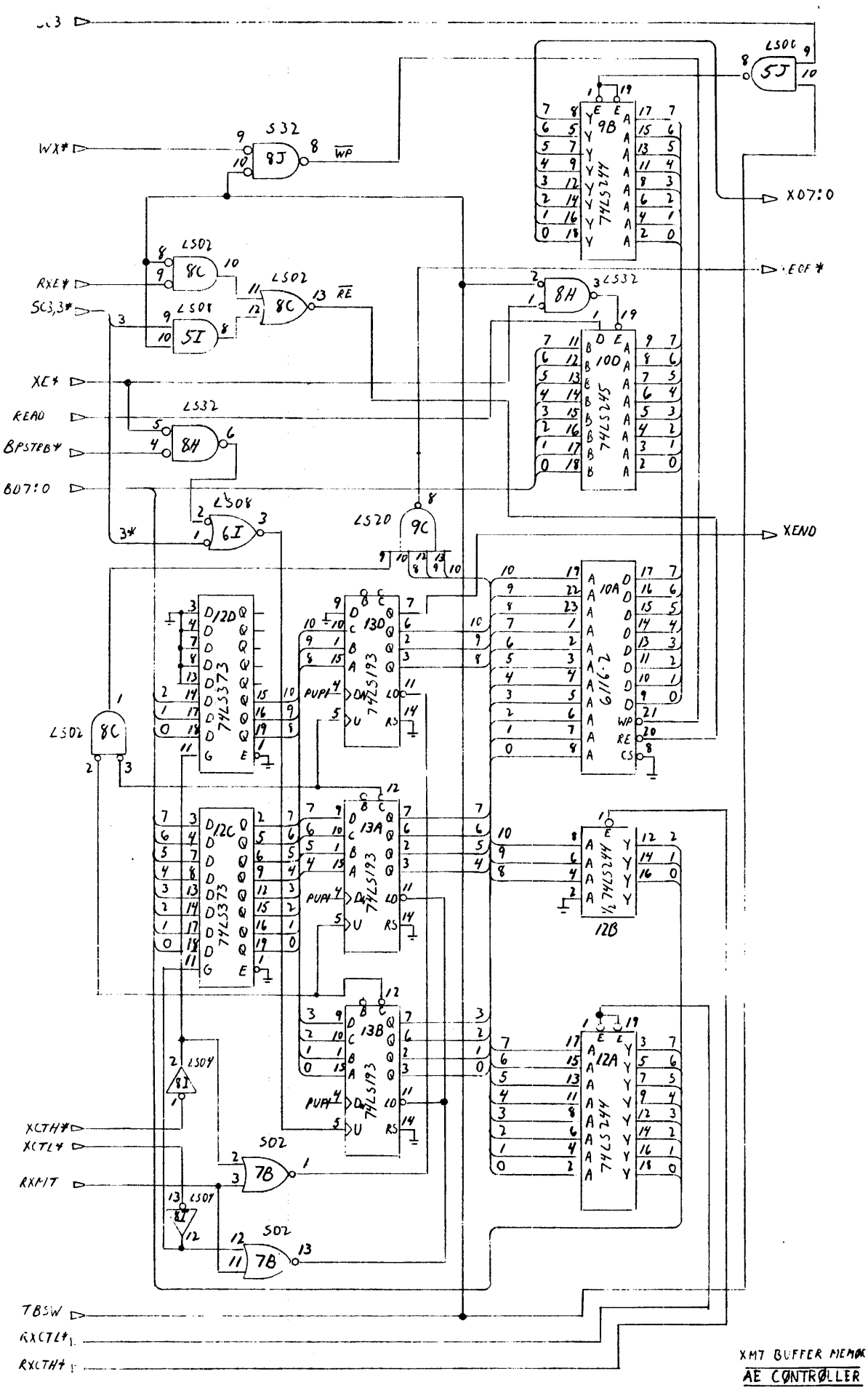
SH.2

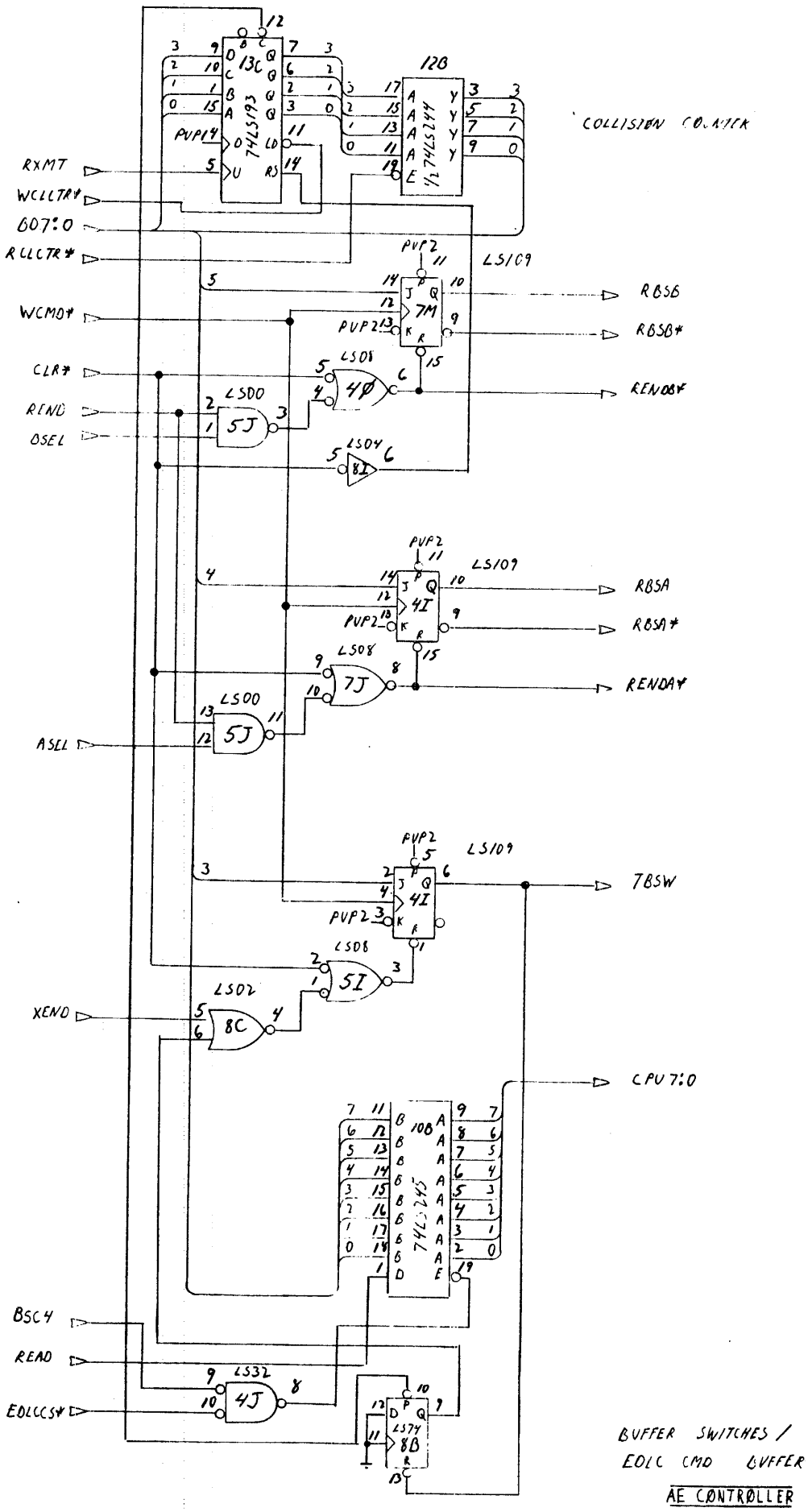

RCV B MEMORY
AE CONTROLLER
SH.3

