

AN1217

Application Note

Interfacing to the MC88110

Introduction

The MC88110 symmetric superscalar™ microprocessor is Motorola's second generation RISC microprocessor. It has separate on-chip instruction and data caches capable of simultaneously issuing two instructions and providing load/store data every clock. The internal architecture is designed to provide sufficiently wide data paths to enable the processor to sustain these issue rates. These paths include 80-bit internal data paths to the register files and separate 64-bit data paths from both caches to the bus interface unit (BIU). The caches interface to the BIU in a Harvard architecture and the BIU continues this 64-bit data path out to the system.

The latest generation of microprocessors are exhibiting higher degrees of integration than their predecessors. The processor itself must, however, interface to the system at some level. The degree of difficulty and efficiency of this interface are of prime interest to the hardware system designer.

The MC88110 bus interface is designed to provide a path that can sustain the data and instruction bandwidth requirements of a processor running at 50 MHz. The MC88110 bus interface has many degrees of sophistication. This document presents applications information derived from designing an application development system (ADS) around the MC88110.

MC88110 Overview

The MC88110 incorporates many features to improve efficiency and bus bandwidth. Following are some of these features:

- 64-bit split-transaction external data bus with burst transfers
- Critical-word-first burst cache line fills with instruction and data streaming
- Pipelined load and store operations
- Decoupled data cache accesses
- Data cache write-through and write-back operation
- Run-time reordering of loads and stores
- Internal Harvard architecture
- 8-Kbyte, two-way set associative physically addressed instruction cache
- 8-Kbyte, two-way set associative physically addressed data cache with dual tags for coherency
- Hardware enforced data cache coherency (bus snooping) for multiprocessor applications

For a complete list of the MC88110 microprocessor's features, refer to the *MC88110 Second Generation RISC Microprocessor User's Manual*.

Symmetric superscalar is a trademark of Motorola, Inc.

This document contains information on a new product under development. Specifications and information herein are subject to change without notice.



Application Development System (ADS) Overview

The ADS is a single master design that implements an interface to the MC88110 using discrete devices (for example, PALs and transceivers) rather than ASICs. Since this is a single master design many of the multimaster capabilities of the microprocessor are not utilized. See Appendix A, "Signal Usage on the MC88110" for signal usage of the MC88110 on the ADS.

The following discussion describes the speed of devices and the setup and hold characteristics that are required to interface to the MC88110. This information can be applied to a design that uses ASICs.

On the ADS the MC88110 interfaces to two primary buses—the peripheral bus and the DRAM subsystem bus. The peripheral bus interface of the ADS has the following peripherals:

- EPROM—512 Kbyte
 - Contains debug monitor
- MC68681—DUART
 - Provides serial port connections
 - Counter
 - Interrupt controller
- Two 40-pin logic analyzer connections
 - Provides logic analyzer connections for hardware evaluation
 - Programmable processor status (PSTAT(2-0)) signal inputs
- Application development interface (ADI)
 - Provides 8-bit parallel port interface to MAC II

The DRAM subsystem can be configured to support the following DRAM sizes:

- DRAM—2/8/32 Mbytes
 - 7:3:3:3 reads at 50 MHz with 70 ns DRAM
 - 5:3:3:3 writes at 50 MHz with 70 ns DRAM
 - Configurable size and speed
 - Software/manual remap allows DRAM to be mapped into EPROM

The memory system also supports an MC88110/MC88410/MCM62110 High Performance Module (HPM) for 1/4 Mbytes of secondary cache that can plug into the same footprint as the MC88110. The DRAM and EPROM subsystems were designed to support both the short (four beat) burst of the MC88410 as well as the long (eight beat) burst capability of the MC88410 onto the system bus.

Additionally, the design uses the junction temperature test capability that is provided by the on-chip thermal resistor and allows the PSTAT signal to be configured at reset.

Specifications

The following list provides the various specifications for the MC88110:

- Clock—33 MHz to 50 MHz crystal oscillator or pulse generator connection to BNC (optional SMA) connector
- RAM size—2/8/32 Mbyte configurations based on DRAM module installed
- EPROM size—512 Kbyte of EPROM
- Terminal and host ports—RS-232 compatible
- ADI—MAC II interface
- Power requirements—+5V + -5% at 8A (maximum)

- Temperature
 - Operation: 25°C
 - Storage: -25 to +85 °C
- Relative humidity—90% (non-condensing)
- Dimensions
 - Length: 9 inches
 - Width: 9 inches
- Layers—6 signal and 4 power/ground

ADS Functional Description

One of the primary design considerations for the ADS was to design a flexible system interface that can interface to both an MC88110 and MC88410 bus with minimal modifications. From the factory, the ADS is designed to support four beat bursting into both the EPROM and DRAM. In order to switch to the “long” burst option (eight beats) of the MC88410, the three PALs that control accesses into the DRAM must use different JEDEC files. If the “short” burst mode is used, then no changes are necessary beyond adding the MC88410 to the design (i.e., by installing an HPM into the MC88110 socket instead of installing the MC88110). The minor difference in requirements between the MC88110 and MC88410 bus interface has been met simultaneously in this design; for more information, see the “Bus Interface Controller” section. See Figure 1 for an overview of the ADS system.

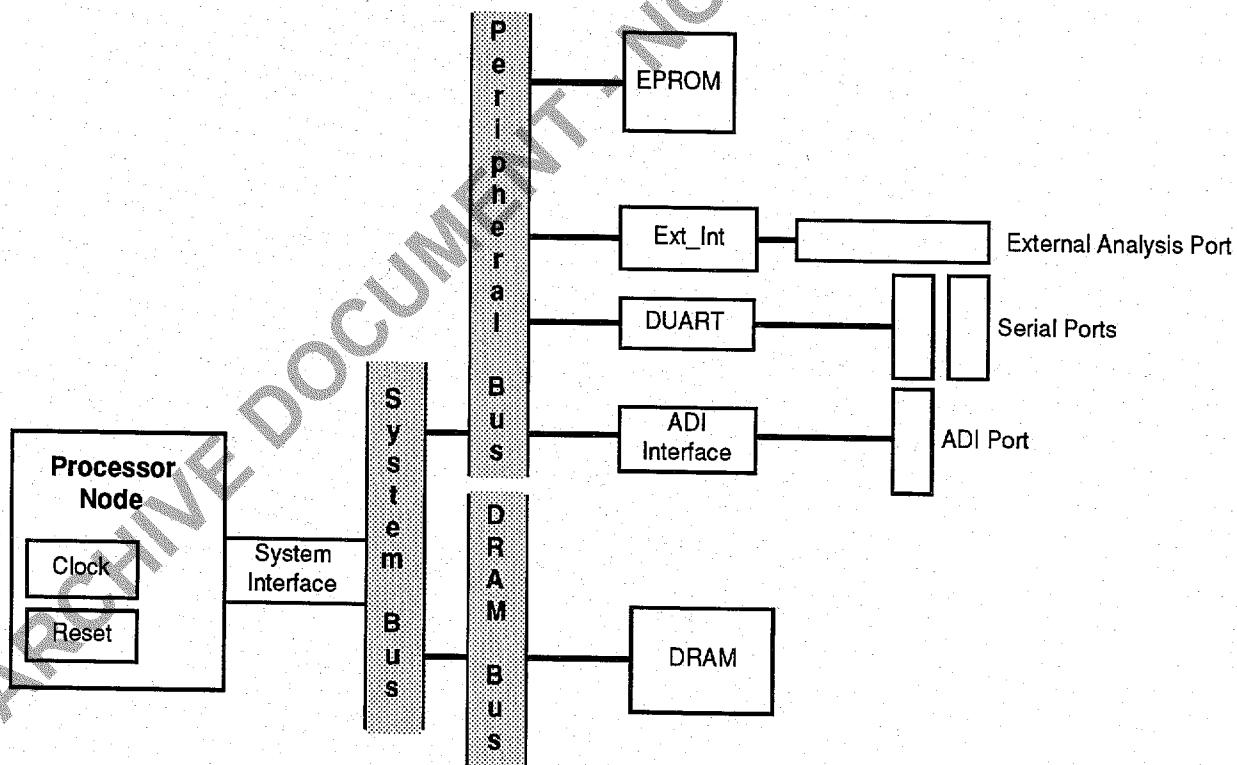


Figure 1. ADS System Overview

Physical Description

The physical location of all main components of the ADS is shown in Figure 2. The board is approximately 9 inches x 9 inches with the MC88110 socket in the center. The layout shown here is particularly well suited for keeping the routing lengths short. Routing considerations are discussed in the "ADS Hardware Design" section.

