Q219 2.6
Light pen/Graphics System

Note The lightpen system is no longer used, however the hardware support is still present on the Q219 card. For this reason we have not deleted references to the lightpen and associated circuitry.

Introduction

The Q219 provides a graphics display and light pen input system. The 16 kilobytes VRAM is displayed as an array of 512 pixels (horizontal) by 256 lines (vertical). The lightpen can detect points within this array.

To increase graphics drawing speed, extensive use is made of special hardware functions which provide automatic address incrementing or decrementing along either or both axes and individual pixel writing at the current address.

Video RAM is arranged such that consecutive bits in each byte correspond to consecutive pixels in the horizontal line. With 512 pixels per line, 64 bytes of VRAM is required per line. Thus there are 64 bytes between a given pixel and the pixel vertically below it. The first byte in VRAM ($8000) appears at the top left corner of the screen.

Locations FCD0 to FCDC perform store and auto-increment/decrement functions as follows:

FCD0 just store at current position
FCD1 store and inc Y
FCD2 store and dec Y
FCD3 store as byte at current Pos
FCD4 store and inc X
FCD5 store and inc X,inc Y
FCD6 store and inc X,dec Y
FCD7 store as byte,and inc X
FCD8 store and dec X
FCD9 store and dec X,inc Y
FCDA store and dec X,dec Y
FCDB store as byte and dec X
FCC4 scroll latch (port A of PIA)

The pattern contained in the register stored at the particular location above may be varied to produce solids or various forms of dotted or dashed lines on the screen.

The current position is established by directly accessing the VRAM with a dummy read. Subsequent use of the special locations then work from that position on the screen in various directions.

The byte mode operations are provided for the writing of alpha-numeric data to the screen. The VRAM arrangement explained above means that the fastest way to place characters on the screen is to write a series of horizontal slice bit pattern to consecutive vertical locations. Note that byte mode operations only work in the vertical (x) direction. This direction requires the actual VRAM address to be advanced by 64 bytes. Byte mode in the (y) horizontal direction is achieved by storing directly into successive locations in VRAM.

All non-byte operations store one point on the screen. What is written to this point (0 or 1) depends on what is in the corresponding bit position of the accumulator used to store the data.

Dot clock Generation

All timing is derived from a crystal oscillator and a counter chain.

The dot clock is constucted around inverter F2 and the 10.38MHz crystal. The string of counters at F9 to F11 and E11 to E13 count the dotclock DOTCLK and with the aid of SROM produce all video related signals.

The screen area is organized as 84 bytes on the line and 304 lines in the frame. The actual displayed area is 64 bytes on the line and 256 lines. One byte time is 8 DOTCLK pulses or 770ns.

The SROM prom at E10 generates the horizontal sync pulse and an advanced blanking signal. The horizontal blank signal (HBLNK) is the 7th bit of the horizontal byte count. The horizontal sync pulse lasts for 6 byte times and starts at the 68th byte position on the line. The vertical sync is generated by gates in E9, F6, D13 and E11. This pulse starts at line 272 and lasts 4 lines.

Gate in E11 generates the last line count (LASTL) at line count 304 and resets the entire counter chain to zero, the first byte and first line of the displayed area.

Video FIFO and Shift Register

(refer schematic Q219-02)

To compensate for the fact that bytes from memory need to be fetched at greater than 1 MHz, (1.185MHz i.e., 64 bytes need to be displayed in one active line time of 54uS) for displaying, a FIFO register (first-in, first-out) is used. Data from the VRAM (MO1-MO8) is latched by octal latch B9. When FIFO input is not full, the IR signal causes a data request to be clocked through flip-flop E6, which in turn generates a shift in (SI) pulse to the FIFO. This occurs every microsecond until the FIFO input queue is full, as indicated by IR returning to a logic zero.

Data is shifted out of the FIFO to the parallel-in serial-out shift register F7 by SO (Shift Out) pulses coming from the bit counter chain and SROM. They start one byte time before the line is unblanked and stop one byte time before the line is blanked. One byte is transferred from the FIFO to the shift register every eight picture bits. The picture information is shifted out by clock pulses from the bit rate oscillator (DOTCLK).

Video Output

Video data from the shift register is EXCLUSIVE-ORed with the INVERT signal from the PIA at D, E4. With a logic 1 at this pin the picture appears normally with set bits in the VRAM displayed as bright pixels. If a logic zero is applied, the picture is inverted (negative) so that ones in VRAM appear as black pixels on a BRIGHT background.

The output from this exclusive-OR gate passes through another, and is gated with the lightpen cursor signal INV.

The output from this gate is serrated at the bit rate by ANDing with DTCLK* at F6. This is done to avoid horizontally adjacent dots merging to appear brighter than vertically adjacent ones.

The serrated video is then blanked by gating with the combined horizontal and vertical blanking signals by F5 and F6. This removes unwanted (inactive) display time from the display. Sync is added to the video by combining the horizontal and vertical sync pulses in F5 and resistor network R10 and R9. This is buffered for low impedance driving by transistor Q1. The composite video signal is available at both 10-way connectors at the front of the card.

Address Decoding

The Lightpen/Graphics Card occupies two areas of memory space. The VRAM uses 16K bytes from $8000 to $BFFF. The various auto-incrementing portholes use 15 bytes $FCD0 to $FCDB, and the PIA and TIMER use FCC4 to FCDF. Addresses in the FCCX to FCDX range are decoded by half of NAND gate A2. VRAM addresses in the $8000 range are decoded by the other half of A2. These are enabled by AND gate D3, and gating together VMA, ENBL and ADD1/ADD1* which makes VRAM processor unique.

Either processor can always access the PIA and Timer on the card, if peripherals are mapped in. See Q256 RAM card for explanation of mapping and peripheral control.

The select signals from this gating are latched, along with the five least significant address bits and the read/write line, by octal latch B3.

The 16 auto-increment functions are decoded from the four least significant address bits by dual one-of-four selector C3.

Data Buffers

Octal buss Buffers B8 and transceiver B7 interface the card to the system data buss. Buffer enables and control signals are generated via the buss control prom BROM. The buss control prom is used to simplify the complicated enabling of the buffers as they must be enabled at several different addresses and modes of operation of VRAM access.

Addressing counters

The address being accessed within VRAM is determined by the state counter chain B4, B5, B6, A4 and A5. These are up-down counters which provide the auto-increment (and auto-decrement) functions. The starting address for any operation is established by any access to the VRAM. This is achieved by the counters being parallel-loaded by ADLD (address load) that occurs when a read is made between $8000 and $BFFF, in a map that contains the graphics ram.

Horizontal incrementing or decrementing is achieved by pulses YUP and YDN clocking the low-order 9 bits of the counter chain. The 3 lowest bits are used to select the bit within the byte. Vertical incrementing or decrementing is achieved by clocking the highest 8 bits of the counter.

Bit Selection

The 16K byte of VRAM is bit addressable, that is each row of chips can be individually written to.

For Byte mode operations, the BYEN signal is TRUE, and when a memory write is performed, the WROM A6 (write control prom) activates all the write lines causing the whole byte to be written irrespective of the current bit position within the byte.

For Bit mode operations, only the bit determined by the contents of A5 will have its W* signal activated .

Vertical Scrolling

(refer schematic Q219-05)

Counter chain D5, D6 and D8 generate the 14 bit address for the VRAM display operation. Each time a byte is fetched into the FIFO an INCA (Increment Address) pulse clocks the counters on to the next byte. The starting line is preset into counters D5 and D6 during line 304 by signal SYNR* produced by F13 and SROM. (This also resets the FIFO and allows it to be filled with the data for the first displayed line before the line is unblanked. This prefilling is necessary as the FIFO is emptied faster than it can be filled, during the active part of a line.)

The starting line number comes from port A of the PIA. Using this side of the PIA allows the starting line to be read from the latch directly via software.

VRAM Address Multiplexing

(refer schematic Q219-06)

Quad data multiplexers C6 to C8 multiplex the addresses of the VRAM between processor accesses (addresses from counter chain B4, B5, A4, B6 and A5) and video display addresses (counter chain D5,D6 and D8).

Switching is done by BADD2X signal which is derived from phase two of the processor to which access has been enabled.

Light Pen Introduction

(refer schematic Q219-04)

The Light Pen is used to generate co-ordinate data from Hit and Touch signals coming from the light pen. These come onto the card via the lower 10-way connector.

he video chain when a touch and hit occur from the lightpen. The co-ordinates are then available to be read from the card by the processor.

pen, which generates a Touch signal. The interface can be configured to generate an interrupt when the touch signal is activated.

The card can also be configured so that the Hit signal causes the picture information to be inverted, which generates a "cursor" on the display at the position the pen is currently "seeing".

The Hit signal from the pen is de-glitched by the interface so that random noise due to external interference will not cause false triggering.

Hit Deglitcher

Deglitching operates by ensuring that light pen hits are repeated a minimum number of times on successive display lines.

Counter E2 is normally in its terminal count state, counting being inhibited by RCO being high, which forces P low. A Lightpen Hit pulse arriving at gate D1 will RESET the counter via its clear input, CLR. This will clock the bit and byte latches via LPBITS through gate D3 and flip-flop E6 after being synchronized with the valid bit clock.

The counter then procedes to count line blanking pulses, HBLNK. Once two lines has been counted, the next Hit pulse will cause a logic 1 to be clocked by flip-flop D2. On the next byte parallel load, PLOAD*, this is propagated to the second flip-flop D2, generating a LPSTB pulse to clock the line latch. If no Hit occurs during the third line after the first hit, the count continues to its terminal state (16) without a LPSTB being generated.

Hit and Touch Receivers

The Hit and Touch lines from the lightpen are TTL compatible signals, active low. They are terminated at the receiving end by resistor networks R1, R2, R6 and R7. Depending on the state of the mode bit TFH the Touch signal may be ANDed with the Hit signal to generate a valid Hit pulse.

Note that gating is arranged to disable the cursor when the Touch is asserted. This is to ensure that the pen has something to "see" when activated.

The lightpen has its own supply regulator to reduce noise sensitivity. This is provided by a 78L05 device.

Co-ordinate Latches

(refer to drawing Q219-03)

The current bit within the byte is counted by F9. The byte on the line is counted by F10 and F11. Their contents are latched by D9 and D10 when a valid hit occurs by LPBITS, (deglitched Hit).

The least significant bit within the byte is available at bit 7 of port B of the PIA. The rest are read by the processor when LPL1* is active.

The line in the frame is counted by E13 and E12 and latched at D12 by LPSTB. This octal latch is read when LPL2* is active.

Control PIA

(refer schematic Q219-01)

Peripheral Interface Adapter D, E4 provides read/write ports for a variety of different functions. The side A is configured as outputs and are used to control the screen scrolling. The B side is input and output and controls screen inversion, processor access to VRAM, TFH (touch for hit) mode, lightpen cursor enable, lightpen touch status and the least significant bit within the byte counter.

The PIA is also used to generate interrupts. Two separate interrupt outputs are possible, one for Touch and one for Hit. These are generated by the PIA CA1 and CB1 outputs respectively.

Timer

An M6840 Timer is provided for general system use. This device contains three independent 16-bit counters which can be configured under software control to count external events or internal clock pulses. Counter 1 is clocked by HBLNK to count lines, counter 2 is clocked by VSYNC to count frames, and timer 3 is clocked by a 1MHz system timing signal to count microseconds.

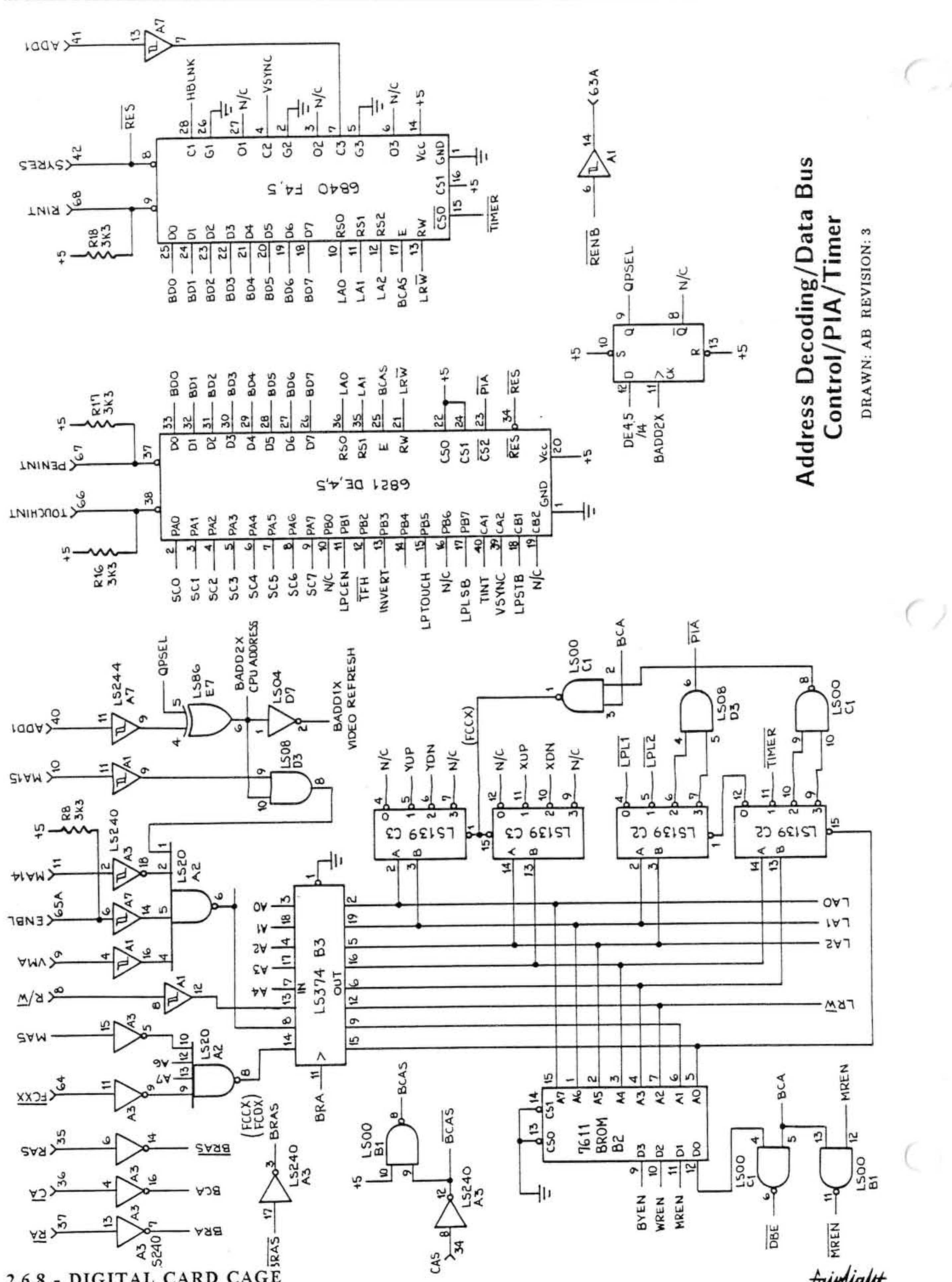
For full operational specifications of the 6840 refer to manufacturers data sheets.