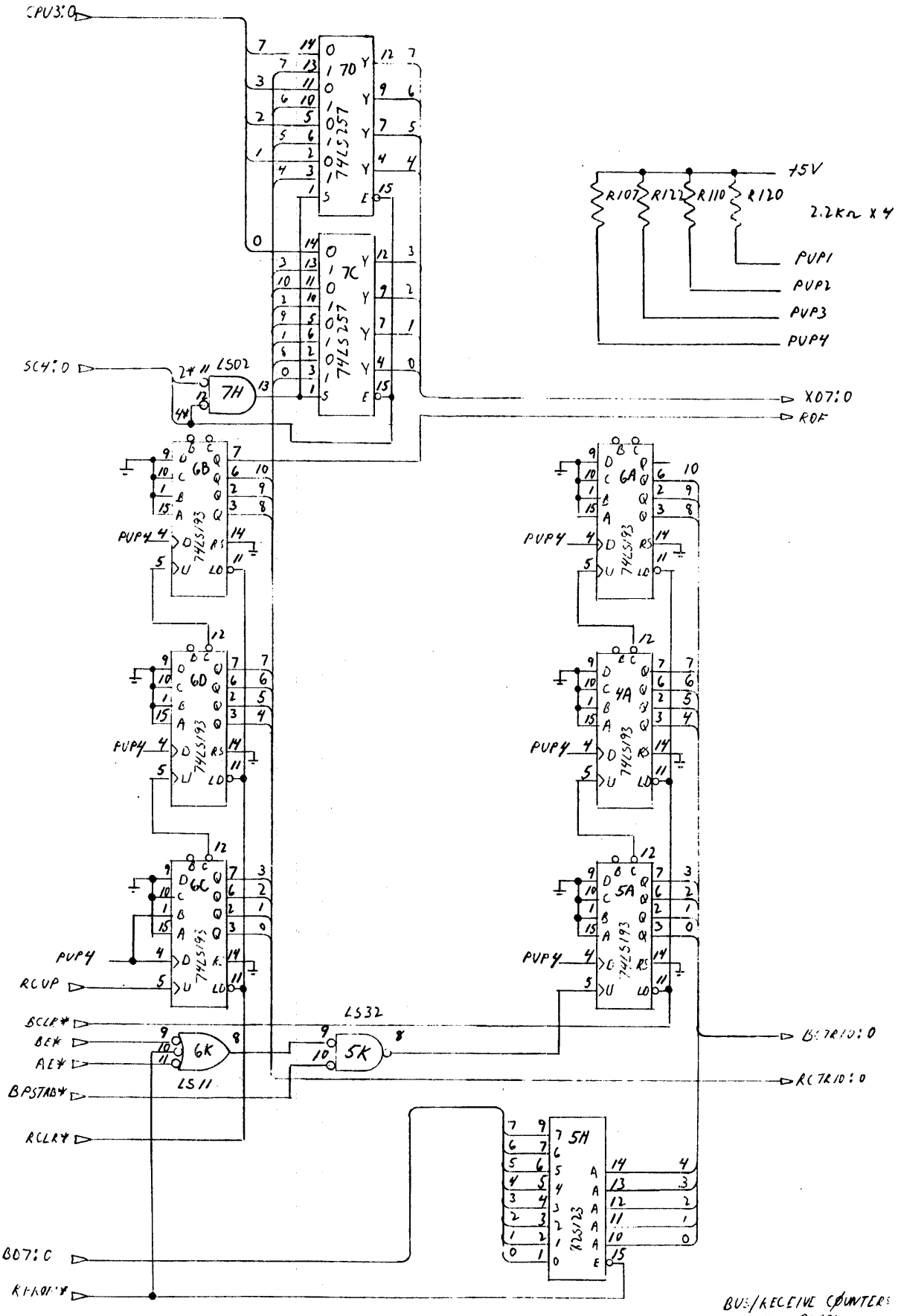


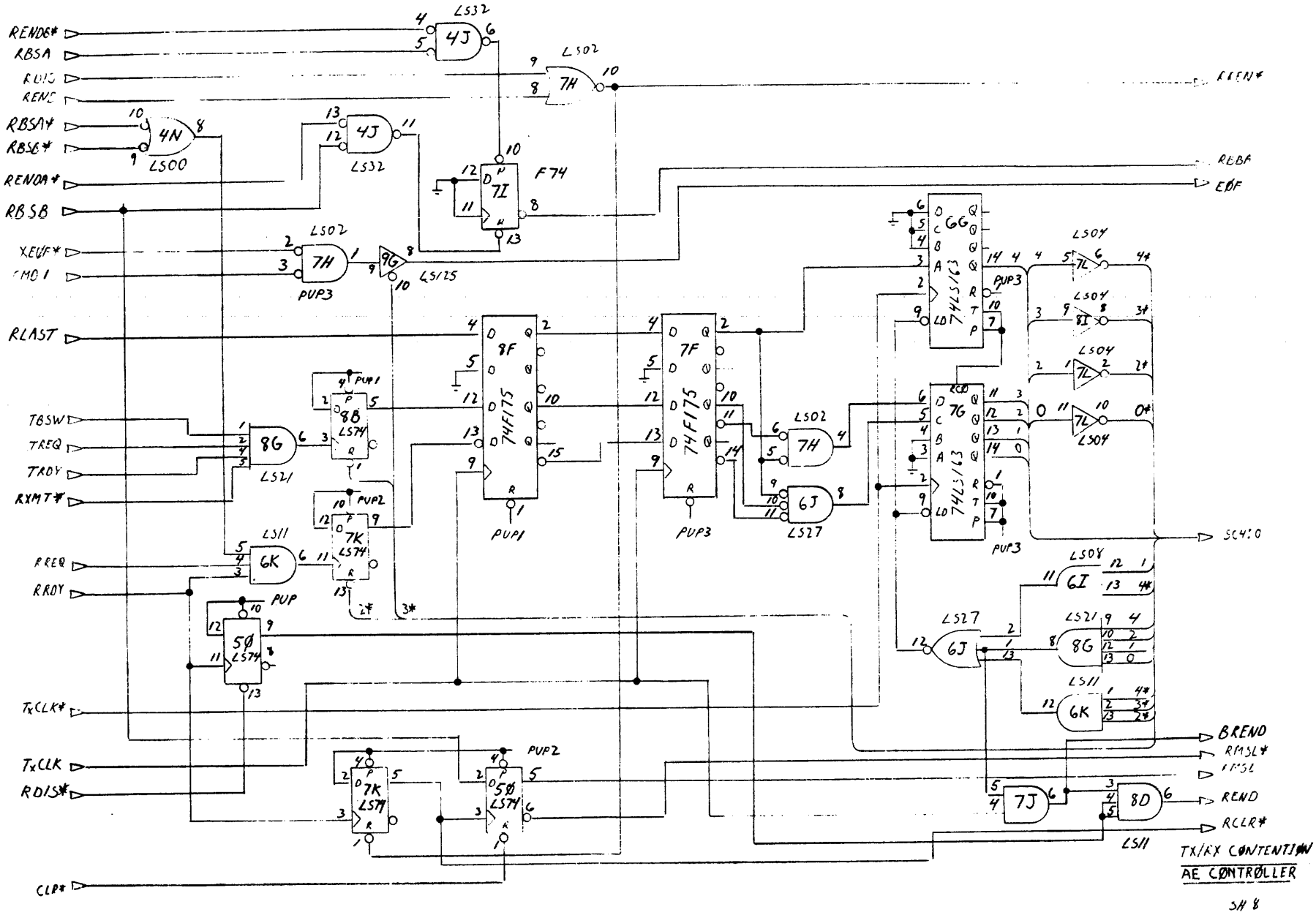
RCV A MEMORY
AE CONTROLLER