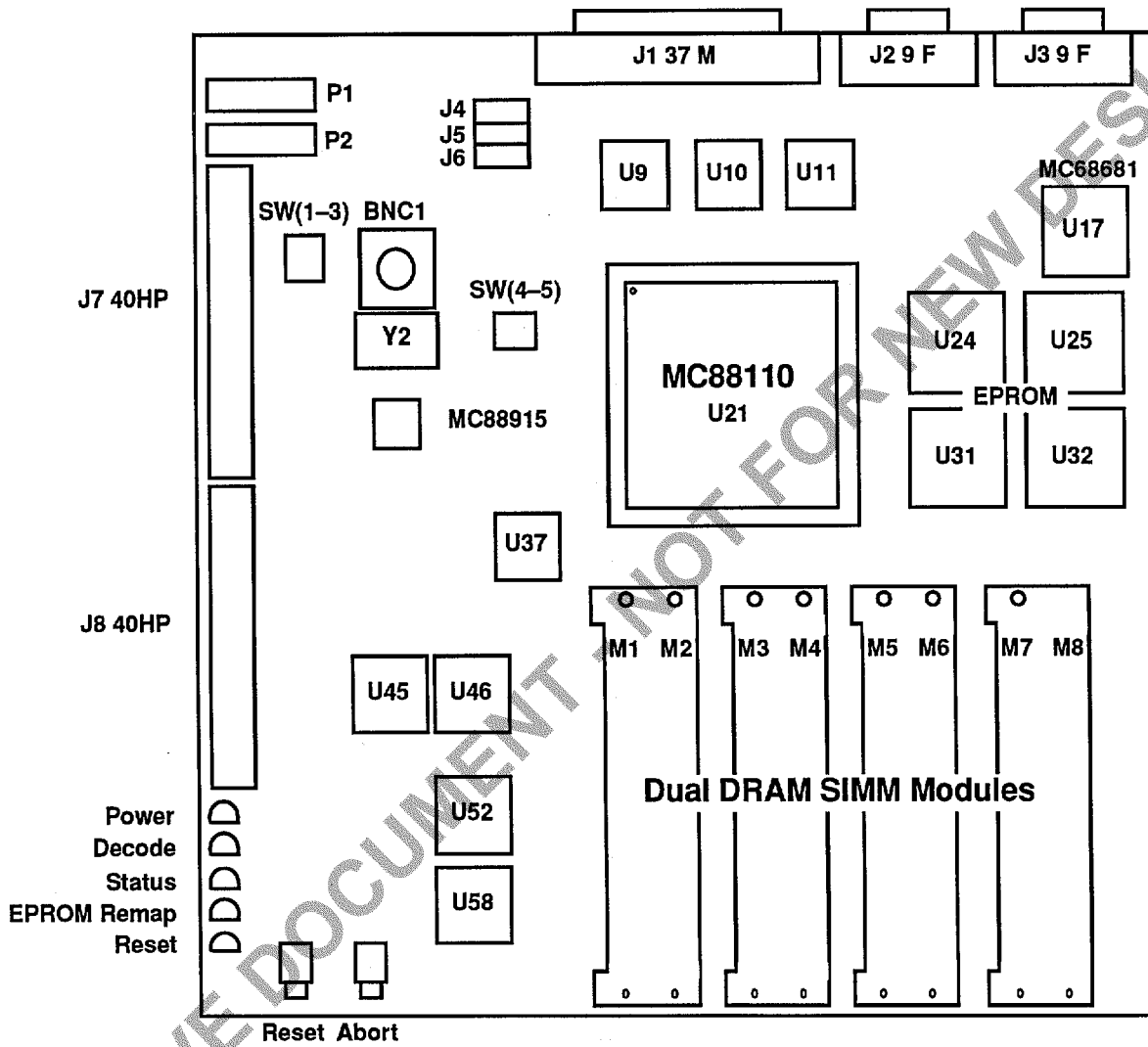


Figure 2. ADS Physical Representation

Hardware Configuration

Table 1 shows the different configurations of the ADS. A discussion of the configurations follows.

Table 1. ADS Configurations/Defaults

Configuration	Initial Setting
2/8/32-Mbyte DRAM	8 Mbyte
32/64- byte burst access	32-byte burst
External/on-board clock source	On-board clock source
PSTAT(2-0)	(000)
Junction temperature/HPM	HPM mode

2/8/32-Mbyte DRAM

The size of the DRAM is determined by the size of DRAM SIMM placed in the Dual SIMM Module Sockets at location M1-M8. See the "DRAM" section for a complete description of the DRAM.

32/64-Byte Burst Access

The EPROM interface supports burst transactions of any duration. The ADS comes equipped to support a 32-byte burst access to DRAM. In order to support the 64-byte burst transaction of the HPM, the PALs that control the DRAM accesses at locations U45, U46, and U52 need to be switched. All of the PALs on the ADS are socketed to facilitate switching them out.

External/On-Board Clock Source

The clock signals on the ADS are generated from an MC88915 clock driver. This device is particularly well suited for MC88110-based systems and has additional functionality that is used on the ADS. This functionality includes multiple Q clock outputs that are tightly coupled, two clock source inputs as well as 2xQ and Q/2 outputs.

The clock source for the ADS can come either from the BNC/SMA (BNC is generally installed) connector at location BNC1 or from the oscillator at location Y2. These locations are shown in Figure 3. The dip switch at location SW(4-5) (switch 4) selects the source that is used.

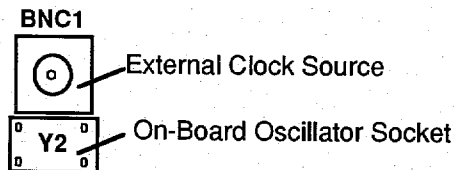


Figure 3. Clock Sources

The crystal oscillator may be replaced by disconnecting power from the ADS and removing the crystal oscillator from location Y2.

PSTAT(2-0) Signals

The MC88110 displays additional internal state information through the use of the PSTAT output signals. These signals can be configured to display different types of information based on settings that are sampled during reset. The setting that is used to configure the MC88110 for a particular session is determined by the settings of the dip switch at location SW(1-3); see Figure 4. Detailed information pertaining to the

different modes of the PSTAT signals can be found in the *MC88110 Second Generation RISC Microprocessor User's Manual*.

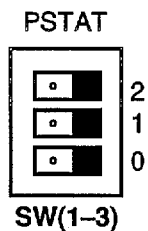


Figure 4. PSTAT Configuration Dip Switch

Note that the example shown is $PSTAT(2-0) = \text{^}b000$ during reset.

Junction Temperature and HPM Support

The ADS can be used to evaluate the MC88410 secondary cache controller by placing an HPM into the MC88110 microprocessor's socket. There are five signal timing differences between the MC88410 system bus and the MC88110 processor bus. These timing differences, shown in Table 2, must be accounted for in designing a memory system that can be used with either one.

Table 2. MC88110/MC88410 Timing Differences

Signal	MC88110	MC88410 Full-Speed	MC88410 Half-Speed
\overline{TA} (input)	\overline{TA} asserted with data	$\overline{S_TA}$ asserted one CLK before data	$\overline{S_TA}$ asserted with data
\overline{TRDY} (input)	Sampled one CLK after \overline{DBG} is asserted	Sampled with $\overline{S_DBG}$	Sampled one HCLK after $\overline{S_DBG}$ is asserted
Data (input)	9 ns setup, -3 ns hold	2.5 ns setup, 2 ns hold (to MCM62110 array)	2.5 ns setup, 2 ns hold (to MCM62110 array)
Data (output)	4 ns minimum, 15 ns maximum propagation delay	5 ns minimum, 8 ns maximum propagation delay	5 ns minimum, 8 ns maximum propagation delay
\overline{BR} (output)	\overline{BR} asserted one CLK after $\overline{SSTAT1}$ on snoop hits	$\overline{S_BR}$ asserted one CLK after $\overline{S_SSTAT1}$ on snoop hits	$\overline{S_BR}$ asserted with $\overline{S_SSTAT1}$ on snoop hits

Since this is a single master design in which the MC88110 is permanently parked on the bus, the requirements for the implementation of the two shaded rows have already been satisfied. Part of the design criteria for an HPM using an MC88410 is that if the system bus is running in full-speed mode, then the transfer acknowledge (\overline{TA}) signal indication for a normal data transfer acknowledge must arrive one clock before the data. Alternatively, if the MC88410 system bus is run in half-speed mode, then the HPM should be provided with a signal in order for it to synchronize itself, namely the half-speed clock (HCLK) signal.

The ADS supports these design requirements to enable an HPM to be used. It does this by utilizing two signals (RES1 and RES2) on the MC88110 for multiple purposes. The RES1 and RES2 signals that allow access to the internal resistor (used to determine junction temperature) can be used on an HPM's MC88110 footprint to bring up extra signals. The RES1 signal on the ADS footprint has an HCLK routed to it. The RES2 signal has an early \overline{TA} signal routed to it. The HCLK routed length from MC88915 to the RES1 signal on the MC88110 is 7.01 inches. Appendix B, "Layout and Routing ADS" includes layout summaries of the ADS MC88915 layout/routing solution.

In order for the RES1 and RES2 signals to be used for their primary function of junction temperature characterization, these signals must be isolated from the devices that are providing the alternative functions.

This is accomplished by cutting the traces that have deliberately been routed on the top layer in the location designated by the silkscreen. These locations are shown in Figure 5.

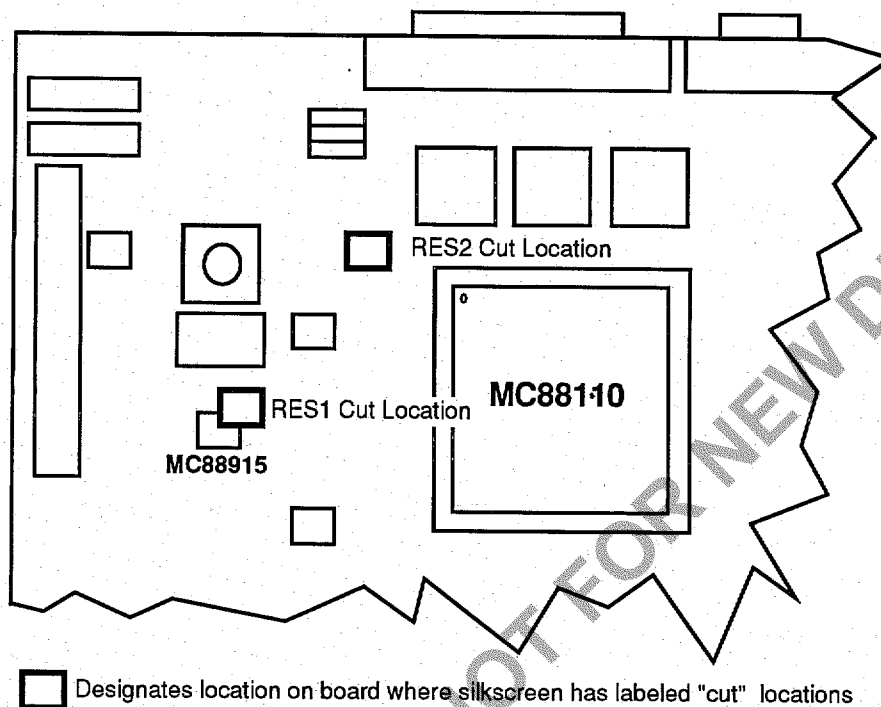


Figure 5. Dual Function Cut Locations

The differences in the data timing specifications are discussed in the "ADS Hardware Design" section. For a complete discussion of the system bus compatibility, refer to the *MC88410 Secondary Cache Controller User's Manual*.

ADS Hardware Design

The MC88110 bus is fully synchronous. This means that inputs and outputs of the MC88110 are specified with respect to the input clock. While asynchronous interrupt and reset signals are allowed, they must meet the specified setup and hold times to guarantee that they are seen on a particular clock.

MC88110 Outputs

Internally, the MC88110 generates four quadrature clocks that are phase locked to the input clock signal. All outputs from the MC88110 are specified with respect to these quadrature clocks. The resulting AC timing is shown in Table 3. For the latest AC timing specifications, refer to the *MC88110 RISC Microprocessor* technical data sheet.