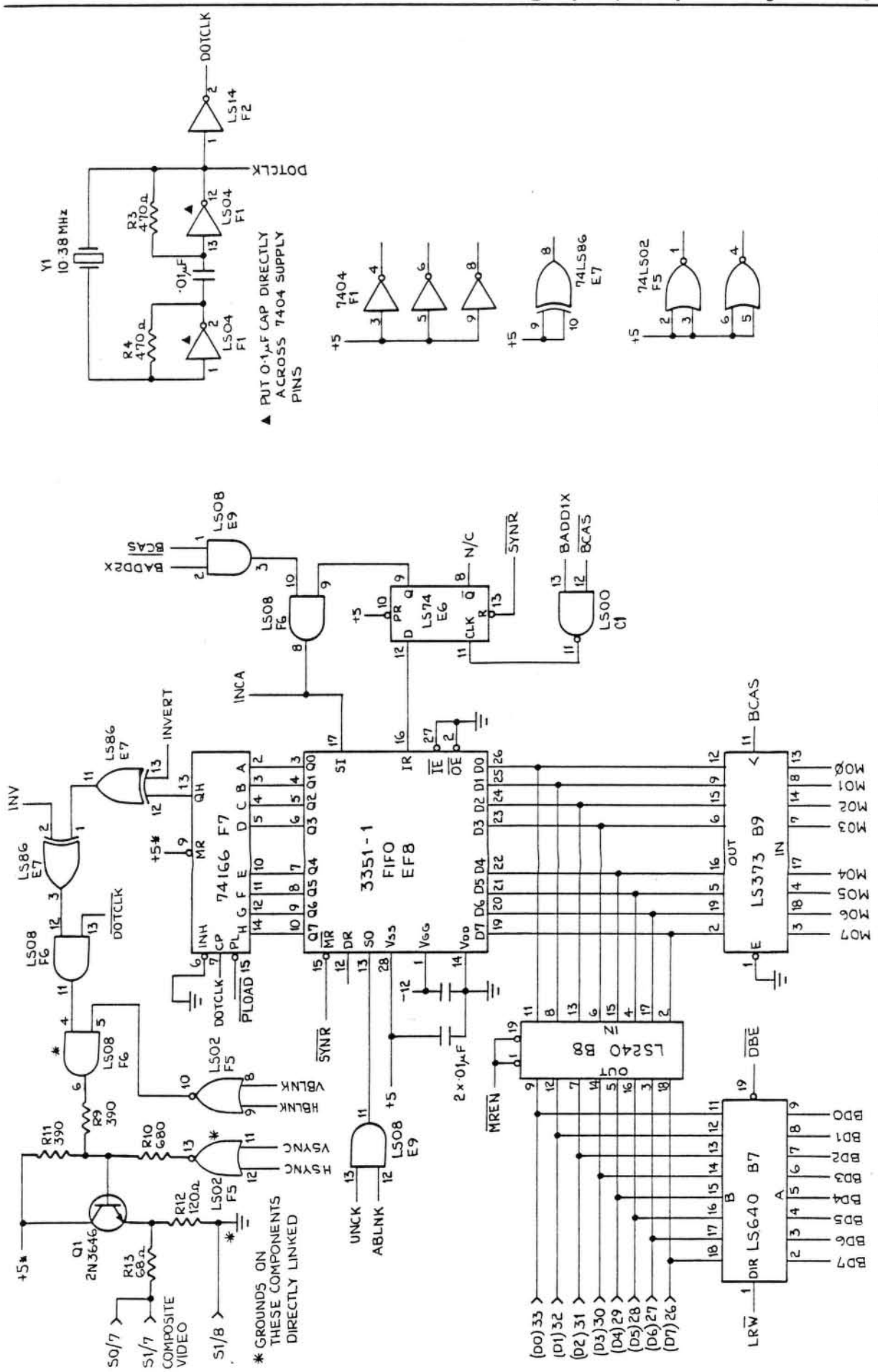
Video Memory Ram

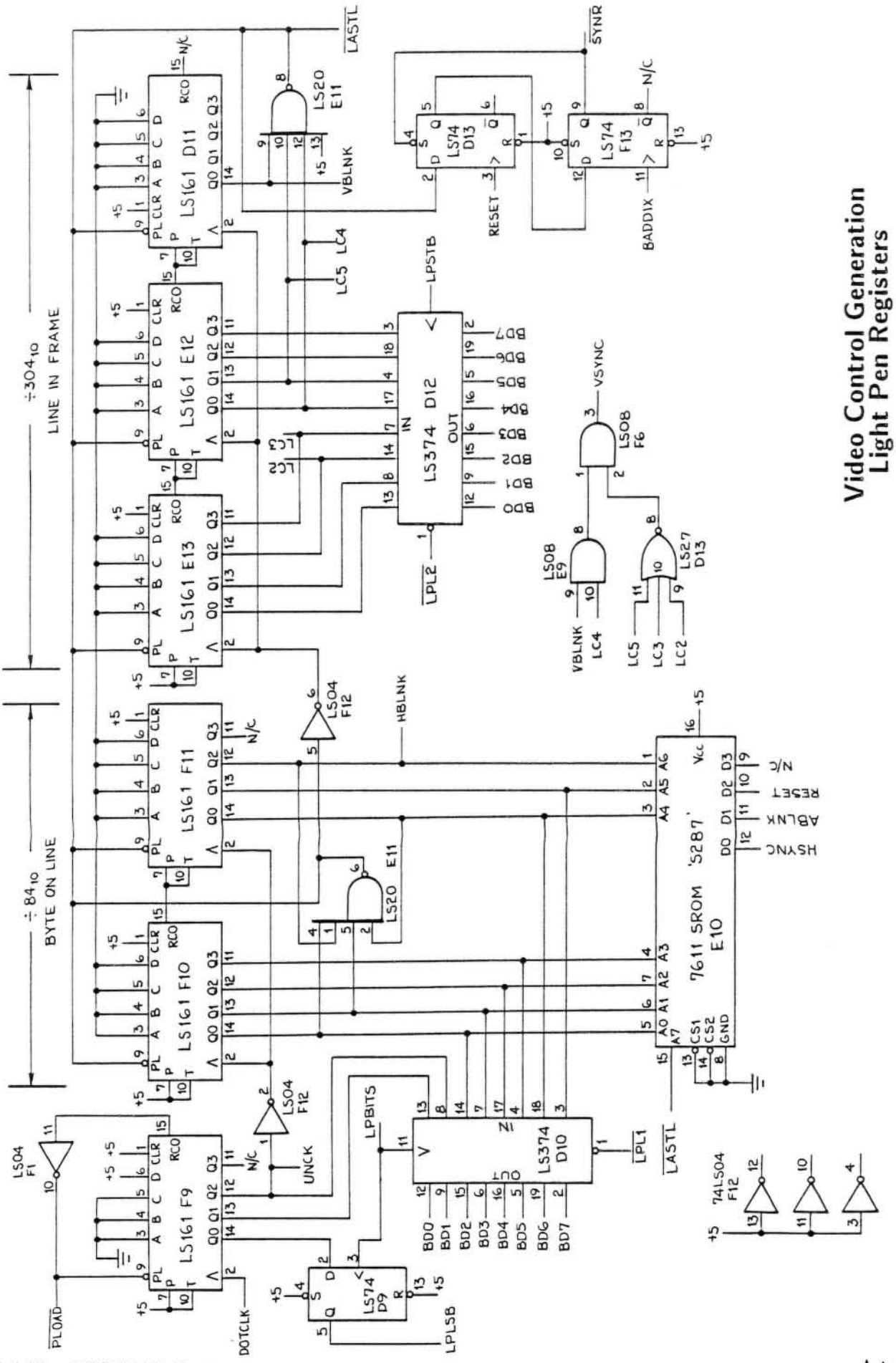
(refer to drawing Q219-06)

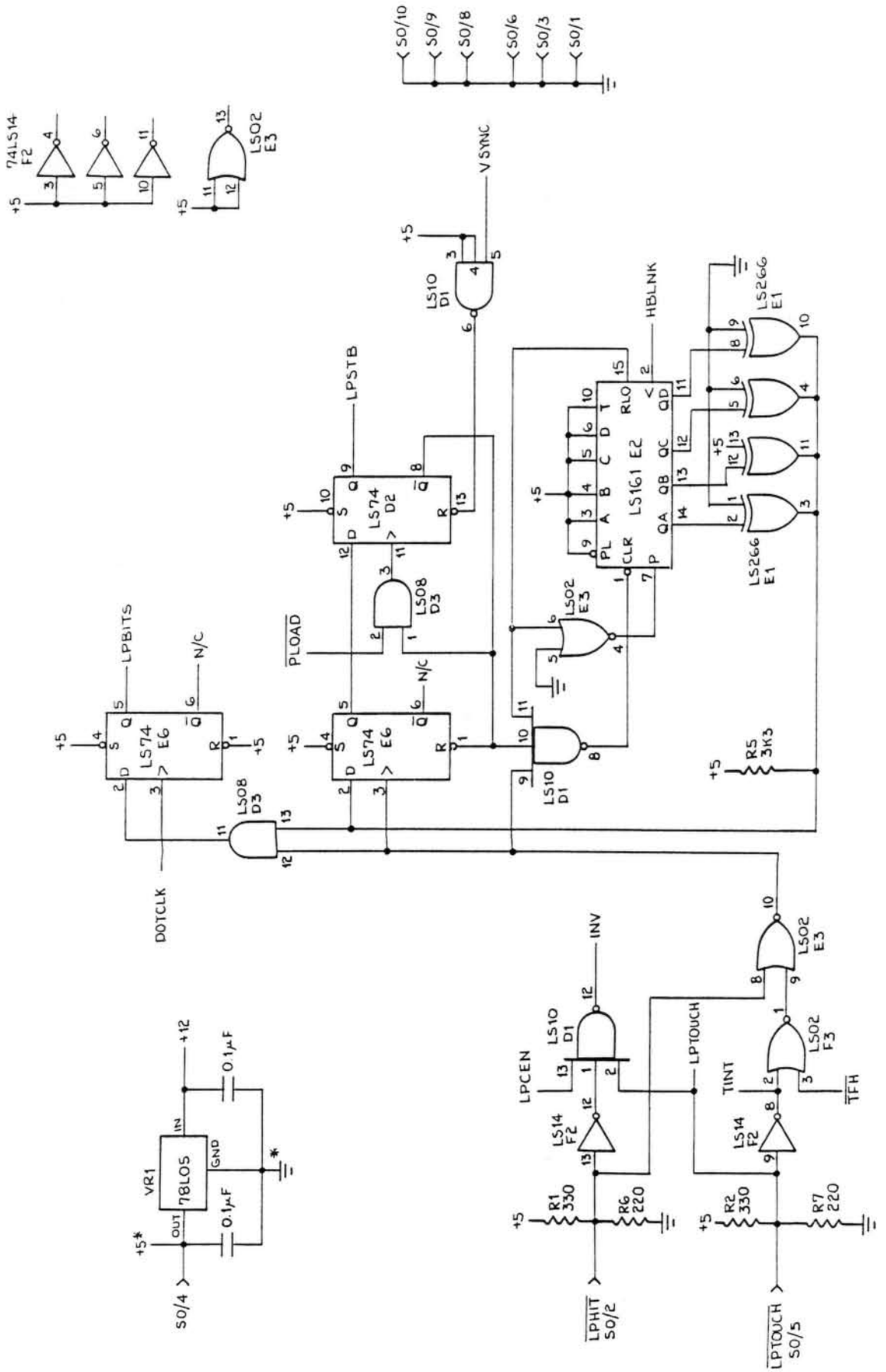
The 16 kilobytes of VRAM is provided by eight 4116 16,384 bit dynamic MOS devices. The timing required for these devices is preset in the system timing signals. These signals are buffered and fed directly to the rams as $\overline{\text{BRAS}}$, $\overline{\text{BCAS}}$ and $\overline{\text{BCA}}$. Memory refresh is done in the continuous access for screen refresh.

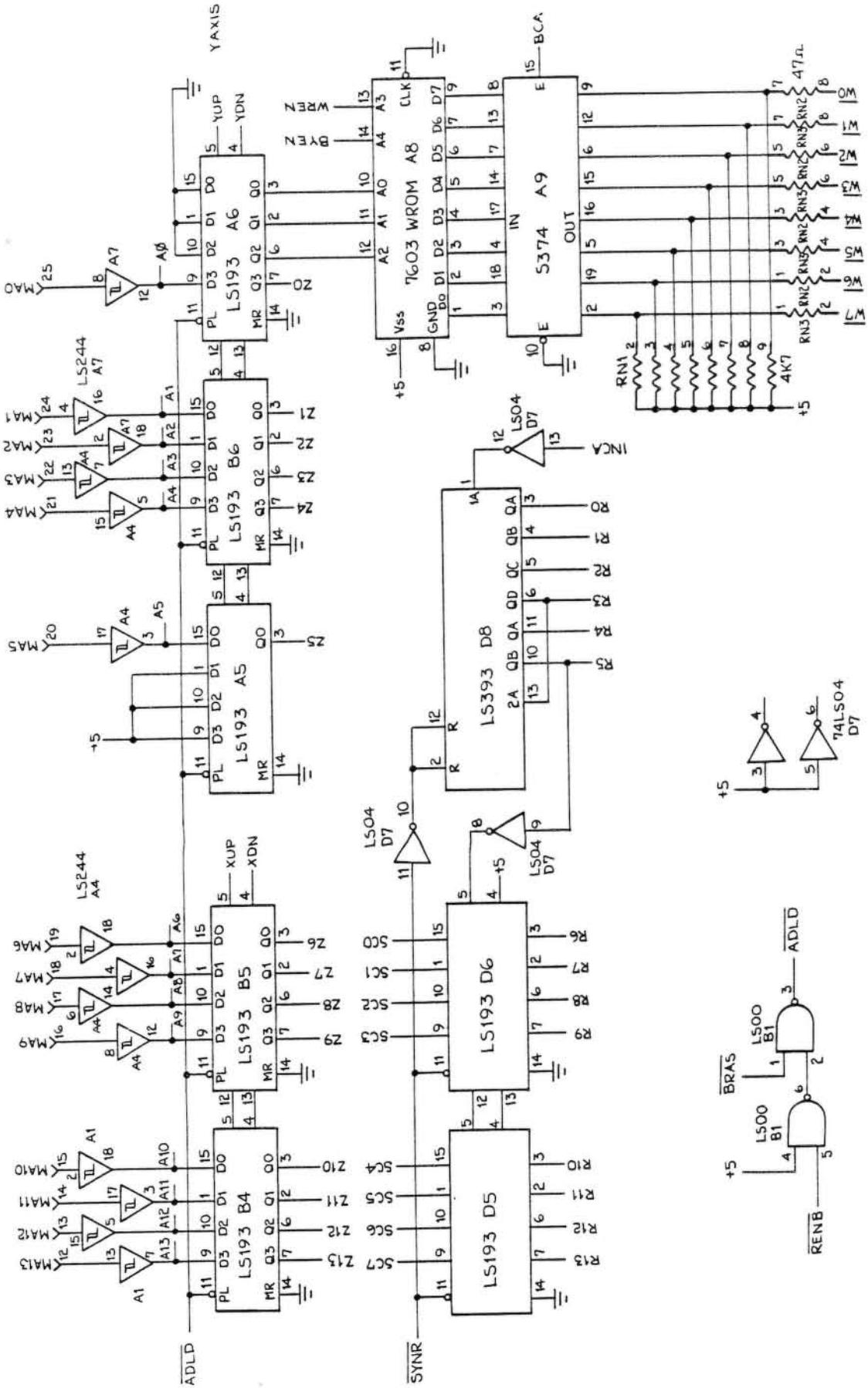
Whenever the VRAM is directly accessed, the decoding at A2, 7D, 7E generates RENB, (active low). This disables the memory card in the 16K block used by VRAM and saves duplicating maps in the Q256 ram card.