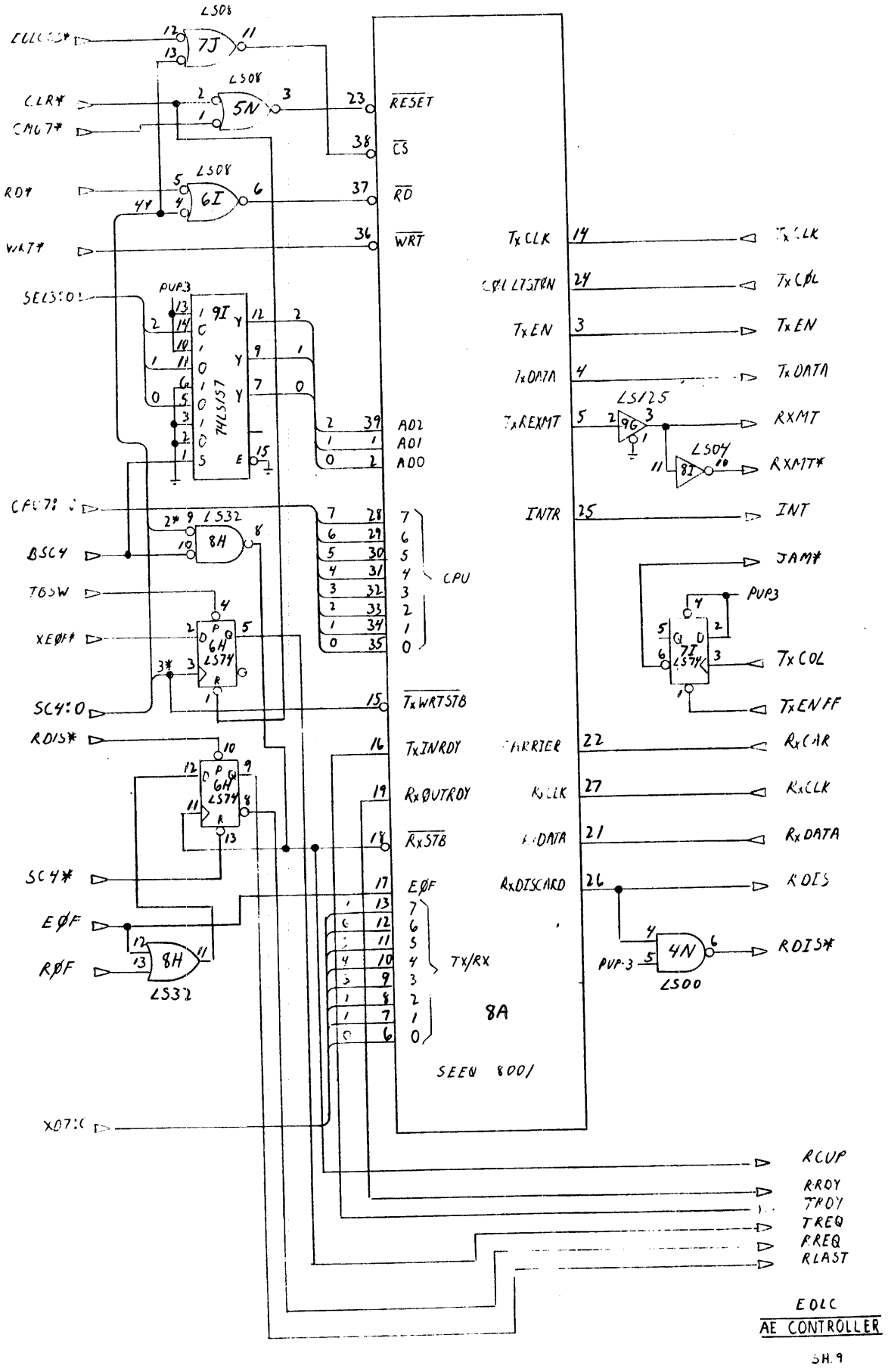
SHY

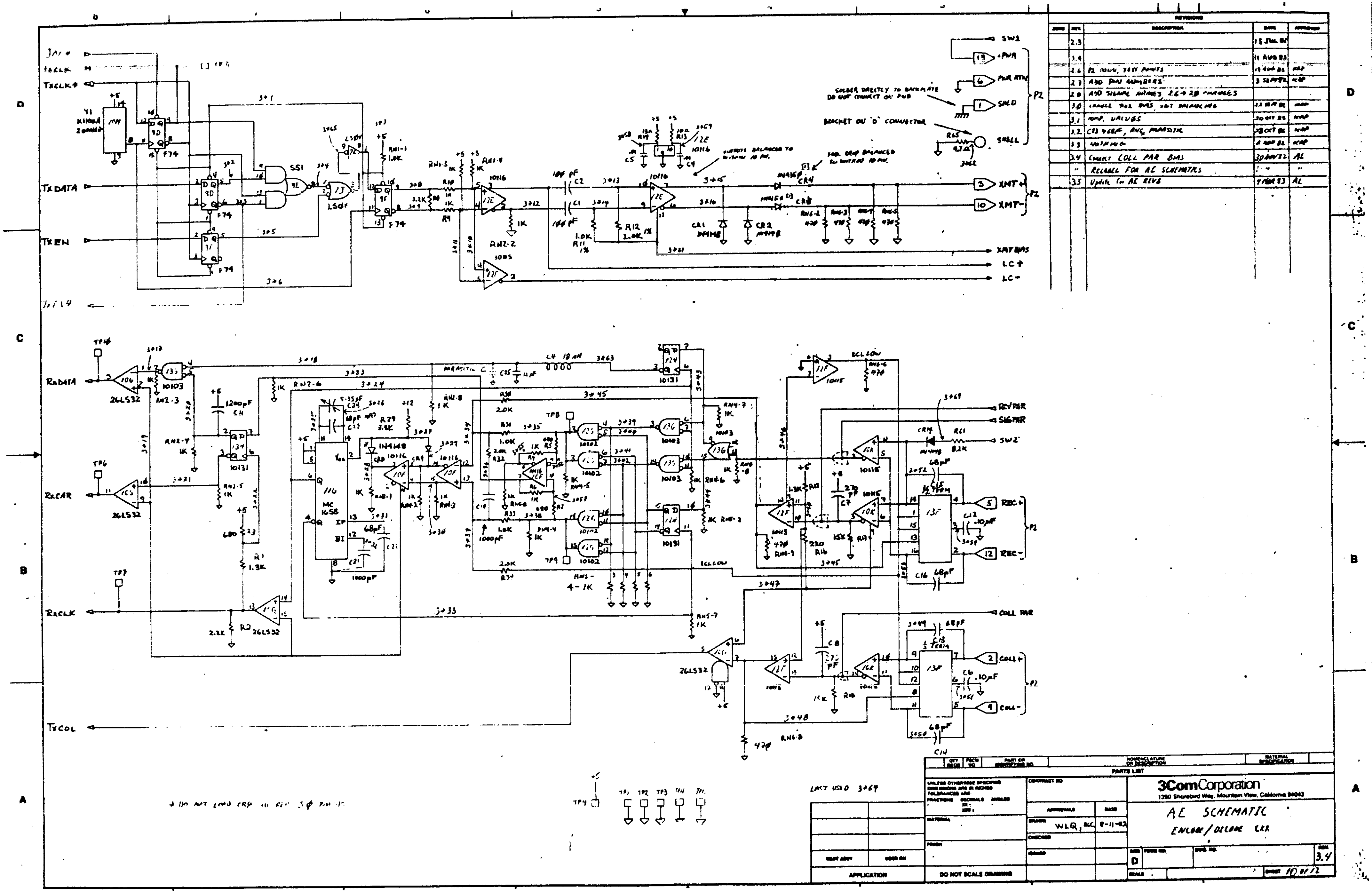

WX*
LS32
8J
WP
LS00
5J
X07:0
LS02
8C
RXE*
SC3,3*
LS04
5I
RE
EOF*
LS32
8H
XC*
READ
BPSTRB*
B07:0
LS08
6I
LS20
9C
XEND
74LS373
12D
12C
74LS193
13C
13B
13A
74LS244
9B
10D
10A
12A
12B
6116-2
LS04
8I
XCTH*
XCTL*
RXMIT
LS02
7B
TBSW
RXCTL*
RXCTH*
XMT BUFFER MEMORY
AE CONTROLLER
SHS