Table 3. MC88110 Output AC Timing Specifications

V_{dd}=5.0 V_{dc} ± 5%, GND=0 V_{dc}

Num.	Notes 1,5	Characteristic	40 MHz		50 MHz		Unit
			Min	Max	Min	Max	
19	4	CLK to address valid	5	17	4	15	ns
20	4	CLK to address invalid	5	–	4	–	ns
21	2	CLK to \overline{TS} , \overline{ABB} , \overline{DBB} asserted, negated	0	11	0	10	ns
22		CLK to data, byte parity, BPE out valid	5	17	4	15	ns
23		CLK to data, byte parity, BPE out invalid	5	–	4	–	ns
24		CLK to data, byte parity, BPE out hi-impedance	5	15	4	12	ns
25	4	CLK to address hi-impedance	5	15	4	12	ns
26		CLK to \overline{TS} , \overline{ABB} , \overline{DBB} hi-impedance	11	20	9	17	ns
27		CLK to output lo-impedance	5	–	4	–	ns
27A		CLK to \overline{ABB} , \overline{DBB} , \overline{TS} lo-impedance	0	–	0	–	ns
28		CLK to \overline{BR} asserted	0	11	0	10	ns
29		CLK to \overline{BR} negated	0	11	0	10	ns
30		CLK to $\overline{SSTAT1}$ and $\overline{SSTAT0}$ asserted	0	11	0	10	ns
31	3	CLK to $\overline{SSTAT0}$ and $\overline{SSTAT1}$ negated	–	17	–	15	ns
32		CLK to $\overline{SSTAT0}$ and $\overline{SSTAT1}$ hi-impedance	17	27	14	22	ns
33		CLK to PSTAT2–0 valid	5	17	4	15	ns
33A		CLK to PSTAT2–0 invalid	5	–	4	–	ns

NOTES:

1. All outputs except \overline{TS} and $\overline{SSTAT0}$ – $\overline{SSTAT1}$ are specified with an output load of 50 pF and a line length of 6 inches. \overline{TS} and $\overline{SSTAT0}$ – $\overline{SSTAT1}$ are specified with a load of 60 pF and a line length of 6 inches. All output timing specifications assume a board impedance in the range of 50–90 ohms and a dielectric constant in the range of 2–6. All output specifications are measured from the 1.4 V of the input CLK to the TTL level (0.8 or 2.0 V) of the signal in question. Outputs are measured both at the pin and at the end of the 6-inch line.
- 2–5. Are not shown here. For more complete information, refer to the *MC88110 RISC Microprocessor* technical data sheet.

Bus Arbitration Signals

There are two sets of AC electrical specifications on the MC88110 microprocessor's output signals. These are 0/10 ns for arbitration signals (for example, transfer start (\overline{TS}), address bus busy (\overline{ABB}), data bus busy (\overline{DBB}), and bus request (\overline{BR})) and 4/15 ns for transfer attribute signals (for example, address bus (A31 through A0), read/write (R/ \overline{W}), and transfer burst (\overline{TBST})). Since this design is a single master design, it is not necessary to arbitrate for mastership of the bus. The MC88110 always has the bus (i.e., if it is parked permanently). The following discussion, which relates to \overline{TS} , \overline{ABB} and \overline{DBB} , can be applied to systems that are required to arbitrate for the bus since these signals have the same output transition specifications.

The memory interface on the ADS is governed by state machines resident in 7 ns registered PALs that are required to meet the setup and hold requirements for the PALs when the ADS is running at 50 MHz. At 50 MHz (a 20-ns period) the \overline{TS} signal has 10 ns of setup to the next rising clock edge. A 7 ns PAL requires 7 ns setup to its rising clock edge. This allows 3 ns of margin with which to account for clock skew and board layout considerations. The next slower speed PAL, 10 ns, requires 10 ns of setup, which would not allow any margin and is therefore unacceptable. The hold requirement of a 7-ns registered PAL is 0 ns. The AC electrical specifications show that \overline{TS} can negate as early as 0 ns. While this meets the hold requirements for the PAL, little margin is provided for clock skew between the PAL and the MC88110. Note 1 of Table 3 states that \overline{TS} is specified with a load of up to 60 pF and a line length of 6 inches. The 0 ns hold specification for the MC88110 is representative of an unloaded signal. By loading this signal, as we have done on the ADS, we provide additional margin to allow for clock skew which may be present as a result of process, board layout, etc. When designing a system at 50 MHz, care must be taken during design process (for example, layout, signal loading, and clock routing) to account for signal skews that may be added to the design.

It is also worth noting that all of the transfer attribute signals from the MC88110 have output timings of 4/15 ns from clock rising. These signals do not meet the setup times to the 7 ns PALs until two clocks into the transaction (rising edge of T2); see Figure 6. The \overline{DBB} signal is used to qualify combinatorial signals that use transfer attribute signals from the MC88110 as inputs (for example, chip selects). The \overline{DBB} signal asserts one clock into the transaction (i.e., data bus is always granted) and as a result, the transfer attribute signals are valid. The \overline{ABB} signal cannot be used to qualify these signals since it does not meet the setup time for the transfer attribute signals.

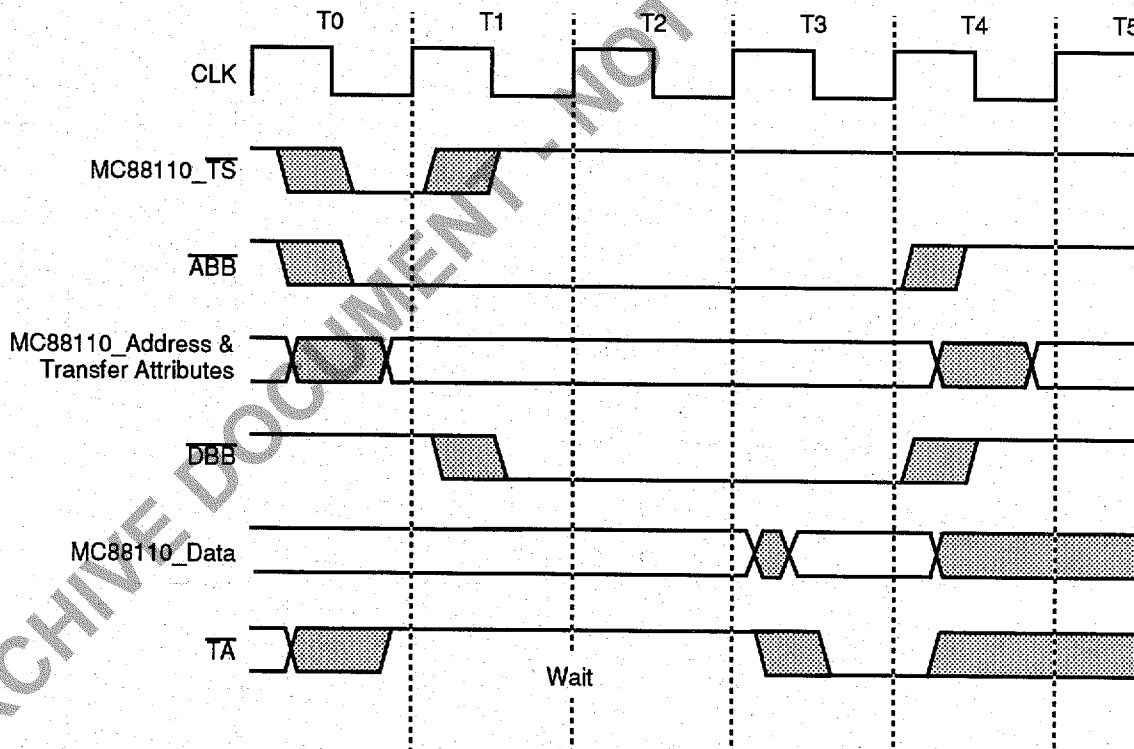


Figure 6. MC88110 Read Cycle Example

Address Decode

All resources on the ADS are memory mapped using A31–A29 signals. This divides the memory map up into eight 512-Mbyte sections which are summarized in Table 4. The 'Section' column refers to the hex encoding of the high order address nibble (A31–A28). In all instances, the peripherals do not use the entire

512-Mbyte address space allocated to them. This results in these peripherals being 'mirrored' into the higher address space (for example, EPROM(0x00000000) = EPROM(0x0XX00000)).

Table 4. ADS Memory Map

Section	Memory Map	Decode	Comments
0/1	0x00000000 – 0x0007FFFF	EPROM	512-Kbyte EPROM
0/1	0x00080000 – 0x1FFFFFFF	EPROM	EPROM space mirrored
2/5	0x20000000 – 0x5FFFFFFF	Reserved (see note)	
6/7	0x60000000 – 0x7FFFFFFF	DUART	DUART channel A base register (HOST) is at location 0x60000007. Channel B (Terminal) base register is at 0x60000047.
8/9	0x80000000 – 0x9FFFFFFF	ADI port	ADI port is memory mapped to location 0x80000004
A/D	0xA0000000 – 0xDFFFFFFF	Reserved (see note)	
E/F	0xE0000000 – 0xE01FFFFF	DRAM	2 Mbyte (256 Kbyte SIMM's)
E/F	0xE0200000 – 0xE07FFFFF	DRAM	8 Mbyte (1 Mbyte SIMM's)
E/F	0xE0800000 – 0xE1FFFFFF	DRAM	32 Mbyte (4 Mbyte SIMM's)
E/F	0xE2000000 – 0xFFFFFFFF	DRAM	DRAM space mirrored

NOTE: Transactions to reserved space are terminated with TEA resulting in an instruction or data access exception being taken.

Data

Data from the MC88110 is available 4/15 ns into the data bus tenure. Since \overline{DBG} is asserted to the MC88110 for all of its transactions, the data bus tenure begins in the clock after \overline{TS} (i.e., T1). The MC88110 meets the speed requirements for all of the peripheral devices on the ADS. A similar problem of qualifying these data signals, as shown for the address signals, would exist in systems that require that data be latched as early as possible. Fortunately, buffers require less setup time than 7 ns PALs; typically 2.5 ns for FAST family logic compared to 7 ns for 7 ns PALs.

MC88110 Inputs

All inputs to the MC88110 are specified with respect to the quadrature clocks. The resulting AC timing requirements are shown in Table 5. For the latest AC timing specifications, refer to the *MC88110 RISC Microprocessor* technical data sheet.

Table 5. MC88110 Input AC Timing Specifications

$V_{dd}=5.0 V_{dc} \pm 5\%$, $GND=0 V_{dc}$

Num.	Notes 1,3	Characteristics	40 MHz		50 MHz		Unit
			Min	Max	Min	Max	
5		Data, byte parity in valid to CLK (setup)	12	–	9	–	ns
6		Clk to data, byte parity in invalid (hold)	-4	–	-3	–	ns

Table 5. MC88110 Input AC Timing Specifications (Continued)

$V_{dd}=5.0 V_{dc} \pm 5\%$, $GND=0 V_{dc}$

Num.	Notes 1,3	Characteristics	40 MHz		50 MHz		Unit
			Min	Max	Min	Max	
7		\overline{PTA} , \overline{TA} , \overline{TEA} , \overline{TRTRY} , \overline{AACK} valid to CLK (setup)	12	-	9	-	ns
7A		\overline{ARTRY} , \overline{SHD} valid to CLK (setup)	12	-	8	-	ns
8		CLK to \overline{PTA} , \overline{TA} , \overline{TEA} , \overline{TRTRY} , \overline{AACK} , \overline{ARTRY} , \overline{SHD} in invalid (hold)	-4	-	-3	-	ns
9		\overline{DBG} and \overline{BG} valid to CLK (setup)	12	-	9	-	ns
10		CLK to \overline{DBG} and \overline{BG} invalid (hold)	-2	-	-1	-	ns
11	4	Address valid to CLK (setup)	6	-	4	-	ns
12	4	CLK to address invalid (hold)	2	-	2	-	ns
13		\overline{SR} in valid to CLK (setup)	12	-	9	-	ns
14		CLK to \overline{SR} in invalid (hold)	-2	-	-1	-	ns
15		\overline{ABB} and \overline{DBB} in valid to CLK (setup)	12	-	9	-	ns
16		CLK to \overline{ABB} and \overline{DBB} in invalid (hold)	-2	-	-1	-	ns
17	2	\overline{NMI} , \overline{INT} , \overline{RST} and \overline{DBG} valid to CLK (setup)	6	-	4	-	ns

NOTES:

1. All input specifications are measured from the TTL level (0.8 or 2.0 V) of the signal in question to the 1.4 V of the input CLK. Input timings are measured at the pin.
2. These signals will pass through one clock of debounce circuitry internal to the processor before being functionally recognized. They need to be held asserted for at least the full span of one clock cycle.
- 3-4. Are not shown here. For more complete information, refer to the *MC88110 RISC Microprocessor* technical data sheet.