Address Decoding/Data Bus
Control/PIA/Timer
DRAWN: AB REVISION: 3


Master Clock, Video Output
Data Bus Buffers
DRAWN: AB REVISION: 3
Y1 10.38 MHz
R3 470Ω
R4 470Ω
.01μF
LS04 F1
LS14 F2
DOTCLK
▲ PUT 0.1μF CAP DIRECTLY ACROSS 7404 SUPPLY PINS
7404 F1
74LS86 E7
74LS02 F5
LS08 E9
LS08 F6
LS74 E6
LS00 C1
BADDRX
BCAS
BADDIX
SYNR
N/C
INCA
LS86 E7
INVERT
INV
74166 F7
3351-1 FIFO EF8
LS373 B9
LS240 B8
LS640 B7
DBE
MREN
LRW
R11 390
R9 390
R10 680
R12 120Ω
R13 680Ω
Q1 2N3646
LS02 F5
VBLNK
HBLNK
VSYNC
HSYNC
UNCK
ABLNK
2 x .01μF
COMPOSITE VIDEO
S0/7
S1/7
S1/8
* GROUNDS ON THESE COMPONENTS DIRECTLY LINKED
(D0) 33
(D1) 32
(D2) 31
(D3) 30
(D4) 29
(D5) 28
(D6) 27
(D7) 26
BD0
BD1
BD2
BD3
BD4
BD5
BD6
BD7
M00
M01
M02
M03
M04
M05
M06
M07


Video Control Generation
Light Pen Registers
DRAWN: AB REVISION: 3
÷304 10 LINE IN FRAME
÷84 10 BYTE ON LINE
LS161 D11
LS161 E12
LS161 E13
LS161 F11
LS161 F10
LS161 F9
LS374 D12
LS374 D10
LS374 D9
LS574 D13
LS574 F13
LS20 E11
LS08 F6
LS08 E9
LS27 D13
LS04 F12
LS04 F1
7611 SROM '5287' E10
74LS04 F12
LASTL
SYNR
VBLNK
LPSTB
LPL2
LPL1
RESET
BADDIX
VSYNC
HBLNK
LPBITS
UNCK
LPLSB
DOTCLK
PLOAD
LC5 LC4
LC3 LC2
BD0 BD1 BD2 BD3 BD4 BD5 BD6 BD7
HSYNC
BLNK
RESET
N/C

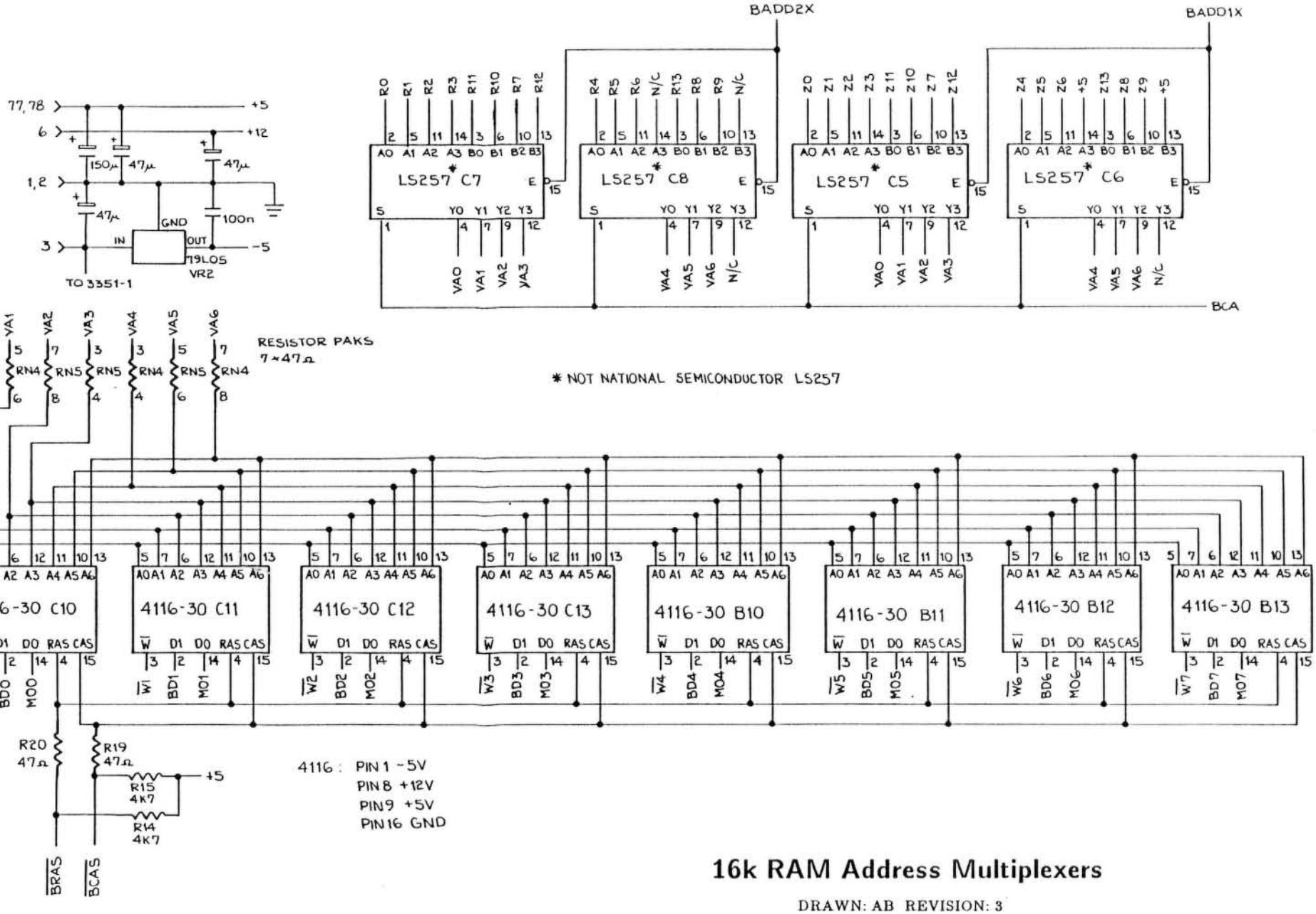

Lightpen Interface
DRAWN: AB REVISION: 3
74LS14 F2
LS02 E3
V SYNC
LS10 D1
LS266 E1
LS161 E2
HBLNK
LPSTB
LS74 D2
LS08 D3
PLOAD
LPBITS
N/C
LS74 E6
R5 3K3
DOTCLK
INV
LPCEN
LPTOUCH
TINT
LS02 F3
LS14 F2
R1 330
R6 220
R2 330
R7 220
VR1 78L05
0.1µF
+12
LPHIT S0/2
LPTOUCH S0/5
TFH
S0/4
S0/10
S0/9
S0/8
S0/6
S0/3
S0/1



VRAM Addressing Write Control Logic

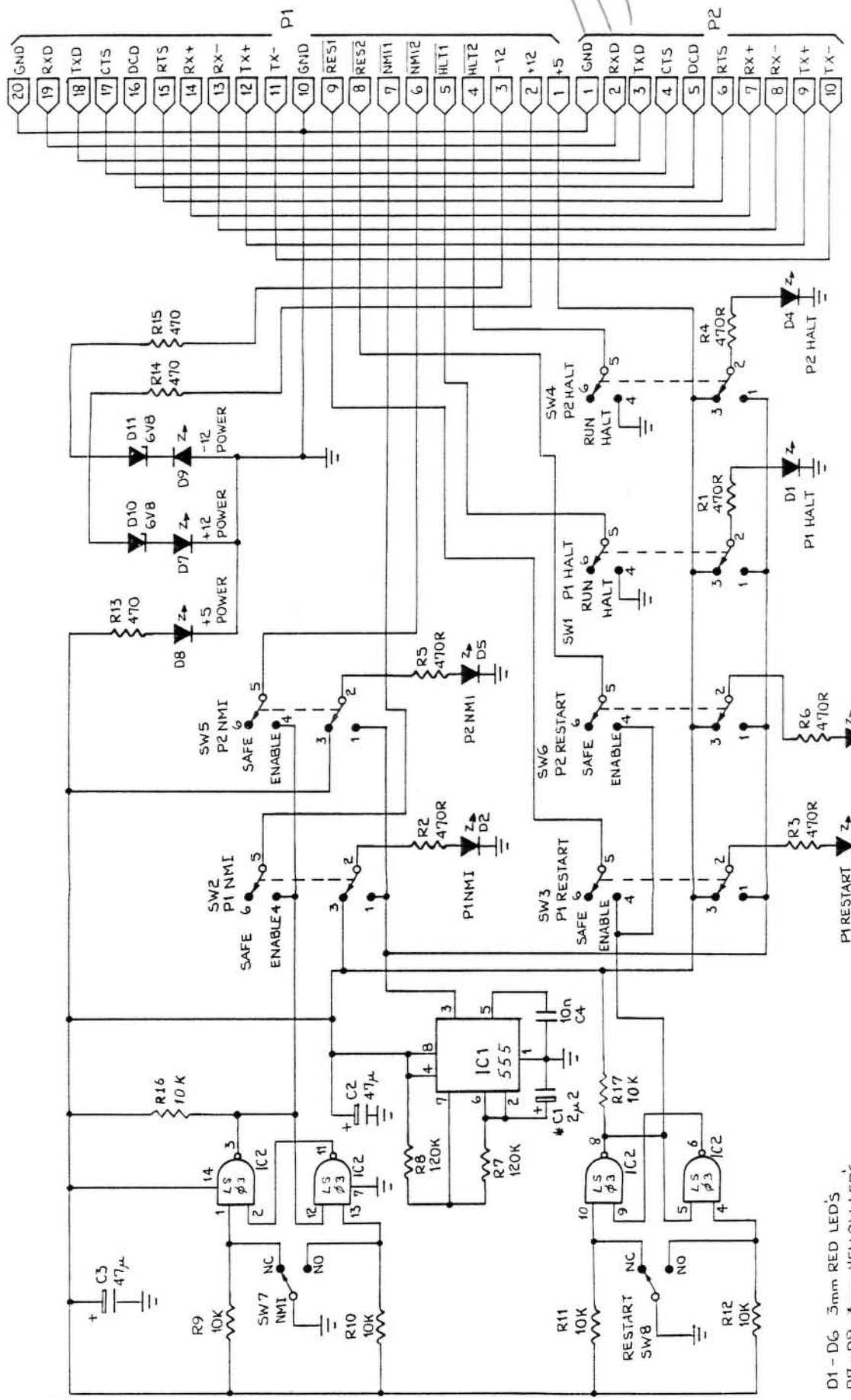
DRAWN: AB REVISION: 3

fairlight



16k RAM Address Multiplexers

DRAWN: AB REVISION: 3



Front Panel Controls

DRAWN: PV REVISION: 2

Q137
Front Panel Control

2.7

Introduction

This board contains the controls used for system debugging and reset. It contains 2 push button swithes, one for RESET and one for NMI.The remaining toggle switches activate halting of each CPU and enabling of the appropriate labelled function.

Operation

The 555 timer is wired as a low frequency oscillator and is used to flash the LED associated with each "armed" switch.

The CMOS NAND gate is used for debouncing the pushbutton switches.

Each switch has two poles. One activates the function while the other is used to connect the oscillator to the LED.

For a function to operate the associated LED must flash.

Communications Port

The front panel also carries a 10 way flat cable connector, P2, that contains serial communications lines from the Q133 card.

Q777
SCSI Interface

2.8

Introduction

The SCSI Interface card adapts the CMI interleaved parallel buss to a SCSI peripheral interface buss which is a high speed 8 bit parallel buss with handshaking. The interface card also contains an EPROM with appropriate driver software which the CMI operating system can load into RAM.

The SCSI Interface contains hardware to implement either Initiator operation or Target operation (as defined in the SCSI specification) under software control. DMA transfers of data between the SCSI buss and CMI memory are implemented by the interface for Initiator operation. In this mode a hard disk or similar device will be selected for either reading or writing and the data transfer will proceed at the maximum rate. The SCSI Interface may be configured to have a SCSI address in the range of 0 - 7 and an external select of this device will generate an interrupt.

The SCSI Interface will normally be used to connect to storage devices such as hard disks and streaming tapes, however any device which conforms to SCSI specifications may be attached to the interface. All communications with SCSI devices attached to the buss conform to a defined message protocol which transfers command, data and status information in distinct phases as defined in the SCSI specification.

Address Map

The Interface is accessed through two locations, FCE2 and FCE3 in a memory map which enables access to peripherals. A 3 bit address register (FCE3) is written to point to additional registers. All access to these registers is subsequently performed through location FCE2.

ADDRESS	READ	WRITE
FCE2	DATA	DATA
FCE3	STATUS	POINTER

Status Register

A read from address FCE3 will return the contents of the status register. This register indicates interrupt status, EPROM status and SCSI buss state.

bit 7 - high indicates EPROM transfer not occurring
bit 6 - high indicates an interrupt condition is present
bit 5 - MESSAGE active on SCSI buss
bit 4 - ACKNOWLEDGE active on SCSI buss
bit 3 - COMMAND active on SCSI buss
bit 2 - BUSY active on SCSI buss
bit 1 - REQUEST active on SCSI buss
bit 0 - DIRECTION active on SCSI buss

Pointer Register

A write to location FCE3 will set the pointer register to select additional registers on the interface. Only bits 1 - 3 are used by the interface. The state of other bits will have no effect.

Data Registers

There are 10 additional registers which may be addressed throug location FCE2 after setting the pointer register. Eight of these are output registers and two are input registers.

POINTER	OUTPUT	INPUT
00	control register	input register
02	DMA address low	X
04	DMA address high	X
06	DMA count low	X
08	DMA count high	X
0A	select EPROM load	X
0C	SCSI data out	SCSI data in
0E	target register	X

Control Register

This register controls DMA operation, interrupts and SCSI operations.

bit 7 - enable interrupts to the CMI
bit 6 - enable DMA operation
bit 5 - enable SCSI buffers
bit 4 - disable auto-increment on DMA address counters
bit 3 - SCSI ATTENTION
bit 2 - SCSI BUSY
bit 1 - SCSI SELECT
bit 0 - SCSI RESET

Input Register

This register provides information about the current state of the SCSI interface and also identifies the preset SCSI address of the card.

bit 7 - interrupt present due to SCSI select
bit 6 - bit 2 of SCSI address
bit 5 - bit 1 of SCSI address
bit 4 - bit 0 of SCSI address
bit 3 - SCSI ATTENTION
bit 2 - SCSI BUSY
bit 1 - SCSI SELECT
bit 0 - SCSI RESET

DMA Address Low
Low byte of DMA start address.

DMA Address High
High byte of DMA start address.

DMA Count Low
Inverted low byte of DMA count.

DMA Count High
Inverted high byte of DMA count.

Select EPROM
A write to this register will enable the logic which loads the contents of the EPROM into CMI RAM. It is necessary to set up the DMA logic before writing to this location.

SCSI Data Out
This register drives the buffers onto the SCSI data lines.

SCSI Data In
This register contains the current state of the SCSI data lines.

Target Register
This register controls the hardware implementing SCSI target operation. In normal operation this register is cleared on reset, disabling all target functions.
bit 7 - MODE active enables Target operation
bit 6 - not used
bit 5 - SCSI MESSAGE
bit 4 - SCSI DIRECTION
bit 3 - SCSI COMMAND
bit 2 - not used
bit 1 - SCSI REQUEST
bit 0 - not used

Address Decoding
Address range $FCE2 - $FCE3 is decoded by gates A1, B3, A4, B1, B2, D1, E1 and latched by F2. NAND gates E2, E2 are used to generate read and write strobes for the status and pointer registers respectively.
Bits 1, 2 and 3 from the pointer register E5 are decoded by E4 and strobed by signal DREG* from F3. Where necessary, select lines are gated with read or write signals by OR gates B8, F3.

Data Buffers
Data from the system data bus is buffered and inverted by C6 then latched by D6 which holds the data across the processor 1 phase. The latched data from D6 (I0-I7) is used by all output registers and DMA counters.
Data read onto the system data bus is taken from input registers A5, C9, C8 and C7 via data bus D0-D7.

DMA Address Counters

Sixteen bit counter chain D3, C3, D2, C2 is used to provide the address for DMA transfers. The starting address for each DMA transfer is established by writing the appropriate byte address to the pointer register followed by writing the DMA address byte to the data register. This is repeated for the other DMA address byte. When DMA transfers take place, the counter chain is automatically incremented by the signal ATB*. DMA address incrementing may be disabled under software control via bit 4 of the control register and gate F4. This feature allows disk reads and writes from a single memory location as implemented on the Floppy Disk control card.

DMA Byte Counters

Sixteen bit counter chain D5, C5, D4, C4 is used to transfer the required number of bytes to or from CMI memory under DMA control. The counters are configured to count up to $0000, so for correct operation it is necessary to load the inverse of the required count. Any number may be specified up to a maximum of $FFFF bytes. Only the specified number of bytes will be transferred to or from memory, regardless of the actual size of the transfer requested by the SCSI bus. This allows less than a sector to be read from disk saving the software overheads necessary for partial sector reads. When the counter reaches 0, the signal FINPS* is generated from ripple carry out. This signal is buffered out to the system bus as VMA by 3 state gate C1 disabling further memory accesses.

DMA Logic

DMA requests may come from either the Device Driver ROM logic or the SCSI interface. These requests are synchronised with processor 2 phase 2 by flip-flop A8. This sets up a DMA request to the processor (RDMA*). DMA cycles are granted by the BACK acknowledge signal on a daisy chain basis through ETL* and ENL*.

When a transfer occurs, the DMAC (DMA CLAIM) line is asserted so that the memory card swaps maps, allowing data to be dumped into memory currently not mapped into the processors address space.This signal is generated by flip-flop B10 and gates A10, A10. During a DMA cycle, onboard select strobes are generated by flip-flop B9 and gates A9, A10. During DMA writes either ROM buffer A5 or SCSI buffer C7 will be enabled depending on the state of flip-flop B10. DMA reads will always be directed at the SCSI port. NAND gate E2 generates the address strobe ATB* which enables the address buffers onto the CMI buss.