13C
74LS193
12C
1/2 74LS244
COLLISION COUNTER
RXMT
WCLLCTR*
D07:0
RCLLCTR*
WCMD*
CLR*
RENB
OSEL
ASEL
XENB
BSC4
READ
EOLCCS*
PVP2
LS109
7M
4Ø
5J
8I
4I
7J
5I
8C
4J
LS374
8B
10B
74LS245
RBSB
RBSB*
RENDB*
RBSA
RBSA*
RENDA*
7BSW
CPU7:0
BUFFER SWITCHES /
EOLC CMD BUFFER
AE CONTROLLER
SH 6


CPU3:0
74LS257
7C
7H
LS02
SC4:0
+5V
R107
R122
R110
R120
2.2kΩ X 4
PUP1
PUP2
PUP3
PUP4
X07:0
ROF
74LS193
6B
6D
6C
6A
4A
5A
PUP4
RCUP
LS32
BCLR*
BE*
AE*
6K
LS11
5K
BPSTAB*
RCLR*
5H
RS23
B07:0
RF*
BUS/RECEIVE COUNTERS
EA PROM
AE CONTROLLER
SH 7


RENDB*
RBSA
RDIS
RENC
RBSA*
RBSB*
LS00
RENDA*
RBSB
XEOF*
CMD 1
RLAST
TBSW
TREQ
TRDY
RXMT*
RREQ
RRDY
TxCLK*
TxCLK
RDIS*
CLR*
LS32
LS02
LS125
LS74
LS21
LS11
LS27
LS04
74F175
74LS163
PUP1
PUP2
PUP3
PUP
F74
REEN*
REBA
EOF
SC4:0
BREND
RMSL*
REND
RCLR*
TX/RX CONTENTION
AE CONTROLLER
SH 8


CLR*
CMD7*
RD*
WR7*
SEL3:0
CA7:3
BSC4
TBSW
XEØF*
SC4:0
RDIS*
SC4*
EØF
RØF
XD7:0
RESET
CS
RD
WRT
AD2
AD1
AD0
CPU
TxWRTSTB
TxINRDY
RxOUTRDY
RxSTB
EØF
TX/RX
8A
SEEN 8001
TxCLK
COLLISTRN
TxEN
TxDATA
TxREXMT
INTR
CARRIER
RCLK
RDATA
RxDISCARD
TxCLK
TxCOL
TxEN
TxDATA
RXMT
RXMT*
INT
JAM*
PUP3
TxCOL
TxENFF
RxCAR
RxCLK
RxDATA
RDIS
RDIS*
LS08
LS32
LS04
LS74
LS125
LS00
74LS157
RCVP
RRDY
TRDY
TREQ
RREQ
RLAST
EDLC
AE CONTROLLER
SH.9



ZONE	REV.	DESCRIPTION	DATE	APPROVED
	2.3		15 JUL 82	
	2.4		11 AUG 82	
	2.6	P2 conn, test points	19 AUG 82	MAP
	2.7	ADD PIN NUMBERS	3 SEPT 82	MAP
	2.8	ADD SIGNAL NAMES, 2.6 → 2.8 CHANGES		
	3.0	CHANGE XMT BIAS, XMT BALANCING	22 SEP 82	MAP
	3.1	COMP. VALUES	20 OCT 82	MAP
	3.2	C23 = 68pF, RNG, PARASITIC	28 OCT 82	MAP
	3.3	NOTHING	[illegible]	MAP
	3.4	Correct COLL PAR BIAS	30 NOV 82	AL
	"	RELABEL FOR AE SCHEMATICS	"	"
	3.5	Update for AE REVS	9 MAR 83	AL