Data

To support the HPM that plugs into the MC88110 socket, the memory system must meet the data setup and hold times for the MCM62110. The data requirements for the MC88110 are 9 ns setup to the rising edge and -3 ns hold to the rising edge. The negative hold time is representative of the MC88110 actually latching data on the third quadrature clock (i.e., half way through clock low). The MCM62110 latches data on the rising clock edge, one quadrature clock later than the MC88110. The resulting data setup and hold times are 9 ns and 2 ns, respectively. In systems that are supporting a zero wait state design, the data from the memory system must be capable of switching to the next "beat" of data in 9 ns ($20-9-2=9$) minus the clock skew between the MC88110 and the MCM62110. The fastest transaction on the ADS is a three-clock burst, which alleviates the problem on this design.

\overline{PTA} and \overline{TA} Signals

The \overline{TA} signal is generated from the pre-pretransfer acknowledge (\overline{PPTA}) signals that are driven by the logic sections controlling their respective accesses. The \overline{PPTA} signal is generated two clocks before the \overline{TA} needs to be generated back to the MC88110. This is possible because the ADS is an entirely deterministic design (i.e., all transactions are serviced at known rates).

The \overline{PPTA} signal is used to generate the \overline{PTA} signal back to the MC88110. That signal prevents further decoupled cache accesses from occurring. The \overline{PTA} signal is also used to satisfy the $\overline{Early_TA}$ signal generation for the HPM since it is guaranteed to occur one clock before the \overline{TA} (this same method is also used for burst transactions). Then the \overline{PTA} signal is clocked to generate the \overline{TA} back to the MC88110. Figure 7 presents a conceptual view of this description. The \overline{PTA} signal is generated correctly for all transactions, even if no data is transferred (for example, invalidation).

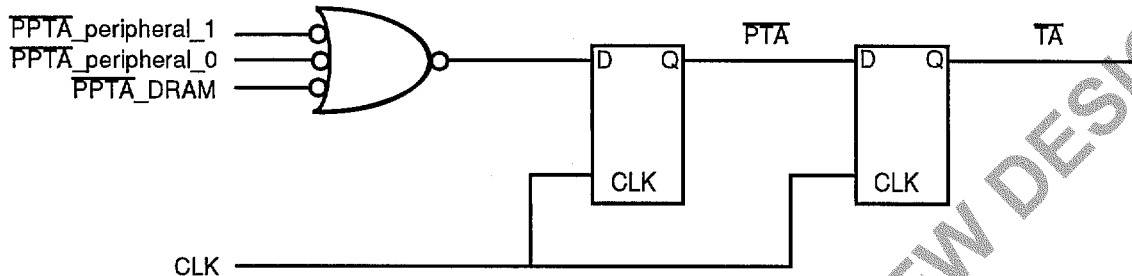


Figure 7. PPTA/PTA Generation

\overline{TEA} Signal

The address decode PAL generates the transfer error acknowledge (\overline{TEA}) signal. The \overline{TEA} signal is used to generate transfer error replies to the MC88110 if a reserved address space is accessed or an invalid transaction is attempted (for example, write to EPROM or burst from ADI port) or the watchdog timer times out (see Appendix C "PAL Equations" for a complete listing of the \overline{TEA} equation). The \overline{TEA} signal is synchronized to the MC88110 bus using a 74F5074 D-type flip flop.

PSTAT(2-0) Signals

The three PSTAT signals provide limited visibility of CPU internal status. The PSTAT(2-0) signals are useful for functions such as debugging system hardware and/or software, monitoring all write (store) traffic to cache memory (even in copyback mode), and evaluating performance.

These bidirectional signals normally function as outputs that are updated every clock cycle; see Table 3, number 33 (4/15 ns). During reset, however, the output buffers are in a hi-impedance state and the signals function as inputs. There are eight sets of internal signals that can be selected for these outputs. The three-bit value loaded through PSTAT(2-0) at reset determines which set of internal signals is selected. The selection can only be made during reset (i.e., this function is not dynamically programmable during normal operation). For a complete listing of the modes in which PSTAT can be configured, refer to the *MC88110 Second Generation RISC Microprocessor User's Manual*.

During reset the signals are sampled on every clock cycle in which reset is asserted (see Table 5, number 5 (9/-3 ns)). There is a minimum three clock cycle delay from the negated edge of reset until the PSTAT output buffers are enabled. This provides a window during which the off-chip driving logic can be put into a state of hi-impedance, thus avoiding possible bus contention when the MC88110 output buffers are enabled.

The ADS drives the PSTAT signals using a 74F244 buffer. The reset signal used as output to enable the buffer is $2\overline{reset}$ & $5\overline{reset}$. These signals are three clocks apart and $2\overline{reset}$ also resets the MC88110. This assures that the buffers are not overdriven when entering or exiting the reset state. This is illustrated in Figure 8.

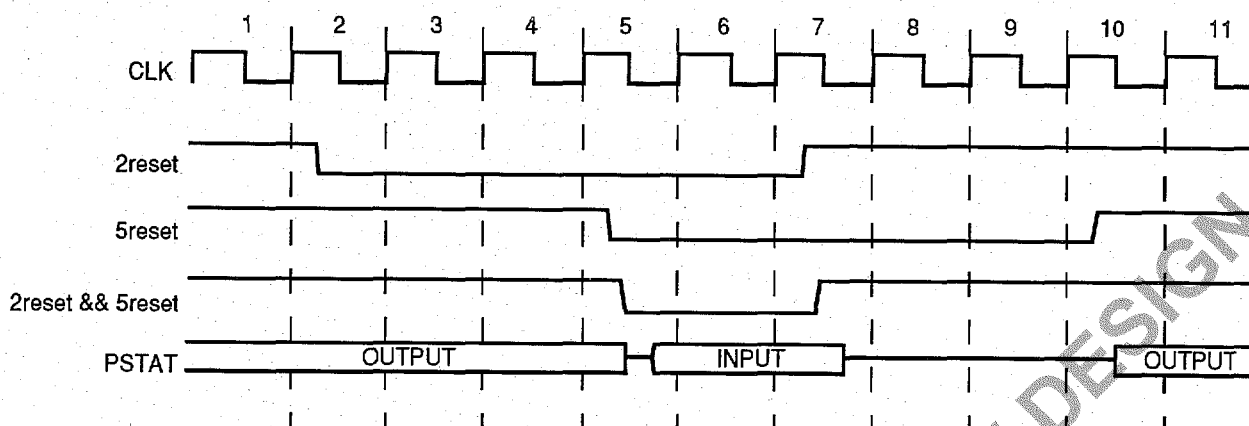


Figure 8. PSTAT Timing

Clock and Reset Signals

The clocking requirements of the MC88110 are shown in Table 6. The MC88915 clock driver is well suited to the needs of the MC88110 and the ADS system. This solution provides five clock outputs tightly coupled at 50 MHz and 1 (each) CLK, 2xCLK and 1/2xCLK outputs. Routing lengths of all clock signals are constant (approximately 7 inches). Separate discrete terminations are provided for each of the clock signals. The layout of the MC88915 is critical to the overall clock solution. Refer to *Low Skew CMOS PLL Clock Driver* (a document in *Timing Solutions* data book, order number: BR1333/D) for a more detailed discussion of layout considerations.

Table 6. MC88110 Clock AC Timing Specifications

$V_{dd}=5.0 V_{dc} \pm 5\%$, $GND=0 V_{dc}$

Num.	Notes	Characteristic	40 MHz		50 MHz		Unit
			Min	Max	Min	Max	
		Frequency of Operation	33	40	33	50	MHz
1		CLK Cycle Time	25	30	20	30	ns
2	1	CLK Rise Time	—	2	—	2	ns
3	1	CLK Fall Time	—	2	—	2	ns
4		CLK Duty Cycle Measured at 1.4 V	40	60	40	60	%
4a		CLK Pulse Width High Measured at 1.4 V	10	22	8	22	ns
4b		CLK Pulse Width Low Measured at 1.4 V	10	22	8	22	ns

NOTE: While the rise and fall times for the CLK input will be measured from 0.8 to 2.0 V, the CLK signal is expected to swing from 0.5 to 2.4 V.

The MC88915 has two selectable clock reference inputs SYNC(1) and SYNC(0). The SYNC(1) input is connected to the crystal oscillator output. The SYNC(0) input is connected to the BNC/SMA connector (BNC1). The reference select signal is connected to SW4, which is the top position of the SW(4-5) dip switch. This allows an external clock to be connected to the ADS board to generate all of the clocking requirements of the board (through the MC88915 clock driver). See Figure 9 for a graphic illustration.

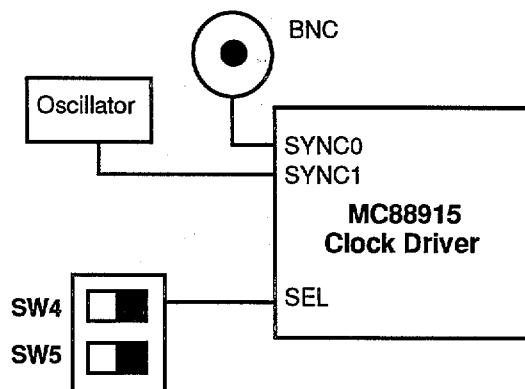


Figure 9. MC88915 Clock Source Selection

Table 7 summarizes the clock signals on the ADS.

Table 7. ADS Clock Summary

MC88915 Clock Output	Frequency (MHz)	Route (Inches)
sys_clk(0)	50	6.92082
sys_clk(1)	50	7.00147
sys_clk(2)	50	6.9364
sys_clk(3)	50	6.93573
sys_clk(7)	25	11.75965
sys_clk(8)	6.25	1.58764

NOTE: Sys_clk(7) is a split route with route to RES1 designed to be cut if RES1/RES2 thermal junction temperature function is used.