SCSI Bus Interface

The SCSI bus is physically a 50 way ribbon cable where each signal is terminated by a 220/330 ohm resistor network. The interface is capable of driving and receiving all SCSI signals except PARITY which is not implemented. Open collector NAND gates E6, E7, E8, E9 and D10 are used as drivers. The SCSI data bus is buffered by C7, and the remaining signals are buffered by hex inverters E10 and E11. All SCSI drivers are gated as three groups:- SCSI data, SCSI control and SCSI target control. These groups are activated by signals EDATA, ESCSI and ECONT which are derived from control latches D8 and D9. Both these latches are reset by SYRES* ensuring that the interface is disabled upon powerup.

Data transfers to and from the SCSI bus are initiated by REQ being asserted. Data is read off the bus by strobing buffer C7. Data written to the SCSI bus is latched by D7 and is held until the next write. This is necessary as SCSI data transfers are asynchronous and data must be maintained on the bus until the target releases REQ. Gates C11 and D11 ensure that data is only enabled onto the SCSI bus as requested by SCSI signal DIR. When a byte is transferred while a request is pending, the strobe sets flip-flop F11 driving ACK to complete the data transfer.

The SCSI interface can generate interrupts from two sources:- status available from target and select from external device. The second of these interrupts is only used in systems which implement multiple initiators, and may be disabled by link W4.

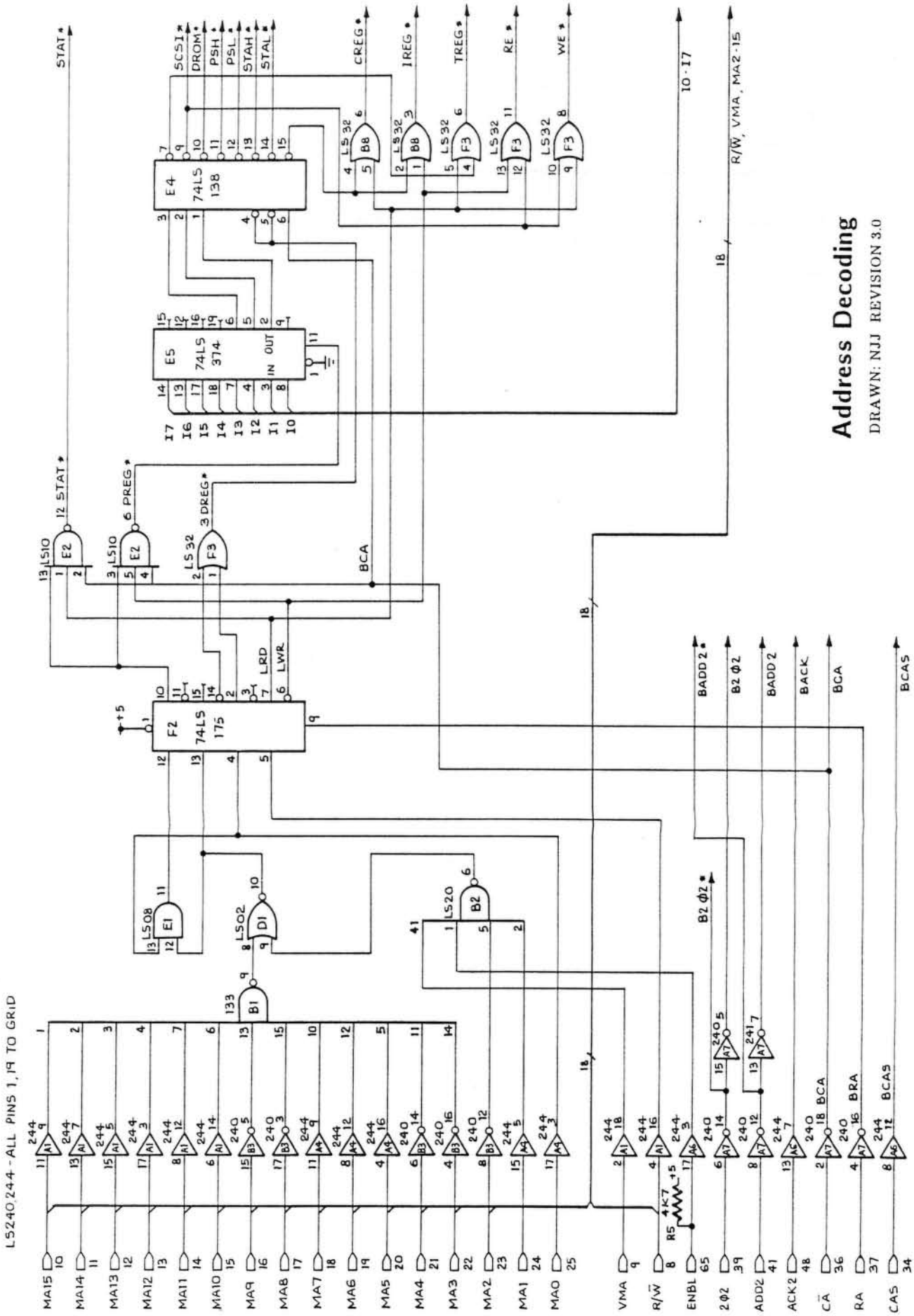
The status interrupt is latched into F11 by the SCSI bus request to transfer a command byte. The latch is enabled by EDMA going high and held clear when this signal is off.

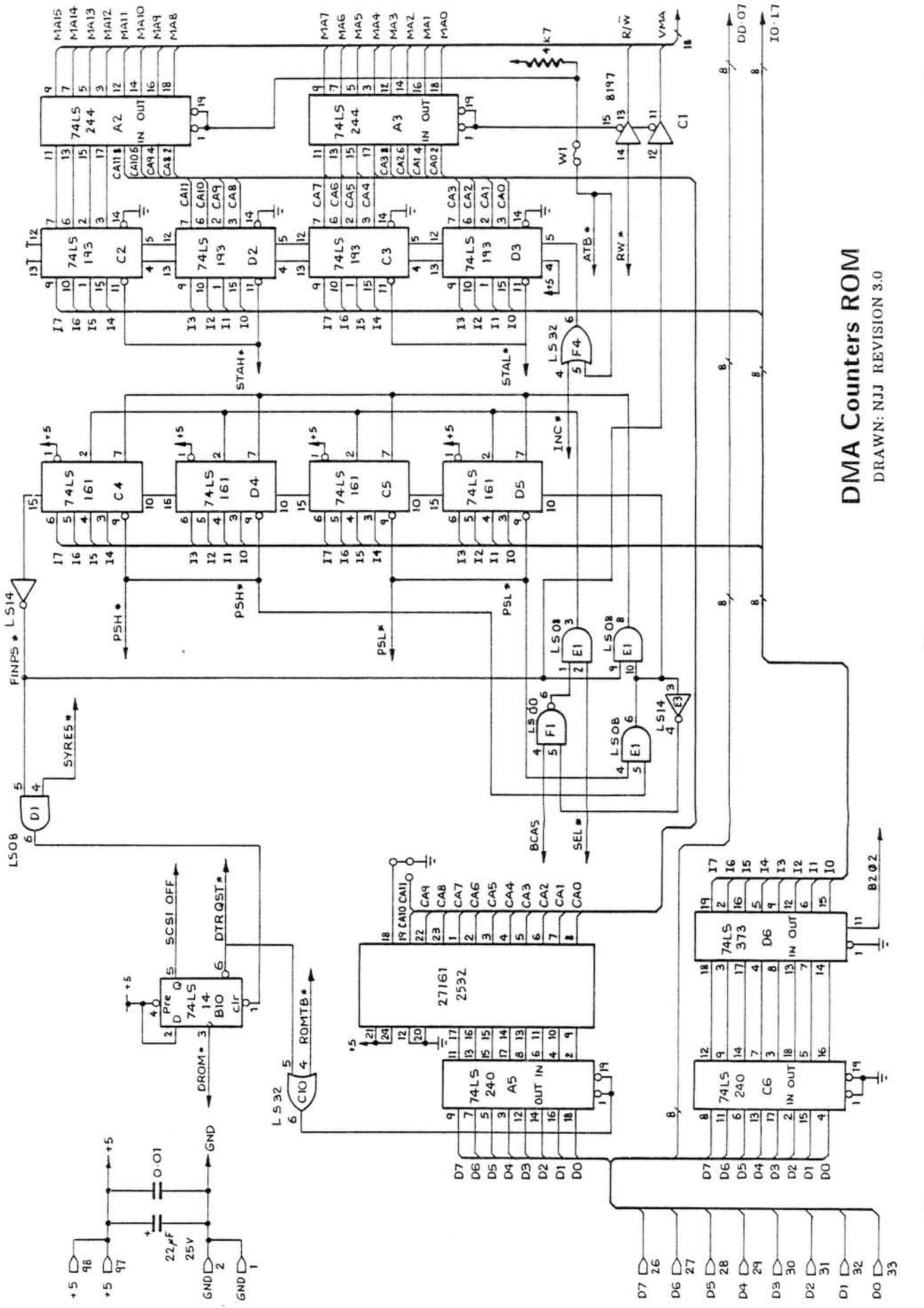
The select interrupt is generated by SEL and the SCSI DATA line set by J0-J2 being simultaneously asserted. B7, C10 and D1 are responsible for this interrupt. Both interrupts are ORed and drive gate D10 onto the CMI buss. Signal EINT is will globally disable all interrupts from this board.

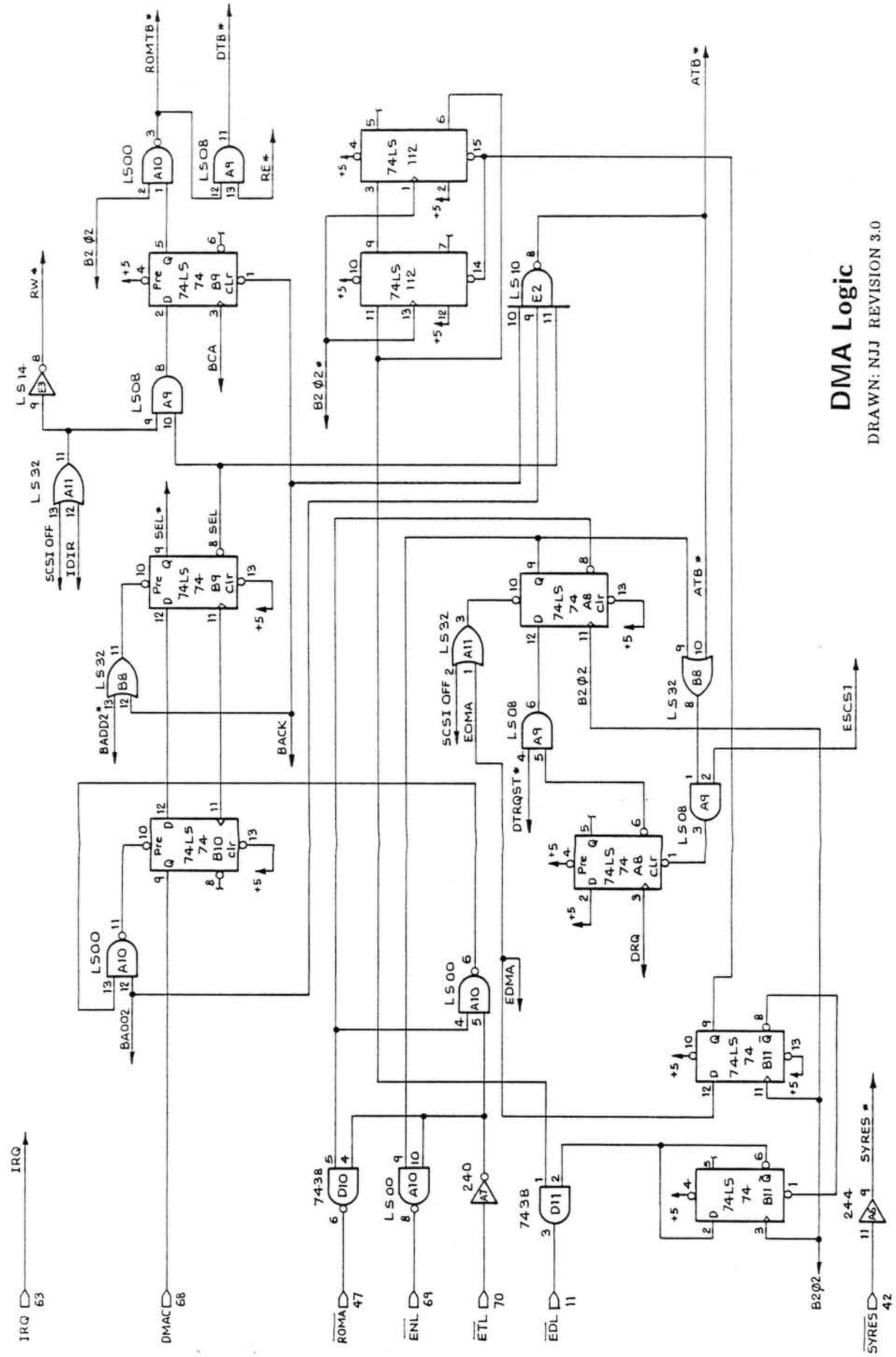
All signals associated with the SCSI interface are read through buffers C8 and C9.

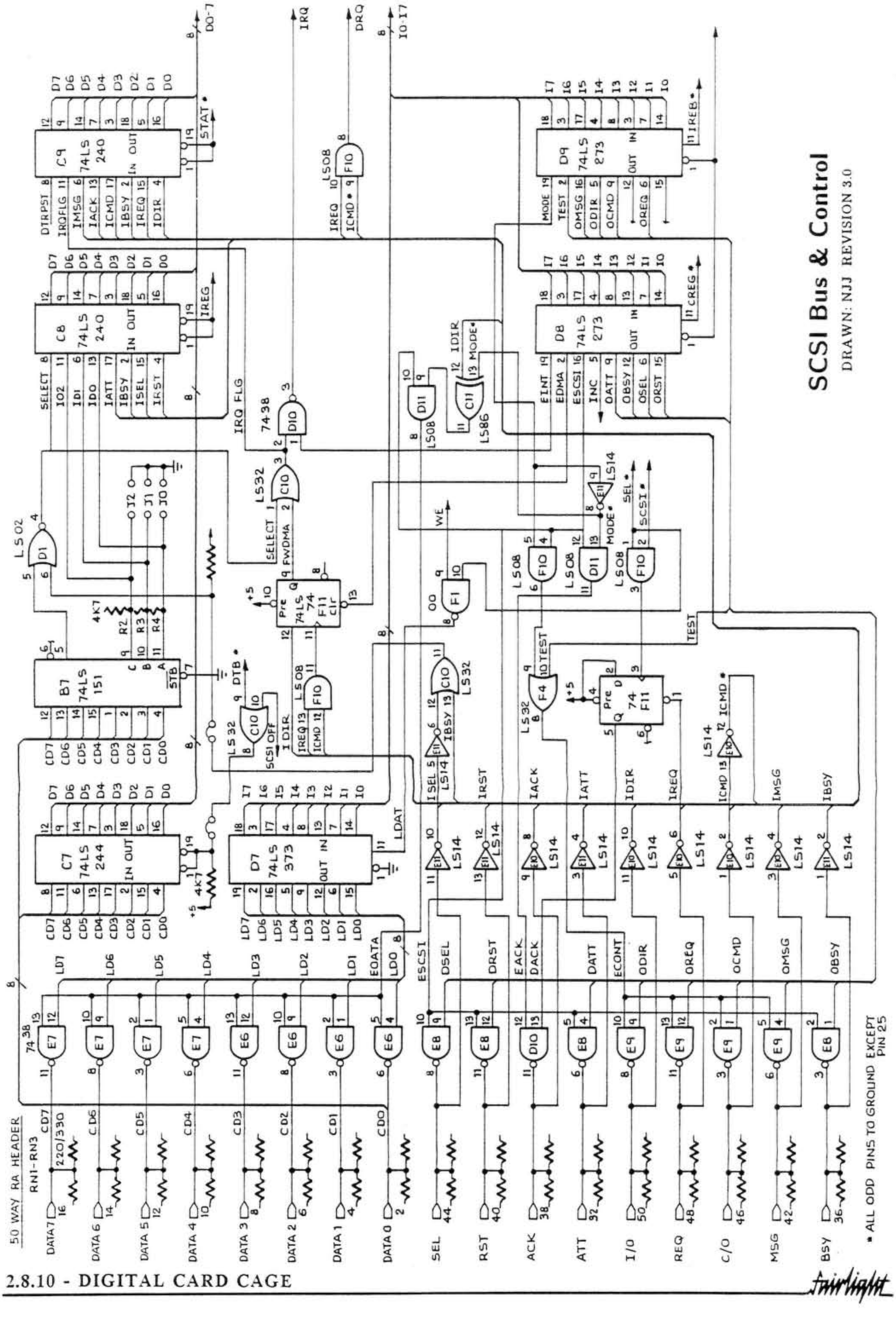
Device Driver ROM

The SCSI controller software may be placed in a 2K or 4K EPROM on the SCSI interface card. This EPROM is not in the processors directly addressable memory - it must be loaded into RAM under DMA control. The least signifigant DMA counter lines are used as addresses on the EPROM, so the EPROM will always be loaded into memory on 2K or 4K boundaries. The DMA transfer is terminated when the byte counter reaches 0000.