3Com Corporation
1390 Shorebird Way, Mountain View, California 94043

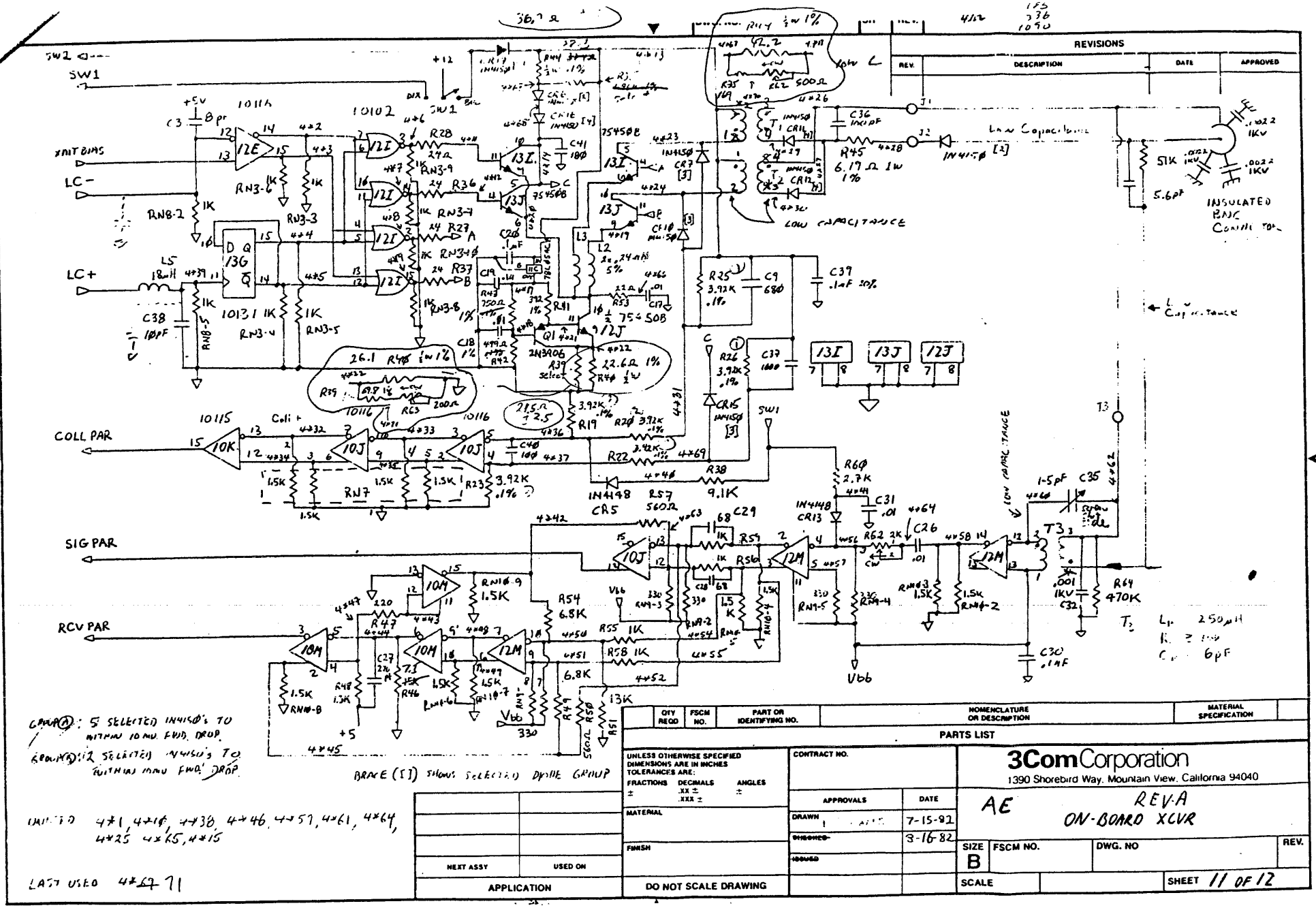
AE SCHEMATIC
ENCODE/DECODE CKT

DRAWN	WLQ, KCC 8-11-82
SIZE	D
REV	3.4
SHEET	10 OF 12

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS DECIMALS ANGLES

PARTS LIST — QTY REQD | FSCM NO. | PART OR IDENTIFYING NO. | NOMENCLATURE OR DESCRIPTION | MATERIAL SPECIFICATION

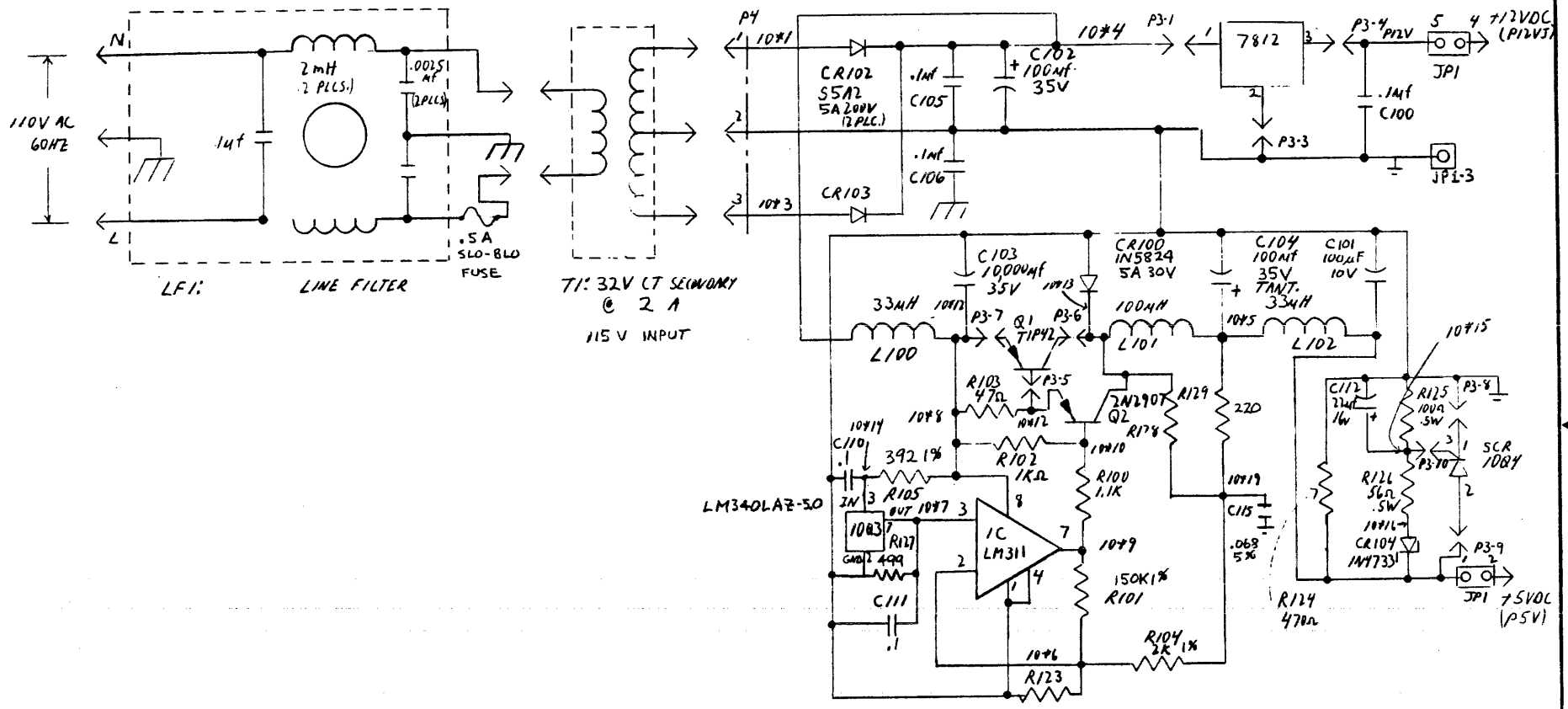
CONTRACT NO. | APPROVALS | DATE | CHECKED | ISSUED | MATERIAL | FINISH | NEXT ASSY | USED ON | APPLICATION | DO NOT SCALE DRAWING | SCALE


REVISIONS
REV.
DESCRIPTION
DATE
APPROVED
SW1
SW2
+12
+5V
XMIT BIMS
LC-
LC+
COLL PAR
SIG PAR
RCV PAR
INSULATED BNC CONNECTOR
LOW CAPACITANCE
Vbb
GROUP (A): 5 SELECTED IN4150's TO WITHIN 10 MV FWD DROP.
GROUP (B): 2 SELECTED IN4150's TO WITHIN 10MV FWD DROP.
BRACE (I) SHOWS SELECTED DIODE GROUP
UNUSED 4*1, 4*16, 4*38, 4*46, 4*57, 4*61, 4*64, 4*25, 4*65, 4*15
LAST USED 4*67 71
QTY REQD