The 25 MHz clock signal (sys_clk(7)) serves a dual purpose. It is used by the refresh circuitry, to divide down to a 6.25 MHz signal (sys_clk(8)), and is also used to provide a half-speed clock (HCLK) signal which is used on the HPM in half-speed mode.

Memory

The ADS platform comes with 2/8/32 Mbytes of DRAM memory and 512 Kbytes of EPROM. The DRAM array is composed of eight SIMM packages to minimize the area required for the DRAM array and to allow for different size configurations. Both the DRAM array and EPROM array respond to accesses by driving data on the entire 64-bit wide data path. Software determines which size SIMM DRAM module is loaded into the SIMM sockets by determining when its data space overlaps into previously mapped space. Table 8 shows which Motorola DRAM SIMMs are supported on the ADS.

Table 8. DRAM Memory Modules

DRAM SIMM	Size
MCM84256	2 Mbyte
MCM81000	8 Mbyte
MCM84000	32 Mbyte

NOTE: DRAM SIMMs that do not require \overline{WE} negated during \overline{CAS} before \overline{RAS} refresh cycles are supported.

DRAM

This design uses 70 ns DRAMs. The DRAM controller is implemented in PALs and is modularized to allow using lower frequencies of operation and slower speed DRAM SIMMs. This is accomplished by changing the JEDEC files of the PALs; see Appendix C "PAL Equations". Note that in order to support slower frequency SIMMs, the JEDEC files must be changed. Slower board frequencies are inherently supported since this is a synchronous design (i.e., slower frequency only lengthens clock periods). However, it is possible that using a slower frequency will allow for the DRAM control signals to be driven earlier in their state machines. This would optimize the bus transactions at the expense of processor frequency. Table 9 summarizes the different DRAM configurations and the resulting access speeds available using this discrete solution. There is no benefit in using 60 ns DRAMs over the 70 ns DRAMs.

Table 9. Burst Read Memory Performance

DRAM Speed	Burst Access 50MHz	Size
100 ns	9:4:4:4R	2/8/32 Mbyte
80 ns	8:3:3:3R	2/8/32 Mbyte
70 ns	7:3:3:3R	2/8/32 Mbyte
60 ns	7:3:3:3R	8*/32 Mbyte

* 1 Mbyte x 8-bit was made available 4Q91 and 256K version is not available.

The signals of primary interest during DRAM accesses are shown in Figure 10. In a DRAM design, ideally the access time into the memory should be specified by the speed of the DRAM (t_{RAS}).

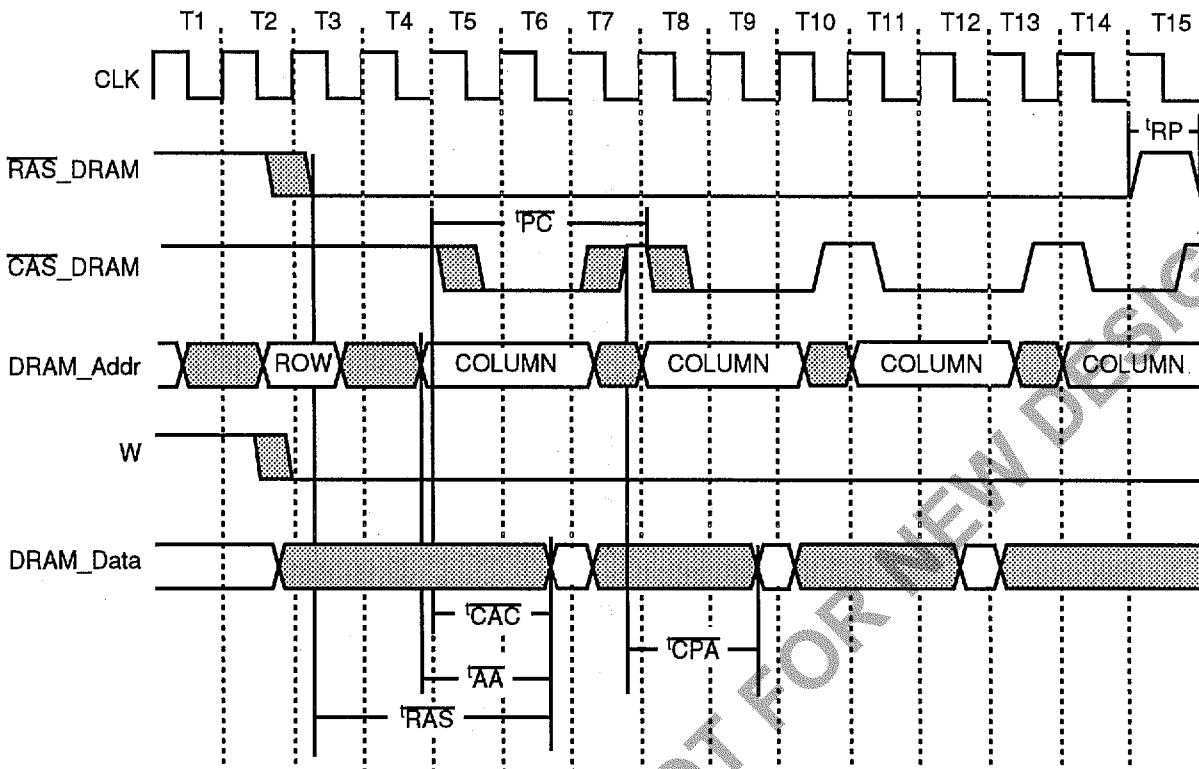


Figure 10. DRAM Signals

This is the case for the ADS DRAM interface and is highlighted in Figure 11. The access time from t_{CAC} , shown as 20.3 ns, has only .3 ns before it becomes the limiting parameter (t_{CAC} equals 20 ns for 60/70/80 ns DRAMs). Thus, it is apparent that 60 ns DRAMs cannot improve the initial access time because t_{CAC} remains identical to the 70 ns specification. This memory interface has a resolution of 10 ns with which to assert/negate signals (for example, 50 MHz:20 ns period, using CLK as an input). As a result, $\overline{CAS}(7-0)$ assertion cannot be improved while still meeting the column address setup and hold requirements to $\overline{CAS}(7-0)$. This results in t_{CAC} becoming the limiting access parameter with 60 ns DRAMs.

The 80 ns \overline{RAS} specification for the 80 ns DRAM results in an initial access one clock slower than the 70 ns DRAM. The burst access time remains a three-clock access for all speed DRAMs except the 100 ns DRAM which requires another one clock on each burst transaction. The burst access speed is determined by four specifications on the DRAM— t_{PC} , t_{CAC} , t_{AA} and t_{CPA} . Figure 11 highlights that this design has 6.5 ns setup to the 74F543s, which require 0 ns setup and 2 ns hold. As a result, the 5 ns difference between t_{AA} on the 70 ns and 80 ns merely cuts into the 6.5 ns of the available margin (maintaining a three-clock burst rate). In going to the 100-ns speed DRAM, the additional 15 ns (50 ns vs. 35 ns) in this specification results in an additional clock on the burst transactions.

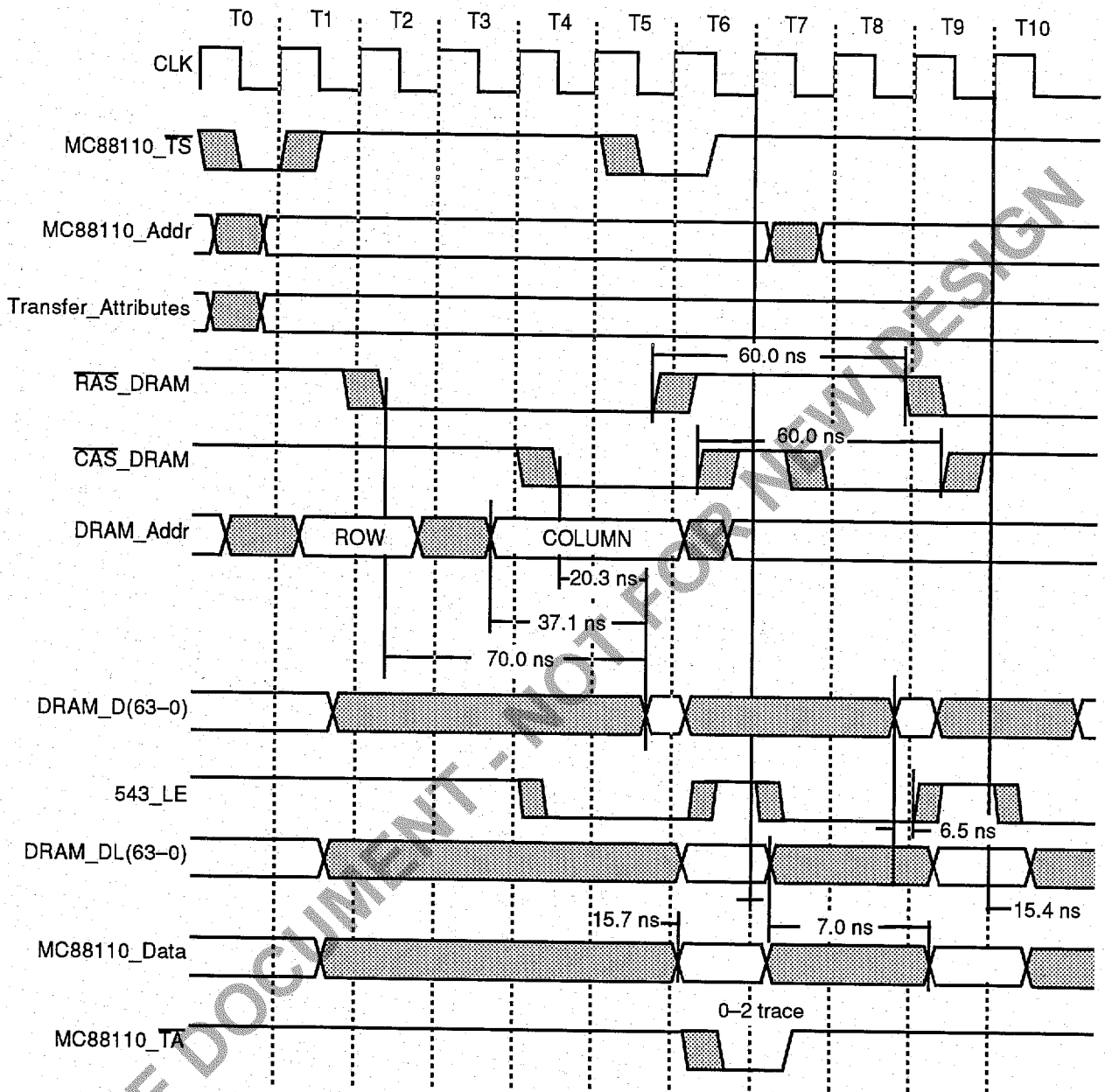


Figure 11. ADS Fast Page Mode Read Cycles

Table 10 summarizes the values of t_{PC} , t_{CAC} , t_{AA} and t_{CPA} for the 4-Mbyte DRAMs.