Address Decoding
DRAWN: NJJ REVISION 3.0
LS240 244 - ALL PINS 1, 19 TO GRID
STAT*
SCSI*
DROM*
PSH*
PSL*
STAH*
STAL*
CREG*
IREG*
TREG*
RE*
WE*
I0-I7
R/W, VMA, MA2-15
E4 74LS 138
E5 74LS 374
F2 74LS 175
E1 LS08
D1 LS02
B1 133
B2 LS20
E2 LS10
F3 LS32
B8 LS32
12 STAT*
6 PREG*
3 DREG*
BCA
LRD
LWR
+5
BADD2*
B2 Φ2
BADD2
BACK
BCA
BCAS
B2 Φ2*
R5 4K7
MA15
MA14
MA13
MA12
MA11
MA10
MA9
MA8
MA7
MA6
MA5
MA4
MA3
MA2
MA1
MA0
VMA
R/W
ENBL
2Φ2
ADD2
ACK2
CA
RA
CAS

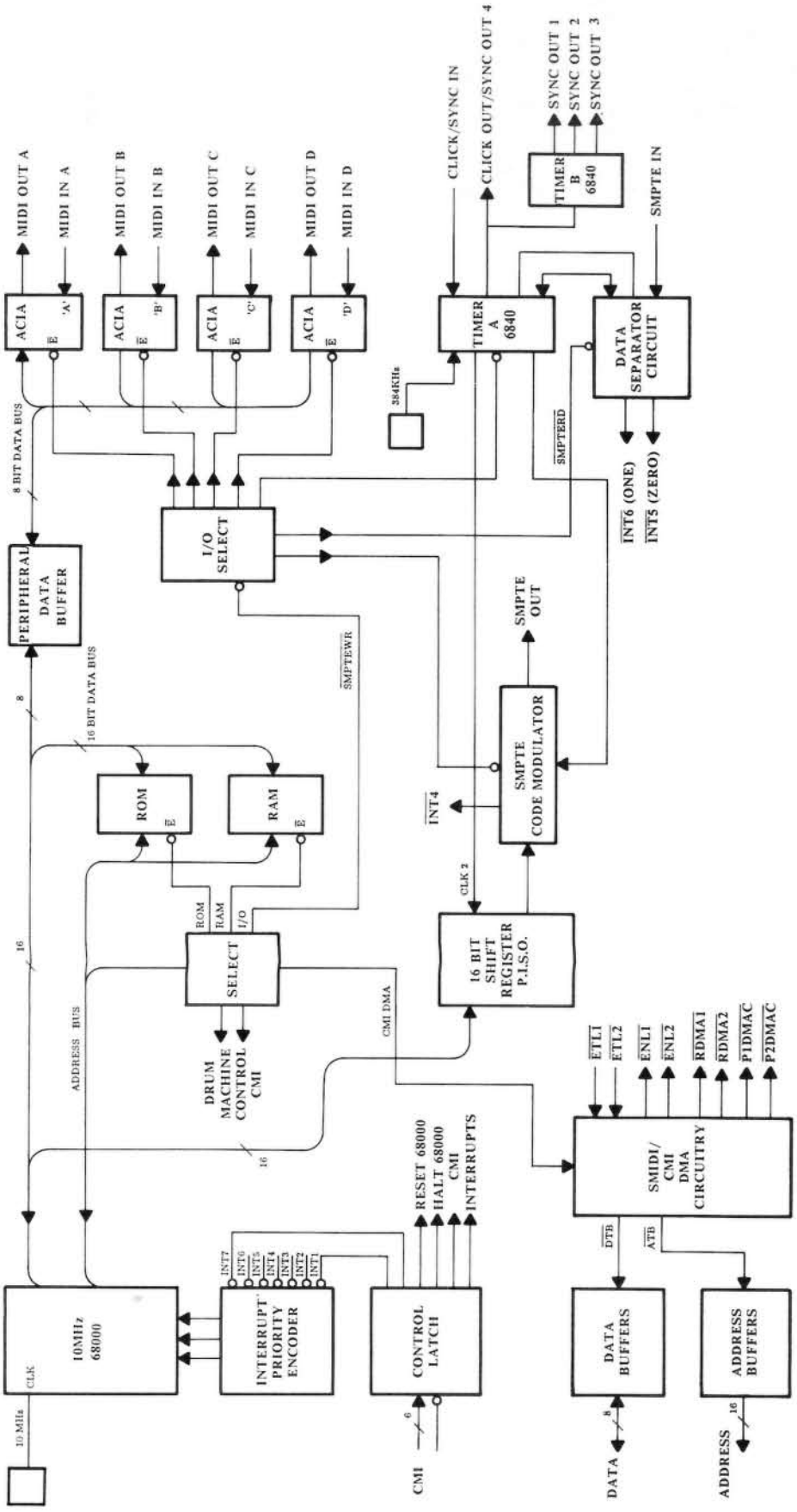

DMA Counters ROM
DRAWN: NJJ REVISION 3.0


DMA Logic
DRAWN: NJJ REVISION 3.0
ROMTB*
DTB*
ATB*
RE*
RW*
B2 Ø2
B2 Ø2*
BCA
SCSI OFF
IDIR
SEL*
BSEL
BADD2*
BACK
SCSI OFF
EDMA
DTRQST*
ESCS1
DRQ
EDMA
BADD2
IRQ
DMAC
ROMA
ENL
ETL
EDL
B2Ø2
SYRES*
SYRES
IRQ
63
DMAC
68
ROMA
47
ENL
69
ETL
70
EDL
11
B2Ø2
SYRES
42
LS00
LS08
LS14
LS32
LS10
74LS74
74LS112
7438
240
244


SCSI Bus & Control
DRAWN: NJJ REVISION 3.0
50 WAY RA HEADER
RN1-RN3
220/330
* ALL ODD PINS TO GROUND EXCEPT PIN 25

CMI-28 2.9
General Interface Card

Block Diagram



Terminology
SMIDI Card: General Interface card or SMPTE/MIDI card.
CPU: Dual 6809 processor (P1 & P2) system usually running OS9.
System RAM: Q256 256K memory with MMU used by the CPU.
68K: Refers to the SMIDI processor - a Motorola 68000.
Hexidecimal numbers are in the form nnnnnnh.
Flip-flops packaged in 74LS74's are referred to as *a* - pins 1 to 7, and *b* - pins 8 to 13.
Titles of the form **CMI-28-x** refer to the circuit diagrams of the SMIDI card, x being the sheet number (1 to 7). Use the diagrams in conjunction with this text.

Introduction

The SMIDI Card plays a central role in the Series III CMI, for all the sequencing is controlled by this card. Of course it has its other function of interfacing the CMI to other digital musical instruments via MIDI and synchronizing sequences to film and video via SMPTE and Click/Sync tracking.

The SMIDI card plugs into slot 18 of the Series III motherboard next to the Waveform Processor Card. A 26-way cable connects the card to two cards (CMI-332, CMI-333) in the Audio Rack.

The **CMI-332** contains the MIDI input and output buffers and the DIN sockets. Drum machine controllers and multi-sync outputs also are on this board.

The **CMI-333** contains the SMPTE analog circuitry (input/ouput) as well as Click In and Out and Metronome Out.

The facilities built in to the SMIDI card are as follows:

- 10MHz 68000 processor
- ROM 8K/16K x 16 bits.
- Private RAM 8K/32K x 16 bits.
- DMA interface to the CPU buss
- 7 bit control latch
- 4 ACIA (68B50)
- 2 PTM (68B40)
- Circuitry for SMPTE reading and generating
- Drum machine controllers

ROMs

The pair of on-board 8-bit ROMs are arranged in parallel to provide 16 bit code. Presently 2764 ROMs (called *KMON-E* and *KMON-O*) are installed, occupying the bottom 8K words of 68K address space (see memory map). The bottom 400h bytes are reserved for 68K exception vectors including reset PC and Supervisor Stack Pointer which are loaded when the processor comes out of reset. The rest of the ROMs are occupied by the 68K monitor, 68K/CPU IO routines, and self tests. Refer to software documentation for further information on these items.

Private RAM

The 8K (or 32K) words of Private RAM on the CMI-28 is accessible only by the 68K. Its base address is 80000h (see memory map).

CPU DMA Interface

The two channels of the DMA interface to the CPU bus appear as two 64K slices of the 68K's 16M memory space. Accesses in the range 40000h to 4FFFFh steal P2 cycles, and select the P2 DMA memory map. Accesses in the range 50000h-5FFFFh steal P1 cycles, and select the P1 DMA memory map. Appropriate initialization of the Q256 memory maps thus allow the 68K to access any physical area of system memory or peripherals whether or not they are mapped into the CPUs' logical spaces. Refer to Q256 documentation for details about system memory management.

Data size mismatch between the 68K and the CPU bus is handled by the hardware. A 16-bit access by the 68K is handled by two separate 8-bit DMA transfers across the CPU bus and the 68K receives one *Data Transfer Acknowledge* ($\overline{\text{DTACK}}$) when both transfers have been completed. Which byte goes first is indeterminate; in any case each byte is always transferred to or from the right place in CPU space.

Access by various devices to the CPU bus is arbitrated by two daisy chains, one for each 6809. Higher priority devices may prevent access by the SMIDI to the CPU bus indefinitely. Refer to a separate document which describes the daisy chain allocation for Series III.

ACIA's and Programmble Timers

The SMIDI card has four ACIA's (68B50) for MIDI input and outputs. They can be access by the SMIDI processor at addresses 60020h to 60050h. The two timers (68B40) can be accessed by the SMIDI processor at addresses 60000h and 60010h.

SMPTE Reading and Generating

To generate SMPTE the 68k processor writes the appropriate SMPTE word to a Parallel-In Serial-Out shift register which is clocked according to the second timer in Timer A which divides a 384kHz clock. A further section of the circuitry reads SMPTE, the timing in this case is controlled by the first timer in Timer A.

Miscellaneous Items

The indivisible Read-Modify-Write instruction TAS is not supported by any of the memory interfaces on the SMIDI. The address strobe signal $\overline{\text{AS}}$ is used as the bus cycle terminator and as this is not negated in between the read and write cycles of the RMW instruction, the 68K would hang.

A no-wait state memory access is executed by the 68K in 8 states, or 400nS at 10MHz.

Processor, ROMs, and Decoding (ref. CMI-28-1)

The 10MHz processor clock (PCLK) is generated by the 20MHz crystal oscillator and made symmetrical by the flip-flop D10*b*. Detailed description of the operation of the 68K can be obtained from the Motorola literature but the following is a very brief indication of how 68K buss cycles proceed. A cycle is initiated by driving the address onto the 23 address lines A1 to A23 following the falling edge of the clock and asserting the address strobe $\overline{AS}$ 50nS later, by which time the address lines should be stable. There is no A0 address line signal. A1 to A23 determine which 16-bit word in memory is to be accessed. Whether the high byte or the low byte or both is to be accessed is determined by the *Upper and Lower Data Strobes*, $\overline{UDS}$ and $\overline{LDS}$, so the least significant address bit A0 is implied by these two data strobes. With a 24 bit effective address width, the 68K can access 16 megabytes of memory.

In a read cycle, $\overline{UDS}$ and/or $\overline{LDS}$ are asserted at the same time as the address strobe. For a write cycle, the processor puts its data on the buss 50nS after $\overline{AS}$ and asserts the data strobe(s) 50nS later still, by which time the data should be stable.

Nothing happens now until the accessed device returns a *Data Transfer Acknowledge* ($\overline{DTACK}$) signal. The processor samples $\overline{DTACK}$ and recognises that it has been asserted on the falling edge of the clock. One clock cycle later, on the next falling edge, the buss cycle is terminated. In the read case, the data is latched into the processor, and address and data strobes negated. In the write case, the strobes are negated then the data buss tri-stated 50nS later. Each half clock cycle which the processor spends waiting for the return of $\overline{DTACK}$ is called a **wait state.**

The upper seven address lines A17-A23 are used in the decoding circuitry E2 (LS32) and E12 (LS259), which divides the address space as shown on the memory map. All decoder outputs depend upon $\overline{AS}$ being asserted so the entire address is stable whenever any output is active. The first 128K bytes is to select ROM's, the second 128K bytes is unused. The next 128K bytes is occupied by the DMA to CMI memory via P1 or P2. The ACIA's and Timers are addressed next, then the private RAM.

The ROMs must reside at the bottom of the memory space because after reset the 68K boots up by fetching its supervisor stack vector and restart program counter from the locations 0 and 4 respectively. 450nS ROMs are currently used requiring the insertion of 6 wait states (each wait state is 50nS long) into each ROM access cycle. The counting of the wait states is performed by the LS161 counter at F1. While $\overline{EPROM}$ is not asserted, the counter is held preset at 13 but when $\overline{EPROM}$ comes active, the counter is released while the ROM(s) are enabled. On the second rising edge of the clock after this, the counter reaches 15 and generates the ripple carry out which clocks the flip-flop at G1*b* to assert $\overline{DTACK1}$. When the processor terminates the cycle, $\overline{AS}$ is negated so BAS goes low and clears the flip flop to remove $\overline{DTACK}$. $\overline{EPROM}$ is negated so the LS161 is put back into preset ready for the next ROM cycle.

The delay of $\overline{\text{EPROM}}$ through the decoding circuitry plus the two counted clock cycles, plus the delay between $\overline{\text{DTACK}}$ and the processor latching the data in provides the required 450nS access time. The complete cycle takes 700nS from bus inactive to bus inactive again.

Memory - ROM and RAM

(refer schematic CMI-28-2)

There are four 28-pin sockets for ROM and static RAM. The minimum configuration is 8k words of ROM (2x2764)and 8k words of static RAM (2x6264). The ROM can be configured for 16k words by breaking the link (LK1) between pin 27 of the ROM's and +5v and join the link LK1 to A15 from the processor and plugging in the appropriate two 27128's. Similarly the RAM can be arranged to accommodate 32k static RAM chips (e.g.MK4856 pseudo-statics) by breaking the links LK2 and LK3 to +5v and connecting A14 and A15 to pins 26 and 1 of the RAM chips (via LK3 and LK2), respectively. Further, there is an option for 64k words of RAM; by soldering two 32k RAM chips on top of each other except for pin-20, the chip select, which should be connected to the pads provided from the select circuitry, the OR gates pins 3 and 11 of D12 (LS32). All these memory expansions will depend on the availability of the chips mentioned.

Note; when plugging in the ROM's, they should be labelled *odd* and *even*; the even one should be plugged into E5,6 (near the 68K) and the odd one into E8 (between the RAM chips).

Memory addressing: ROM starts at 000000h and RAM starts at 080000h.

Static RAM is fast enough (150ns) to not need a delay on the $\overline{\text{DTACK}}$ line. So that when RAM is selected $\overline{\text{DTACK}}$ is also enabled.

Control Latch and Interrupts

(refer schematic CMI-28-3)

The 68K can be reset, halted and interrupted on two levels via a write only latch at FCA0h in the CPU address space. This address is decoded by the large NAND gate B3. The decoder output is latched by flip-flop B2*b* on the rising of BRA and data is written into the LS259 latch B4 at the end of the system CAS pulse. The 74LS14 (A5) permanently buffers all incoming data to the latch.