FSCM NO.
PART OR IDENTIFYING NO.
NOMENCLATURE OR DESCRIPTION
MATERIAL SPECIFICATION
PARTS LIST
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS
DECIMALS
ANGLES
MATERIAL
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
CONTRACT NO.
APPROVALS
DATE
DRAWN
CHECKED
ISSUED
7-15-82
3-16-82
3ComCorporation
1390 Shorebird Way, Mountain View, California 94040
AE
REV.A
ON-BOARD XCVR
SIZE
B
FSCM NO.
DWG. NO
REV.
SCALE
SHEET 11 OF 12

DWG. NO.	SH	REV.

REVISIONS

REV.	DESCRIPTION	DATE	APPROVED



QTY REQD	FSCM NO.	PART OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	MATERIAL SPECIFICATION

PARTS LIST

NEXT ASSY	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±

MATERIAL

FINISH

DO NOT SCALE DRAWING

CONTRACT NO.

APPROVALS	DATE
DRAWN a Cm	11/12/82
CHECKED	
ISSUED	

3Com Corporation
1390 Shorebird Way, Mountain View, California 94040

AE POWER SUPPLY CIRCUIT
3-5-83 - 6...

SIZE	FSCM NO.	DWG. NO.	REV.
B			A

SCALE

SHEET 12 OF 12