Table 10. Fast Page Mode Read Cycles Specifications

Spec/DRAM Speed (ns)	60	70	80	100
t_{PC} —fast page mode cycle time	45	45	50	60
t_{CAC} —access time from \overline{CAS}	20	20	20	25
t_{AA} —access time from column address	30	35	40	50
t_{CPA} — \overline{CAS} precharge time	40	40	45	55
t_{RP} — \overline{RAS} precharge	40	50	60	70

NOTE: t_{RAS} (access time from \overline{RAS}) is the speed of the DRAM

The initial access of a DRAM write transaction occurs two clocks faster than the corresponding read transaction because the data from the MC88110 is available earlier on writes (i.e., the MC88110 knows what the write data is and drives it immediately) and DRAM timing is the only delay. Note that back-to-back transactions incur another two-clock penalty due to the \overline{RAS} precharge requirement for DRAMs.

Table 11 shows the clock delays between transactions inherent to the MC88110 bus. This is shown to emphasize the types of transactions that will occur on the bus and the maximum frequency of those transactions. The table is organized showing the first transaction across the top row which can then be indexed to any type of second transaction to determine how many "free" cycles there will be between the $\overline{TA}/\overline{TEA}/\overline{ARTRY}/\overline{TRTRY}$ of the first transaction and the \overline{TS} of the second transaction (assuming the MC88110 is parked). The last column represents the number of clocks between any transaction that is retried and the transaction being restarted. This parameter is not applicable to this design (i.e., no transactions are retried on this single master design) but is included for completion.

Table 11. MC88110 Transaction Frequency

1st Transaction->/ 2nd Transaction->	Table Search Read	Single Beat Read	Single Beat Write	Burst Read	Burst Write ^{1,3}	Snoop Copyback	Any Instruction Access	Retry
Table Search Read	2	3	3	3	3	2	0	2
Single Beat Read	4	2	2	2	0	2	0	3
Single Beat Write	4	2	2	2	0	2	0	3
Burst Read	4	2	2	2	0	2	0	3
Burst Write ¹	4	2	2	2	2	2	0	3
Snoop Copyback	1	1	1	1	1	2	0	1
Any Instruction Access ²	0	0	0	0	0	0	2	3

NOTES:

1. Burst writes include replace and flush copyback transactions.
2. Code fetch transactions are suppressed after the retry of a data transaction.
3. Data replacement copyback and the transaction that caused it occur back-to-back.

The DRAM controller consists of five PALs. They control the types of accesses being run into the DRAM and most of the control signals required by the DRAM. An additional PAL multiplexes the column address for burst transactions. The last PAL drives \overline{CAS} into the eight separate SIMMs. Table 12 summarizes the

PALs used to control the DRAMs and their outputs. The h1_dram_access_0 PAL also generates some signals that are not specific to controlling the DRAM (i.e., \overline{TA} , \overline{rTS}).

Table 12. DRAM PAL Summary

JEDEC file	Device: Speed	Outputs
h1_dram_access_1.jed	20R8:7 ns	\overline{RAS} _refresh_en refresh_state(2-0) \overline{CAS} _en refresh_req
h1_dram_access_0.jed	20R4:7 ns	\overline{TS} _latched \overline{TA} \overline{rTS} PPTA_dram row_column
h1_dram_sequence.jed	20R6:7 ns	dram_state(4-0) \overline{RAS} _dram_en
h1_col_muxplex.jed	16R6:7 ns	dram_ma(2-0)
h1_byte_select.jed	20L8:7 ns	\overline{CAS} (7-0)

NOTE: See Appendix D, "Schematics", for graphic illustrations of the JEDEC file

There are additional signals related to controlling address and data paths for DRAM accesses. The generation of these and other control signals is discussed in greater detail in the "Bus Interface Controller" section.

EPROM

The EPROM is comprised of four TMS27C210-200 parts. This provides a 64-bit data path and 512 Kbytes of memory. The EPROM resides on the peripheral bus with the ADI port and DUART. Burst accesses into the EPROM are supported. This is determined by checking to see if \overline{DBB} is asserted on the transaction following \overline{TA} . After the initial access is made into the EPROM, \overline{DBB} is checked after \overline{TA} is asserted to determine whether an additional "beat" will follow. If this is a single beat transaction or the last beat of a burst transaction then \overline{DBB} is guaranteed to negate for one clock. During that clock the state machines controlling the access enter an output disable stage where they remain until the EPROM output buffers have three-stated. During this period, accesses onto the peripheral bus are "waited"; however, transactions into the DRAM are allowed to proceed.

Bus Interface Controller

The bus interface controller sequences all transactions on the ADS board. It responds to all transactions initiated by the processor node and also controls the periodic refresh cycles for the DRAMs.

DRAM Control

The dram_state(4-0) state machine resides in h1_dram_sequence PAL. This is a 5-bit state machine which "runs" during DRAM accesses. All accesses by the MC88110 assert \overline{TS} when they begin. \overline{TS} _latched is set when \overline{TS} asserts. Since \overline{TS} only stays asserted for the first access of the transaction, \overline{TS} _latched is also used to signal that a transaction has started. The state machine starts when it sees that a transaction has started from the MC88110 and it does not have a refresh request currently pending. On the next clock, an address decode check is made to determine if it should continue (i.e., a valid DRAM access has begun) or whether it should hold in S1count.

This state machine is used by both h1_dram_access PALs to generate the remaining control signals for the DRAM controller. Table 13 shows the values for the $\overline{\text{RAS}}(7-0)$, $\overline{\text{CAS}}(7-0)$ and row_column control signals for each of the states generated by the dram_state(4-0) state machine. Figure 12 shows the multiplexing technique of the address signals into the DRAM array. The $\overline{\text{RAS}}(7-0)$ control signals are driven by the outputs of two 74AS08 AND gates.

The inputs to these gates are $\overline{\text{RAS}}_{\text{refresh_en}}$ and $\overline{\text{RAS}}_{\text{dram_en}}$. The $\overline{\text{RAS}}_{\text{dram_en}}$ signal is an output signal from h1_dram_sequence.jed PAL based on state information of the dram_state(4-0) state machine. The $\overline{\text{RAS}}_{\text{refresh_en}}$ signal is an output from the h1_dram_access_1 PAL which runs the refresh_state(2-0) state machine during refresh cycles. The refresh_state(2-0) state machine "runs" when a refresh request has been registered and the dram_state(4-0) has reached the S0count state.

The multiplexed address to the DRAM array is provided by using three 74AS258s to multiplex the address and then using eleven 74AS244s to buffer the address to the DRAMs. This keeps the loading on each address line down to 70 pF (maximum), which is the maximum capacitance rating on the SIMMs. Note that the typical capacitive load presented by the DRAM SIMMs is 47 pF. A 1-ns derating factor is added to the timing of the 74AS244s (determined by using typical derating curves provided by TI for its AS family).

In order to conserve power, the 74AS244s are output disabled until a memory cycle is begun ($\overline{\text{ABB}}$ asserts). These address buffers drive regardless of which memory access is being made (i.e., there is not enough time to decode just the DRAM address space). The row_column signal selects whether row or column addresses are being driven to the DRAMs (S input on the 74AS258s). The routed length of the address lines used to access the DRAM sys_addr(24-3) vary from 2.10611 to 5.0837 inches. The lowest three address lines sys_addr(2-0), used by the byte select logic, vary in routed length from 6.66563 to 7.63959 inches. Since this is greater than the specified 6 inches, consideration must be given to the effects of the additional trace length with respect to transmission lines.

Table 13. $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ and row_column Control States

Counter Value	$\overline{\text{CAS}}$ Before $\overline{\text{RAS}}$ Refresh	Single Beat Read	Single Beat Write	110/410 Burst Read	110/410 Burst Write	410L Burst Read	410L Burst Write
S0count/R0count	1/1/1	1/1/1	1/1/1	1/1/1	1/1/1	1/1/1	1/1/1
S1count/R1count	1/0/1	0/1/1	0/1/1	0/1/1	0/1/1	0/1/1	0/1/1
S2count/R2count	0/0/1	0/1/1	0/1/1	0/1/1	0/1/1	0/1/1	0/1/1
S3count/R3count	0/1/1	0/1/0		0/1/0	0/1/0	0/1/0	0/1/0
S4count/R4count	0/1/1	0/0/0		0/0/0	0/0/0	0/0/0	0/0/0
S5count/R5count	0/1/1			0/0/0	0/0/0	0/0/0	0/0/0
S6count/R6count	1/1/1			0/1/0	0/1/0	0/1/0	0/1/0
S7count/R7count	1/1/1			0/0/0	0/0/0	0/0/0	0/0/0
S8count				0/0/0	0/0/0	0/0/0	0/0/0
S9count				0/1/0	0/1/0	0/1/0	0/1/0
S10count				0/0/0	0/0/0	0/0/0	0/0/0
S11count				0/0/0	0/0/0	0/0/0	0/0/0
S12count				0/1/0		0/1/0	0/1/0
S13count				0/0/0		0/0/0	0/0/0
S14count						0/0/0	0/0/0

Table 13. \overline{RAS} , \overline{CAS} and row_column Control States (Continued)

Counter Value	\overline{CAS} Before \overline{RAS} Refresh	Single Beat Read	Single Beat Write	110/410 Burst Read	110/410 Burst Write	410L Burst Read	410L Burst Write
S15count						0/1/0	0/1/0
S16count						0/0/0	0/0/0
S17count						0/0/0	0/0/0
S18count						0/1/0	0/1/0
S19count						0/0/0	0/0/0
S20count						0/0/0	0/0/0
S21count						0/1/0	0/1/0
S22count						0/0/0	0/0/0
S23count						0/0/0	0/0/0
S24count						0/1/0	
S25count						0/0/0	
S26count			0/1/0		0/1/0		0/1/0
S27count			0/0/0		0/0/0		0/0/0
S28count		0/0/0	0/0/1	0/0/0	1/0/1	0/0/0	1/0/1
S29count		1/1/1	1/1/1	1/1/1	1/1/1	1/1/1	1/1/1
S30count							
S31count							

NOTES:

1. The refresh cycle is run with S0count maintained and guaranteed. \overline{CAS} and \overline{RAS} are generated by the h1_dram_access_1 PAL by counting through R0count–R7count.
2. Shaded boxes represent cycles in which \overline{TA} responds to the system bus. This requires that $\overline{PTA}/\overline{PPTA}$ be generated one or two clocks earlier, respectively.