The first 3 data bits from the CPU (D0-D2) are used as an address for the 74LS259 and the fourth data line D3 provides the one-bit data for this latch. With this arrangement if several processors attempt to write to the SMIDI latch there will be no clash.

The bit assignments are as follows:

Latch Address;			Data;	Function;
D2	D1	D0	D3	- Active for:
0	0	0	0	68K Interrupt Level 1
0	0	1	-	n/c
0	1	0	1	CMI Interrupt 2
0	1	1	0	68K Interrupt Level 7
1	0	0	1	CMI Interrupt 1
1	0	1	0	Halt 68K
1	1	0	0	Timer Sync switch
1	1	1	0	Reset 68K

All the interrupts to the 68K from the control latch, the ACIA's, Timers and SMPTE circuitry are fed to the priority encoder C2 (LS148) its 3-bit output is fed directly to the 68K processor. Level 7 (all bits low) is the highest priority (and non-maskable) while level 0 (all bits high) means no interrupt. Interrupts are cleared by the 68K writing either to the control latch itself via the CPU bus (to clear interrupts 1 and 7), or reading ACIA or Timer status or to address 60060h for SMPTE read, or to 60070h for SMPTE write.

To reset the processor, both $\overline{\text{HALT}}$ and $\overline{\text{RESET}}$ must be asserted. The minimum reset period required by the 68K is 10 clock cycles i.e. 1uS.

The state of the $\overline{\text{HALT}}$ and $\overline{\text{RESET}}$ signals are indicated by the LEDS near the front of the SMIDI card. Both off means that the processor is running.

System Reset ($\overline{\text{BSYRES}}$) clears the latch, thus putting the 68K into reset. The level 7 interrupt also generated will be ignored while the processor is reset. A manual control panel may be plugged into 26-way socket on the SMIDI board with a $\overline{\text{HALT}}$/RUN toggle switch and RESET momentary switch.

The 68K has a very powerful interrupt vectoring system which permits the interrupt service routine vector to be provided by the interrupting device. Many such devices may therefore generate interrupts without the need for extensive polling procedures to find out which device is requesting. This facility is not required on the SMIDI and the only interrupts used are the seven Auto-vector interrupts determined by the interrupt level on $\overline{\text{IPL0}}$-$\overline{\text{IPL2}}$. The function code outputs FC0-FC2 are all high during an interrupt vector fetch and this is detected by the AND gate 74LS11 (D2), on drawing CMI-28-1. This results in $\overline{\text{VPA}}$ being asserted which tells the processor to use the autovector instead of an interrupting device-supplied one.

68k/6809 DMA Buss Interface

(refer schematic CMI-28-4)

Communications between the 68k processor and the 6809 CPU is achieved by DMA *Direct Memory Access* on the system buss. The 68k waits until no higher priority device is occupying the buss and then either 6809 (P1 or P2) is temporily hung while the 68k executes a normal buss cycle writing to or reading from memory or a peripheral on the buss. In this manner the entire 64k address space of each 6809 processor appears as a small slice of the 16 megabyte address space of the 68k. Software then defines various protocols for the different processors to pass messages and data to one another by simply placing them in system memory.

The DMA interface provided on the 68K SMPTE/MIDI Card is a very flexible one. It automatically handles either 8 or 16 bit data transfers (doing double cycles across the 8-bit CMI buss in the latter case) and can do so on either P1 or P2 cycles, selecting any desired memory mapping which has been set up on the Q256 memory card.

DMA is initiated by the 68K when it accesses any address in the range 040000h to 05FFFFh. These addresses are decoded by the LS259 (E12) on drawing CMI-28-1 and result in the $\overline{\text{CMI}}$ signal being asserted (low). Since the rest of the interface circuitry is not activated yet, PACK (to be explained later) will be low and a low will be presented at the data input of flip flop C12*a* whose function is to synchronise the transfer with the CMI buss.

Address line A16 is used to select which 6809 processor's buss cycle(s) are to be used for the transfer. The timing signals for both processors are input to LS241 buffer A7 which is wired as a multiplexer:-

If A16 is low, P2ø2 is enabled through to become Pø2, ADD2 becomes PADD and so on.

If A16 is high, P1's timing signals are enabled instead.

By this means, the address range specified above is split in two: from 040000h to 04FFFFh the transfer automatically occurs on P2 buss cycles, while from 050000h to 05FFFFh it occurs on P1 cycles. Refer to the 6809 CPU doc cycles.

Thus at the beginning of the data cycle of whichever processor is selected, the Pø2 signal clocks the LS74, recording the fact that a DMA cycle is required.

All DMA devices are interconnected on the motherboard in a *daisy chain*. Each device is assigned a given priority in the chain and must wait until no higher priority device is already using the buss. The 6809 CPU is the always the last device in the chain. There are two separate daisy chains in the CMI system, one for each 6809 CPU. Since the 68K SMIDI card can perform DMA on either CPU's cycles, it is a member of both chains. $\overline{\text{ETL1}}$, $\overline{\text{ENL1}}$ and $\overline{\text{RDMA1}}$ are the chain signals for P1, $\overline{\text{ETL2}}$, $\overline{\text{ENL2}}$, $\overline{\text{RDMA2}}$ are for P2. Which set is used is again selected by the state of A16 at the time of transfer.

The selected $\overline{\text{ETL}}$ *Enable This Level* signal is low when no higher priority device is occupying the buss. After the $\overline{\text{CMI}}$ signal has been latched, nothing happens until this signal is low, whereupon the $\overline{\text{RDMA}}$ *Request DMA* is driven low through the selected LS10 gate. Any DMA device pulls this open collector line low to to request buss access to the CPU. At the same time, the selected $\overline{\text{ENL}}$ (Enable Next Level) signal is inhibited. Normally, the low on $\overline{\text{ETL}}$ comes in and goes out again on $\overline{\text{ENL}}$ to indicate to lower priority devices that the buss is available but when the 68K requires a transfer $\overline{\text{ENL}}$ is held high to hold up the lower devices.

The CPU acknowledges that it will hang and release the buss for the next cycle by asserting ACK1 or ACK2; the selected ACK signal becomes PACK. When a request has been generated (C12*a* $\overline{Q}$ hi) and this level is enabled (ENL low), the rising edge of PACK clocks a low into flip flop B11*a* to generate $\overline{DCYCLE}$. This signal indicates that the next buss cycle is definitely going to be a 68K DMA transfer and remains asserted until the end of the address phase of the actual DMA cycle.

The other half *b* of B11 is also clocked by PACK to generate the P1 or P2 $\overline{DMAC}$ *DMA Claim* signal as selected by A16. This signal goes to the Q256 RAM card to select the memory mapping which has been set up specifically for the 68K. In this way the 68K may have access to part or all of the same physical memory space as the 6809 CPU or it may have access to an entirely different part of physical memory as required by software. The $\overline{DMAC}$ signal is asserted during the data cycle preceding the actual transfer.

The address phase of the DMA cycle is indicated when $\overline{ATB}$ *Address To Bus* is asserted by the LS10 B10. At this time the lower 15 bits of the 68K address buss are enabled on to the CMI buss through the two LS244's A2 and A3 to select the required location within the 6809 address space. VMA is driven high through LS125 B1 to indicate a *Valid Memory Address* and the 68K R/W line is driven through the same buffer to indicate a read or write cycle.When the 68K performs 8-bit memory accesses, the $\overline{UDS}$ and $\overline{LDS}$ signals (*upper* and *lower data strobes*) indicate whether an even or odd address is being accessed. The sense of these signals are clocked into JK flip flop H12*a* at the beginning of $\overline{DCYCLE}$ to generate HIBYTE and LOBYTE. The latter signal becomes the least significant address line driven onto MA0 through A3. In the case of 16-bit accesses, the hardware automatically requests two successive DMA accesses across the 8-bit CMI buss. Both $\overline{UDS}$ and $\overline{LDS}$ are asserted so that the JK outputs HIBYTE and LOBYTE simply toggle on each access. It does not matter which byte transfers first and in fact this depends on the initial state of H12*a*. LOBYTE directs the data to or from the odd or even address and both signals control whether the higher or lower 8 data lines are directed to the data buss.

The data buss interface consists of Schmitt bidirectional buss transceiver LS640 A6 and bidirectional driver/latches E,D5 and E,D6 (LS646s). The data phase of the DMA transfer is indicated by the assertion of $\overline{DTB}$ *Data To Bus* at the rising edge of BRA when a DMA cycle is in progress. This is performed by flip flop C12*b*. $\overline{DTB}$ enables the buss transceiver A6 and the direction is determined by the 68K R/W signal.

If the 68K is writing to the CMI buss, E,D5 or E,D6 simply act as buffers to transfer the high or low 68K data signals (PD0-15) through to A6. HIBYTE or LOBYTE plus $\overline{CMI}$ being asserted will drive the $\overline{G}$ input of the appropriate LS646 for the duration of the DMA cycle (LS02 and LS32 gates B8 and A8).

When the 68K reads from the CMI buss, E,D5 or E,D6 must latch the data in from the buss to hold it until the 68K terminates its own cycle and latches the data internally, about 50nS after the end of the DMA cycle. 100nS before the end of the data phase, the CMI timing signal CAS goes low, resulting in a rising edge on $\overline{BCAS}$. Data from memory is guaranteed to be valid at this time. B10 generates the LDATA *Latch Data* signal which is ANDed with either HIBYTE or LOBYTE to latch the data coming into the A side of E,D5 or E,D6. The output of the latch (B side of the selected LS646) is driven onto the PD lines until the 68K completes its cycle and negates $\overline{CMI}$.

Termination of the transfer after single or double DMA cycles is controlled by the two flip flops in LS74 C10:

(i) In the single (8-bit) transfer case, either UDS or LDS will be low. This will cause the LS10 A10 to output a high, and $\overline{DTACK2}$ will be generated as soon as LDATA occurs. The 68K will then terminate its cycle immediately, after only one DMA cycle.

(ii) In the double DMA cycle (16-bit) case, both $\overline{UDS}$ and $\overline{LDS}$ are high so $\overline{DTACK2}$ will not be generated until the first flip flop in C10 is set. Initially this flip flop is reset. At the first LDATA pulse a high is clocked in but $\overline{DTACK2}$ is not generated because of the propagation delay through to the next flip flop. Since $\overline{DTACK2}$ is not asserted, the 68K still waits with address and address/data strobes asserted.

If writing, the data remains asserted by the 68K but both address and data are removed from the CMI buss when $\overline{ATB}$ and $\overline{DTB}$ are negated respectively. If reading, the first byte read in is latched and held by E,D5 or E,D6. Since $\overline{CMI}$ will still be asserted and PACK will have been negated, the whole process of waiting for daisy chain priority and DMA requesting begins again in order to perform a second DMA cycle. The second cycle can be held up indefinitely by higher priority devices using the buss after the first cycle. When the second LDATA edge comes along the high on the LS10 output is clocked into the second C10*a* flip flop and $\overline{DTACK2}$ is asserted.

On the next falling edge of PCLK, the 68K recognises that $\overline{DTACK}$ has been asserted. On the second falling edge of PCLK the data is latched internally for a read, and the address and strobes are released. The low on BAS resets the flip flops at C10.

Debugging Notes for the DMA Circuitry

If the timing circuitry of the DMA interface is faulty, the most likely result is that $\overline{DTACK2}$ will never be generated and the 68K will simply hang which makes debugging easy. In this case, check first that the address decoding is generating $\overline{CMI}$, then that the daisy chain signals are present. Then look for an 800nS pulse on $\overline{DCYCLE}$ (pin 5 B11*a*), indicating that DMA cycles are actually ocurring. Continue through to the $\overline{ATB}$, $\overline{DTB}$ and $\overline{LDATA}$ signals, checking not only that they are generated but also that they get to their respective destinations in the circuitry.

If the DMA cycles are being synchronised and timed correctly check that the address buffers and data buffer/latches are being enabled and clocked at the correct times.

If all timing circuitry is correct, the last possibilty is data or address buss shorts, open circuits or faulty drivers. Special test ROMs are available which cause the 68K to repetitively copy bytes and words from one location to another in CMI memory. The 6809 monitor can then be used to deduce which data or addresses cause problems.

SMIDI Card Peripheral Select

(refer schematic CMI-28-5)

The ACIA's and Timers work from an 8-bit data buss with (asynchronous) interfacing circuitry.

Initially the flip-flops (F2) are cleared causing a high $\overline{\text{DTACK3}}$ output setting the LS646 transceiver (G4) into the transparent mode. The direction of data flow is determined by the R/W line with the IO selected. Without IO line selected it appears in write mode. The peripheral is selected by the LS138 enabled by the $\overline{\text{CS}}$ signal. The first flip-flop F2*a* is clocked on the first falling edge of E with the IO select and the data strobe high (ie either $\overline{\text{LDS}}$ or $\overline{\text{UDS}}$ low). The $\overline{\text{Q}}$ output of F2*a* is applied to the NAND gate (H1), asserting $\overline{\text{CS}}$. Selecting the peripheral at this time ensures that the peripheral has adequate address setup time.

On the next falling edge of E, the $\overline{\text{Q}}$ output of F2*b* is clocked low asserting $\overline{\text{DTACK3}}$ and latching data in the transceiver (G4) The asserted $\overline{\text{DTACK3}}$ signal deselects the peripheral by causing $\overline{\text{CS}}$ to go high. Flip-flop F2*a* is cleared by IO going low when the access terminates. Clearing flip-flop F2*a* also initializes the interface circuitry for the next access.

The ACIA's are selected by E10 (LS138) and appear at addresses; 60020h, 60030h, 60040h and 60050h. They share a common interrupt level (level 3). Their transmit and receive data lines are wired to the 26-way connector to be connected to the MIDI drivers and opto-coupler receivers.

The programmable timers appear at the general addresses; 60000h, and 60010h has an interrupt to the 68K (level 2, $\overline{\text{INT2}}$).

SMIDI ACIA's and Timers

(refer schematic CMI-28-6)

There are four different peripheral circuits on the SMIDI card. Firstly there is the four ACIA's (68B50) (G7-11) which are the MIDI ports A,B,C, and D, respectively. Then there is the Timer (68B40) (G5,6), *Timer A* which is used in conjuction with the SMPTE read and generate circuits (which are the other two peripheral circuits) as well as the Click In and Out. There is another Timer (68B40) (H,G3), *Timer B* which does the Multi-sync outputs.

The ACIA's and Timers are driven also by the E (enable) signal from the 68K. The frequency of this clock is one-tenth of the 68K clock (10MHz) with a 60/40 duty cycle (6 clocks high, 4 clocks low)

The Sync switching circuitry consisting of the EXOR gate acting as an inverter (C1) and the two switched gates (LS125) (B1) provide a facility to *divide by one*, which the Timers do not have. When the SYNCSW line is low (from the Control Latch (B4)) the signal coming

in at Click In will be connected to Sync Out 4 directly as well as clocking the third timer in Timer A (G5,6) and clocking all timers in Timer B. When SYNCSW is high the clock signal coming in at Click In is first divided by timer 3 in Timer A before clocking the three timers in Timer B.

The ACIA's can interrupt the 68K on interrupt level 3, ($\overline{\text{INT3}}$).

The Timers can interrupt the 68K on interrupt level 2, ($\overline{\text{INT2}}$).

SMPTE Generating Circuitry

(refer schematic CMI-28-7)

An oscillator (3.84MHz) is divided by 10 (G2,H1,G1*a*) to provide a standard for generating the 3 different rates of SMPTE code (24,25 and 30 fps) All three are denominators of 384,000. Further division, depending on the frame rate selected, is done by the second timer in Timer A(G5,6) giving the signal CLK2, which is the bit rate for a SMPTE *one* (ie 160 bits per frame). This is in turn divided by 2 (C11*b*) giving CLK1 which is the bit rate for a SMPTE *zero* (ie 80 bits per frame). When a SMPTE word is ready it is written to the Parallel-In-Serial-Out registers LS165 (C7,C8) at address 60070h (through B9,B8 and B7). When this writing takes place the interrupt $\overline{\text{INT4}}$ if it has been asserted is now cleared. The data in the shift registers (C7,C8) is clocked out by CLK1, a 4-bit counter LS161 (D11) is also clocked which causes the interrupt on level-4 (INT4) when it reaches its terminal count of 16. Now, if a *zero* is shifted out from C8 the flip-flop C11*a* is toggled at the rate determined by CLK2, but if a *one* is shifted out from C8 the flip-flop C11*a* is toggled at the CLK1 rate. Thus, the word stored on the shift registers is output in SMPTE form.