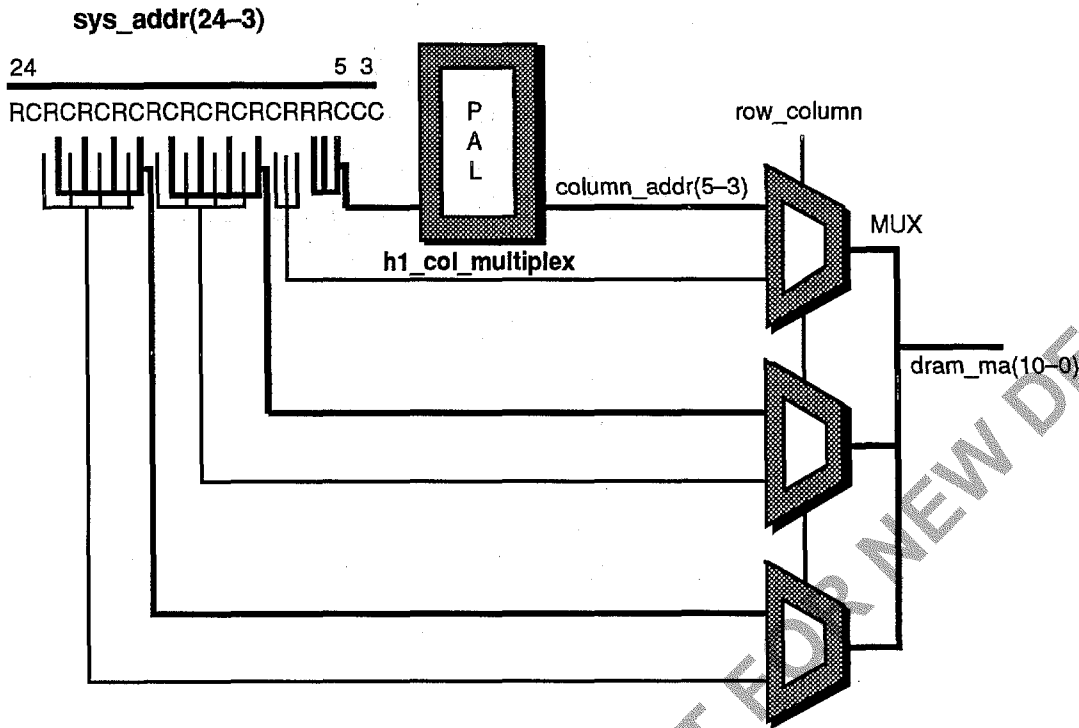


Figure 12. MC88110 DRAM Address Paths

The address signal breakout from the MC88110 into the DRAM array is shown in Figure 13. This information is shown in a different format in Table 14 and Table 15.



Figure 13. MC88110 DRAM Address Breakout

DRAM $\overline{\text{RAS}}$ precharge times are met by having the $\text{dram_state}(4-0)$ state machine count another clock at the conclusion of the access before coming back to $S0$ count. Then, if another DRAM access has started ($\overline{\text{TS_latched}}$ asserted), it can immediately begin that access. If an access to any other resource has begun, the state machine does not wait for the $\overline{\text{RAS}}$ precharge time (i.e., begins immediately).

Table 14. MC88110 DRAM Address Breakout (1 of 2)

Row/Column Assignments	
Column Address	Row Address
A23	A24
A21	A22
A19	A20
A17	A18
A15	A16

Table 14. MC88110 DRAM Address Breakout (1 of 2) (Continued)

Row/Column Assignments	
Column Address	Row Address
A13	A14
A11	A12
A9	A10
A5	A8
A4	A7
A3	A6

Table 15. MC88110 DRAM Address Breakout (2 of 2)

Byte Lane Assignments						
24	23----20	19----13	15----12	11----8	7-----4	3-----0
R	RCRC	RCRC	RCRC	RCRC	RCRR	RCCC

The data path for the next access does not 'align' until \overline{DBB} asserts. This is accomplished by latching the control signals for the previous transaction using \overline{DBB} . The next transfer attribute signals are not seen until \overline{DBB} asserts (forcing the latch driving those signals to go transparent). In this fashion, the transfer attributes for the previous transaction are latched and that transaction is allowed to complete. Delaying the next transaction's transfer attributes does not incur a penalty since all devices in this system have a minimum of two clock access latency. The system data lines vary in routed lengths from 5.05484 to 9.3204 inches. Due to the specific layout used, the longer data routes are matched up with the shorter address path routes. In this fashion, trace delay due to board effects is equalized across the bus.

The $\overline{CAS}(7-0)$ signals are generated by the h1_byte_select PAL. The timing of these \overline{CAS} signals is controlled by the CAS_en signals assertion from the h1_dram_access_1 PAL. This signal is asserted based on information from the dram_state(4-0) state machine. Which specific SIMMs \overline{CAS} signal should be asserted is determined from the transfer attribute signals for the current transaction (that is, \overline{TBST} , $\overline{TSIZ}(1-0)$ and $A(2-0)$).

Thermal Measurements

The MC88110 thermal specification is a maximum junction temperature of 110°C. Traditionally, thermal specifications of ambient or case temperature required calculations based on several variables that introduce error. To compensate for these errors, package and heatsink manufacturers added margin to their specifications (variables). These traditional methods can lead to needless oversizing of fans, heatsinks, and reduced system performance.

To improve upon these traditional methods of calculating thermal characteristics and establishing heat sinks accordingly, a thermal resistor that allows the customer to directly measure junction temperature has been incorporated on each MC88110 die. This allows the customer to accurately size fans and heatsinks while maintaining the highest frequency of operation. Each die's resistor has its own thermal coefficients and must be calibrated for accurate thermal measurements.

To determine the unique thermal characteristics for the thermal resistor, measure the voltage drop across RES1 and RES2 at a minimum of two controlled temperatures (requires a high precision variable voltage supply, amperage meter and oven), while driving a known current (for example, 1mA) into the RES1 input

and out of the RES2 output; see Figure 14. Note that ideally, these measurements are made on the actual board in the actual (or worst case) environmental conditions.

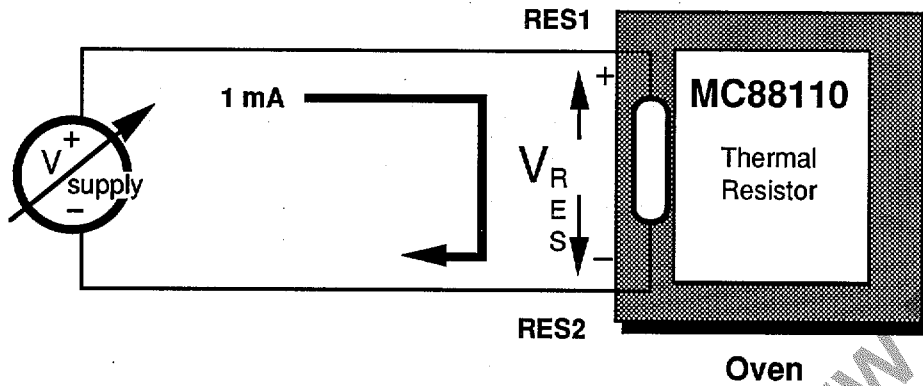


Figure 14. Thermal Test Fixture

The construction of a graph showing the linear relationship between voltage drop across the thermal resistor and the known thermal temperature can now be formed. This graph is shown in Figure 15. Then, while running the device under actual operating conditions and measuring the voltage drop across the resistor for the same current being driven into the resistor, a value can be determined for the junction temperature of the die by cross referencing on the graph generated for known values. Note these values are preliminary and are subject to change.

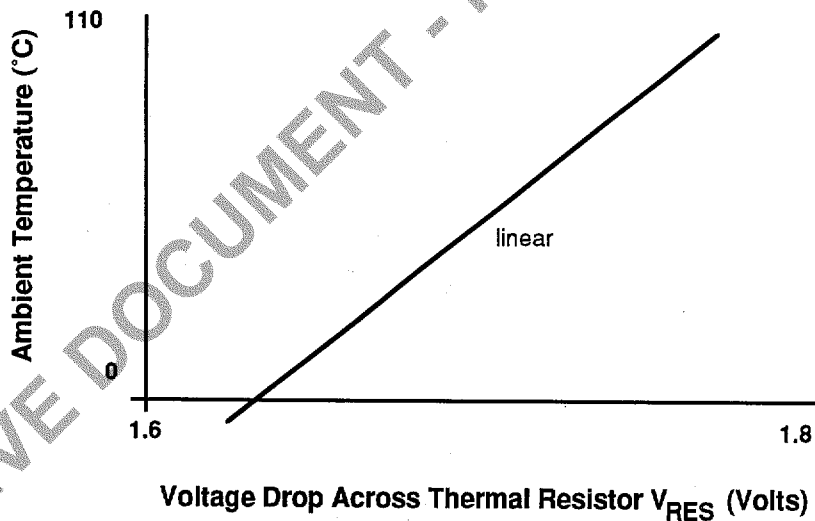


Figure 15. Voltage vs. Temperature Characterization Graph

In order to use the ADS board for thermal analysis, the RES1 and RES2 signals must be isolated from their alternative functions (HCLK and $\bar{E_TA}$). This is accomplished by cutting these traces as shown on the PC boards silkscreen. This allows these signals to be driven by the variable voltage supply in order for characterization data to be gathered. Note that this data is only applicable for the ADS board and the environment in which the data is gathered. Care must be taken when applying the findings of these measurements to other system solutions.

Appendix A
Signal Usage on the MC88110

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

The MC88110 RISC microprocessor will initially be released in a 299-pin PGA package. The signal breakout on the MC88110 is summarized in Figure A-1. The signals specific to ADS design are summarized in Figure A-2. Note that 38 of the signals used in the MC88110 design are not used in the ADS design. This is due to the fact that this is a single processor design that does not need to concern itself with arbitration or snooping. Refer to Appendix D, "Schematics" for the actual circuit depiction of the MC88110 implementation.