$\overline{\text{INT4}}$ is cleared by writing to the shift registers (C7,C8), i.e. by signal $\overline{\text{SMPTERD}}$ at address 60060h.

SMPTE Reading Circuitry

(Refer schematic CMI-28-7 & Timing Diagram)

SMPTE code coming from tape, being converted to TTL signal levels by the CMI-333 board, are received by the CMI-28 through pin 17 of the 26-way connector. The edge-detecting circuitry consisting of the EXOR gates (C1) and the resistor-capacitor combination create a pulse (at pin 6 of C1) for every up or down transition of the incoming signal.

The required output from this SMPTE data separator is to have one interrupt occur for every SMPTE *one* read and another interrupt for every SMPTE *zero* read. This process can be followed through with the timing diagrams. The 68B40 timer is set, according to the frame rate of the SMPTE being read, to 3/4 of the time for one bit cell. The circuit then detects whether there has been a transition in that time or not. If there has been a transistion then an interrupt on level 6 occurs and a *one* is read, if no transition occurred then an interrupt on level 5 occurs and a *zero* is read.

$\overline{\text{INT6}}$ and $\overline{\text{INT5}}$ are cleared by the $\overline{\text{SMPTERD}}$ signal through accessing address 60070h. This will reset flip-flops in D1 making the outputs of the NAND's E1 high.

Pin Connections for the 26-way Connector

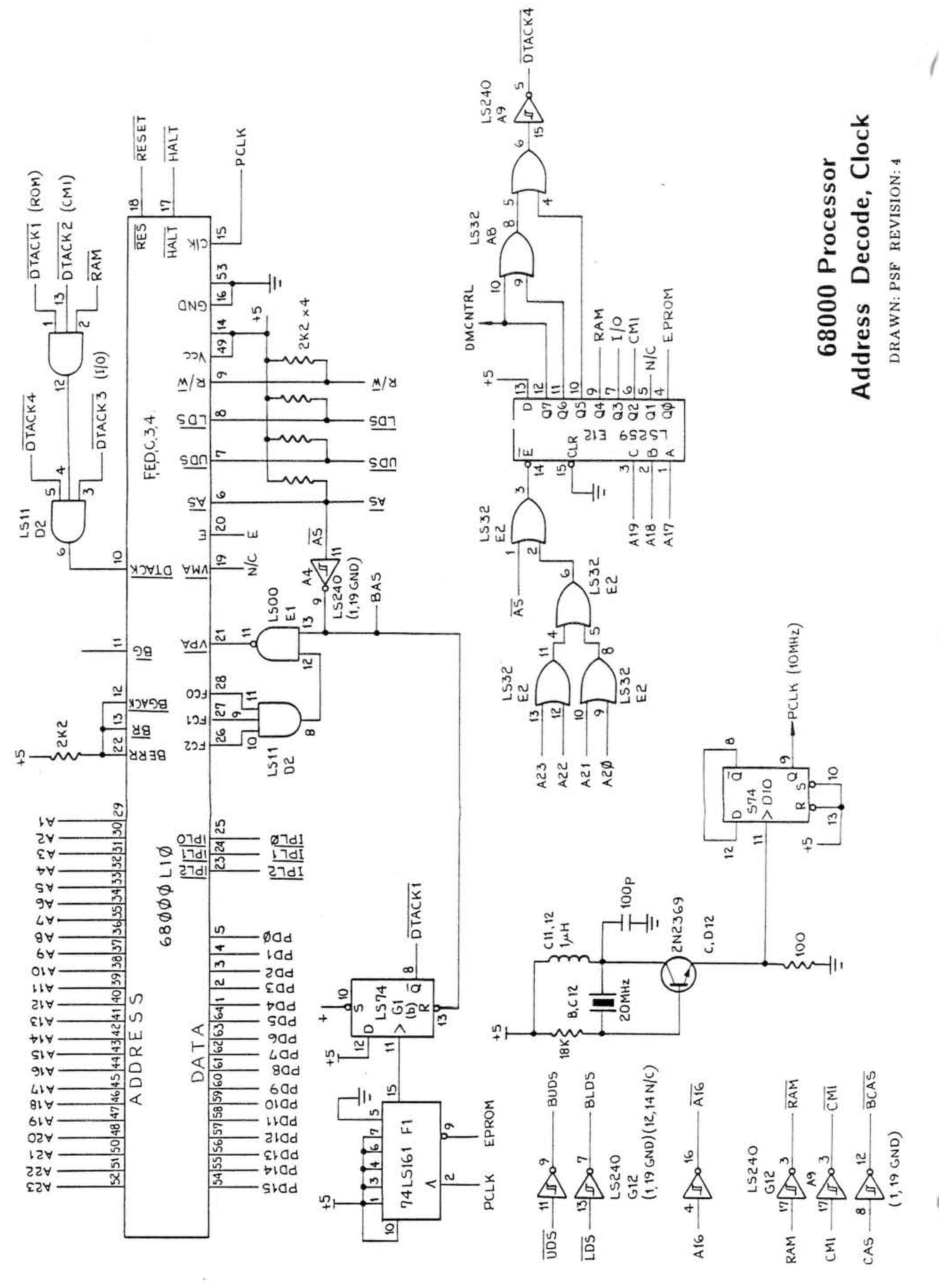
(between the CMI-28 and CMI-332 and CMI-333)

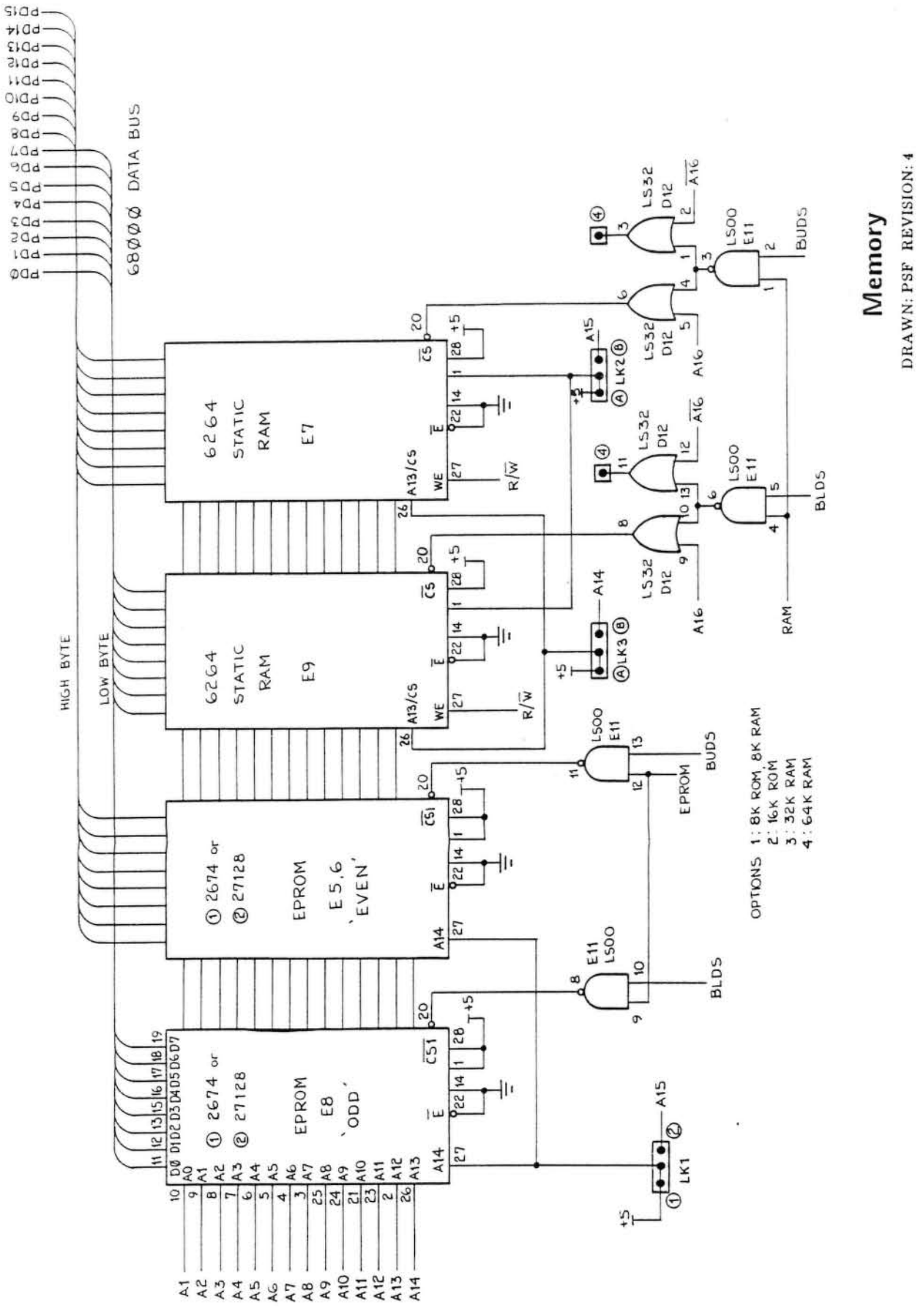
Pin 1	MIDI out A.	
Pin 2	+5 volts.	
Pin 3	MIDI in A.	
Pin 4	SYNC out 1.	
Pin 5	MIDI out B.	
Pin 6	SYNC out 2.	
Pin 7	MIDI in B.	
Pin 8	SYNC out 3.	
Pin 9	MIDI out C.	
Pin 10	Digital Ground.	
Pin 11	MIDI in C.	
Pin 12	Digital Ground.	
Pin 13	MIDI out D.	
Pin 14	RESET/START.	
Pin 15	MIDI in D.	
Pin 16	RUN/STOP.	
Pin 17	SMPTE code in.	
Pin 18	Digital Ground.	
Pin 19	SMPTE code out.	
Pin 20	CLICK out; SYNC out 4.	
Pin 21	CLICK in.	
Pin 22	(CMI332-3) Analog Ground.#	(CMI28) n/c.*
Pin 23	(CMI332-3) +15 volts.#	(CMI28) CPU Halt switch.*
Pin 24	(CMI332-3) -15 volts.#	(CMI28) Digital Ground.*
Pin 25	(CMI332-3) n/c.#	(CMI28) CPU Reset switch.*
Pin 26	(CMI332-3) n/c.#	(CMI28) Digital Ground.*

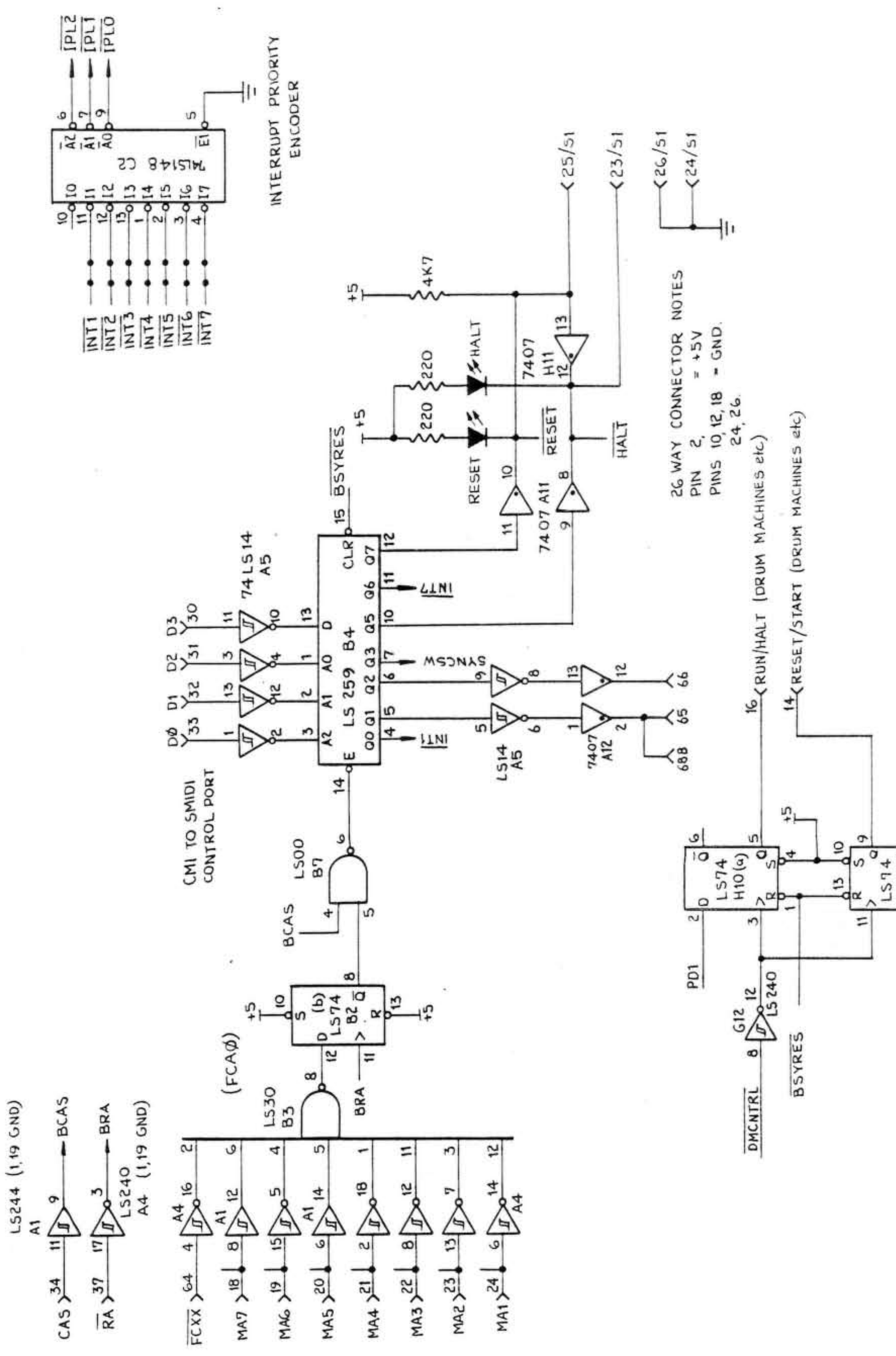
Notes:

- these connections are on Audio Rack only.

* - these connections (from the CMI28 board only) are for debugging purposes only. If two push-button switches are connected between pins 23 & 24 and pins 25 & 26, they can be used to manually halt and reset the 68K processor, respectively.

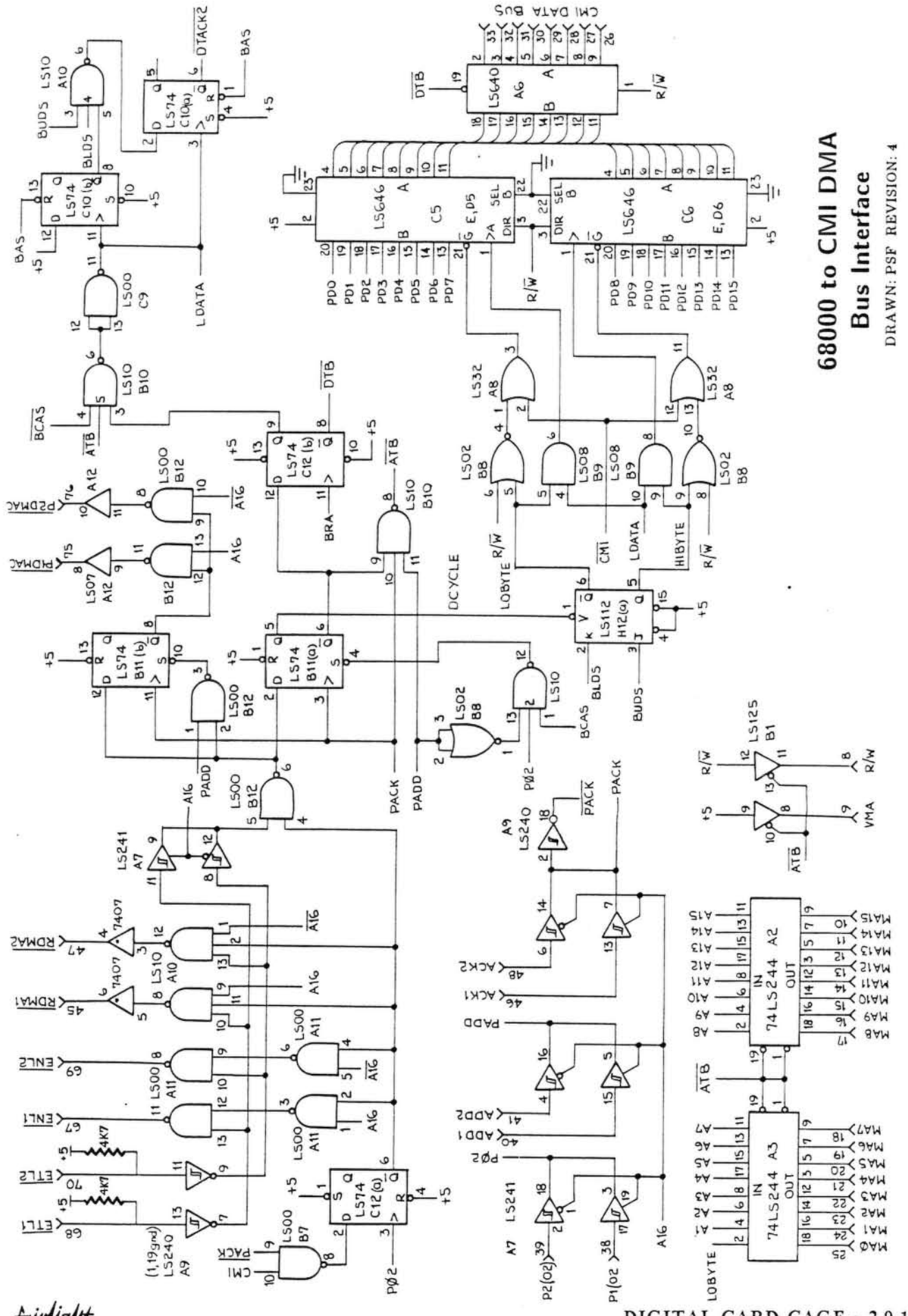

68000 Processor
Address Decode, Clock
DRAWN: PSF REVISION: 4
68000 L10
ADDRESS
DATA
74LS161 F1
LS74
LS259 E12
LS32
LS240
LS11
LS00
2N2369
20MHz
PCLK (10MHz)
DTACK1 (ROM)
DTACK2 (CMI)
DTACK3 (I/O)
DTACK4
RESET
HALT
PCLK
EPROM
DMCNTRL

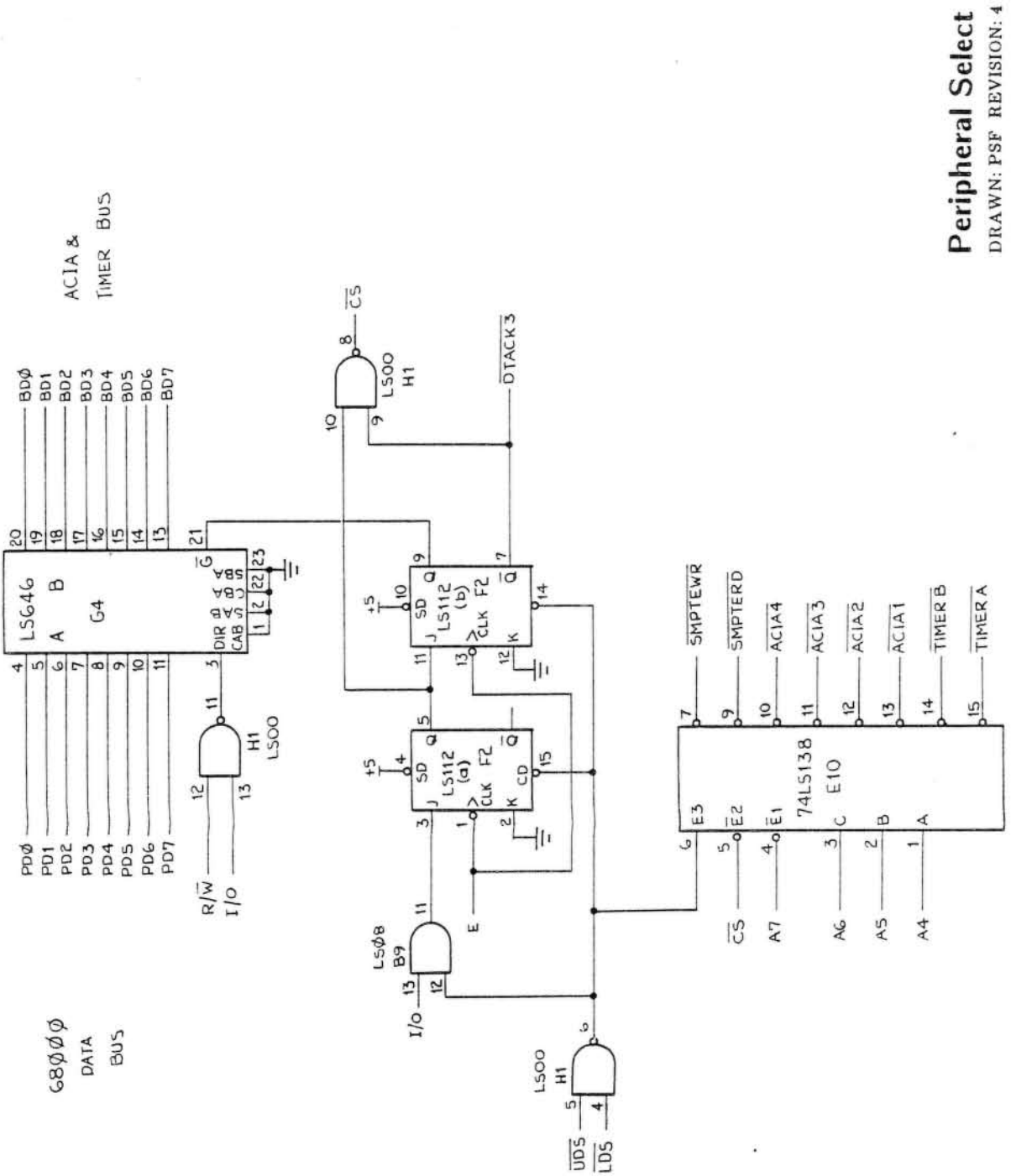

Memory
DRAWN: PSF REVISION: 4
68ØØØ DATA BUS
68ØØØ ADDRESS BUS
HIGH BYTE
LOW BYTE
6264 STATIC RAM E7
6264 STATIC RAM E9
EPROM E5,6 'EVEN'
EPROM E8 'ODD'
① 2674 or ② 27128
LS32 D12
LS00 E11
LK1
LK2
LK3
BUDS
BLDS
EPROM
RAM
R/W
OPTIONS 1: 8K ROM, 8K RAM
2: 16K ROM
3: 32K RAM
4: 64K RAM

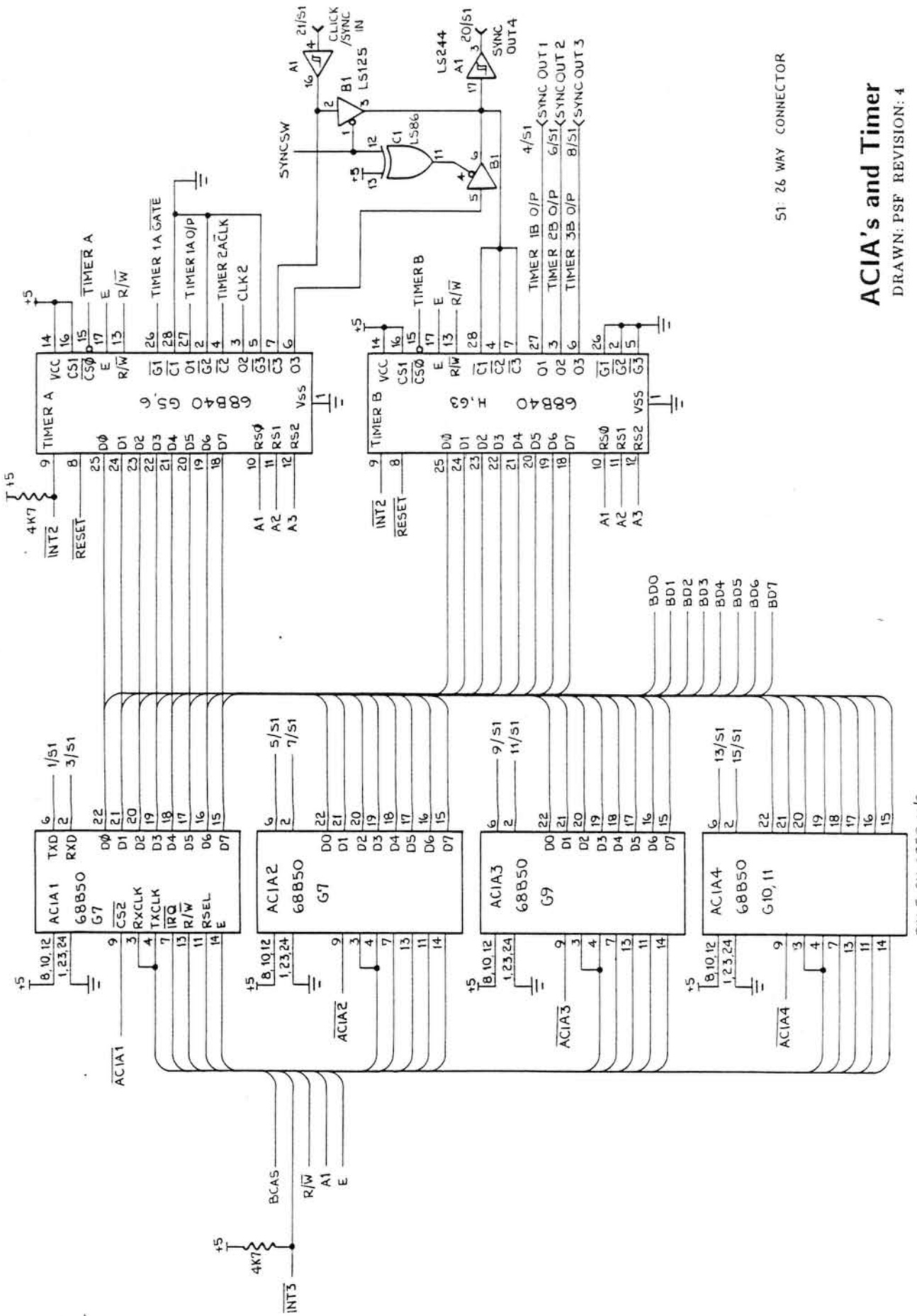


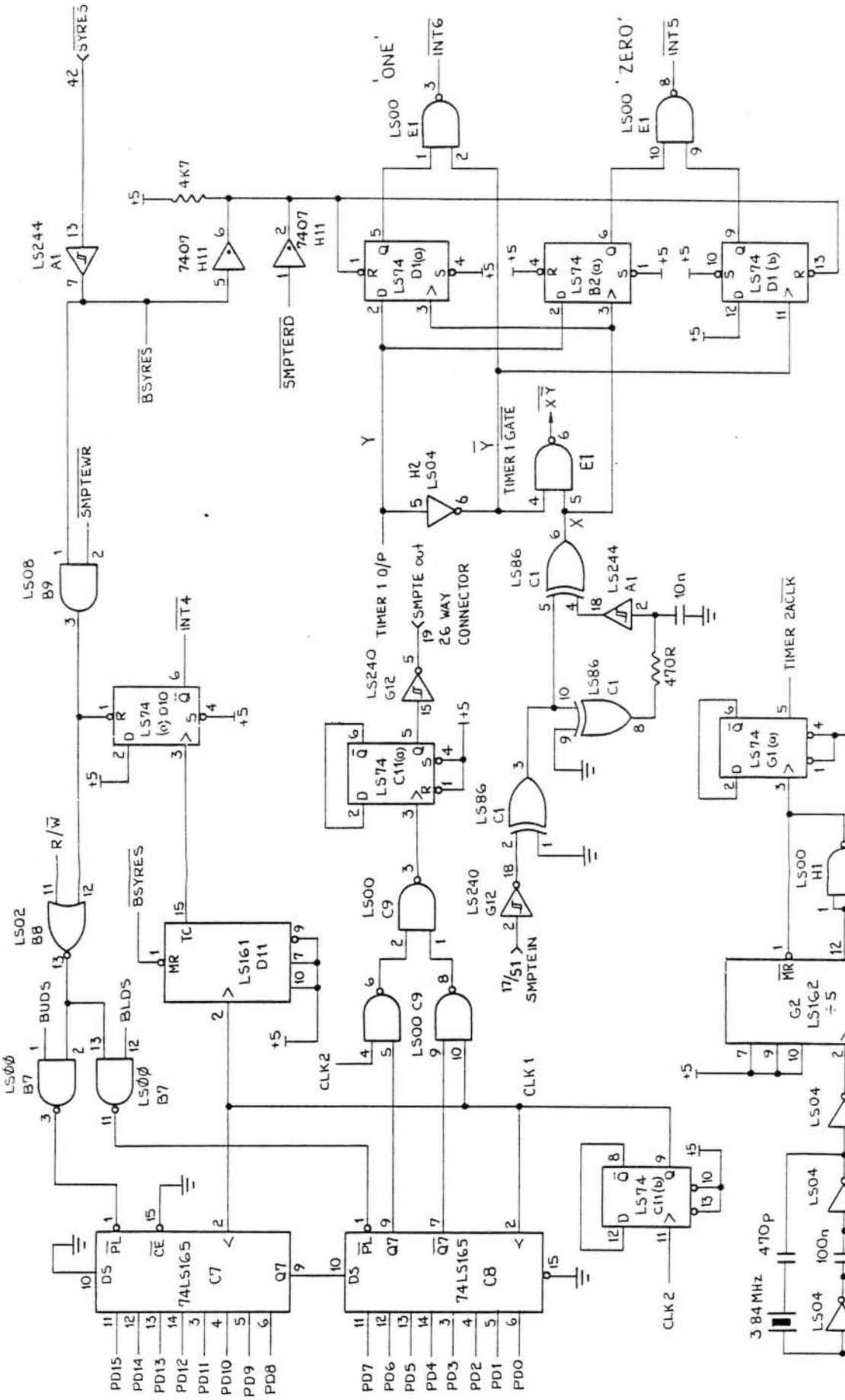
Halt/reset control, Interrupt priority

DRAWN: PSF REVISION: 4


68000 to CMI DMA
Bus Interface
DRAWN: PSF REVISION: 4


Peripheral Select
DRAWN: PSF REVISION: 4
ACIA & TIMER BUS
68000 DATA BUS
LS646
G4
LS112 (a) F2
LS112 (b) F2
LS08 B9
LS00 H1
74LS138 E10
DTACK3
SMPTEWR
SMPTERD
ACIA4
ACIA3
ACIA2
ACIA1
TIMER B
TIMER A
R/W
I/O
E
UDS
LDS


ACIA's and Timer
DRAWN: PSF REVISION: 4
51 26 WAY CONNECTOR
TIMER A
TIMER B
ACIA1
ACIA2
ACIA3
ACIA4
68850
SYNCSW
CLICK /SYNC IN
SYNC OUT4
SYNC OUT 1
SYNC OUT 2
SYNC OUT 3
TIMER 1A GATE
TIMER 1A O/P
TIMER 2ACLK
CLK2
TIMER 1B O/P
TIMER 2B O/P
TIMER 3B O/P
TIMER A
TIMER B
INT2
RESET
INT3
BCAS
R/W
A1
E
4K7
LS125
LS244
LS86
PIN 5 ON 6850 N/C


SMPTE Code Generator,
Data Separator and Reader
DRAWN: PSF REVISION: 4
'ONE'
'ZERO'
SYRES
SMPTEWR
SMPTERD
BSYRES
TIMER 1 GATE
TIMER 1 O/P
SMPTE out
26 WAY CONNECTOR
TIMER 2 CLK
SMPTE IN
CLK1
CLK2
R/W
BUDS
BLDS
74LS165
LS161
LS162
3.84MHz
PD15
PD14
PD13
PD12
PD11
PD10
PD9
PD8
PD7
PD6
PD5
PD4
PD3
PD2
PD1
PD0