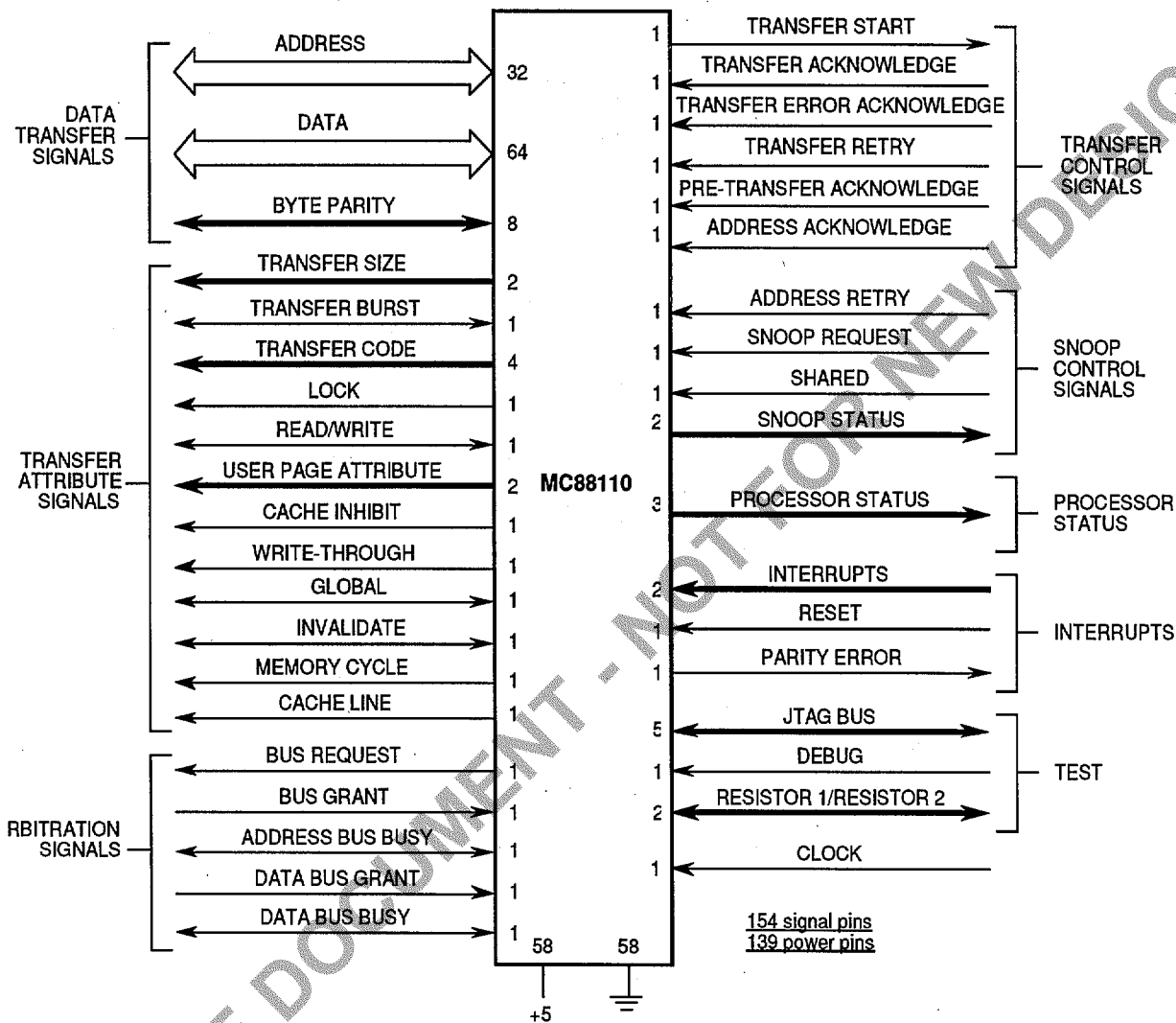


Figure A-1. MC88110 Signal Summary

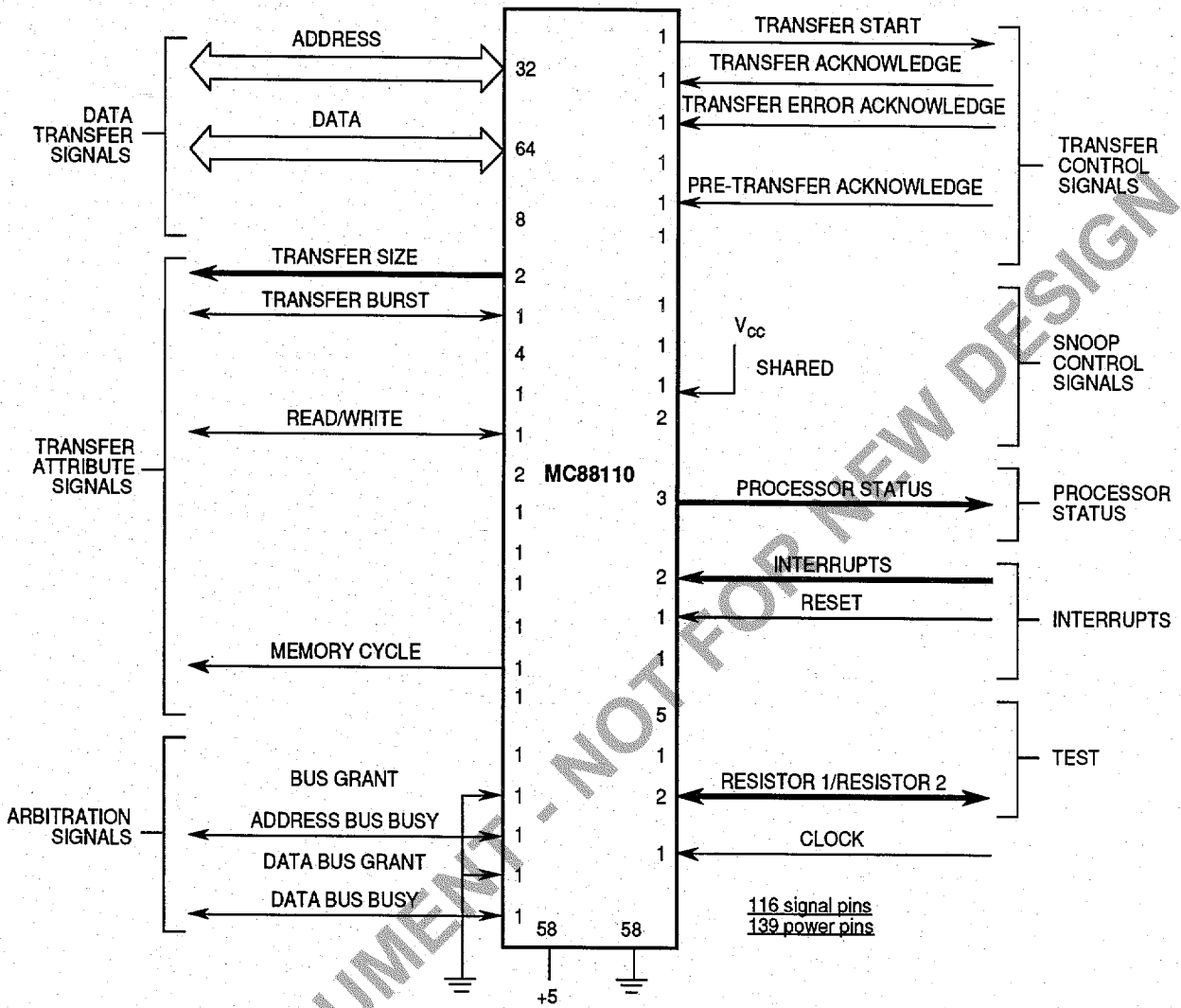


Figure A-2. ADS Actual Signal Usage

Appendix B
Layout and Routing ADS

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

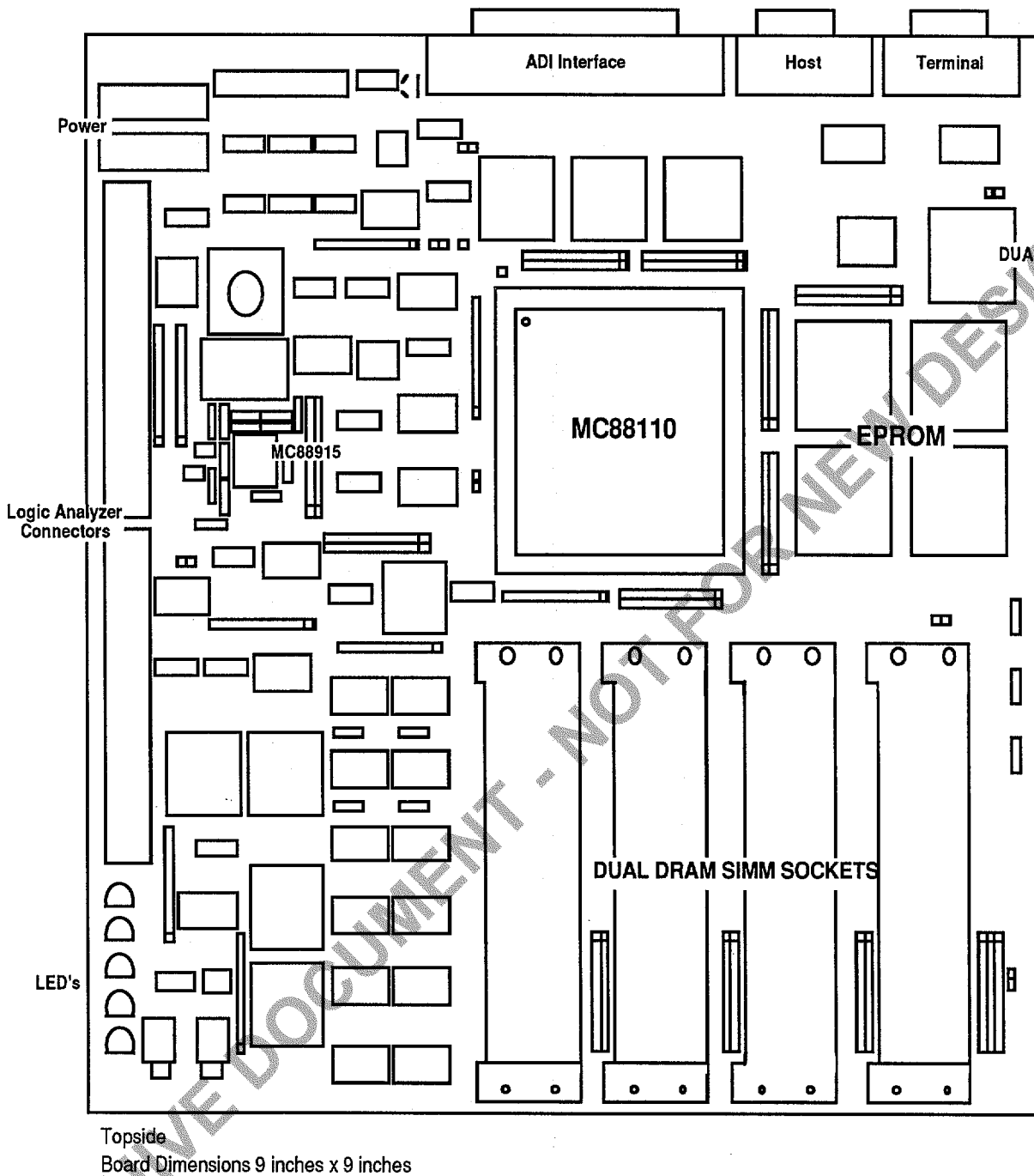


Figure B-1. Application Development System Layout (Topside)

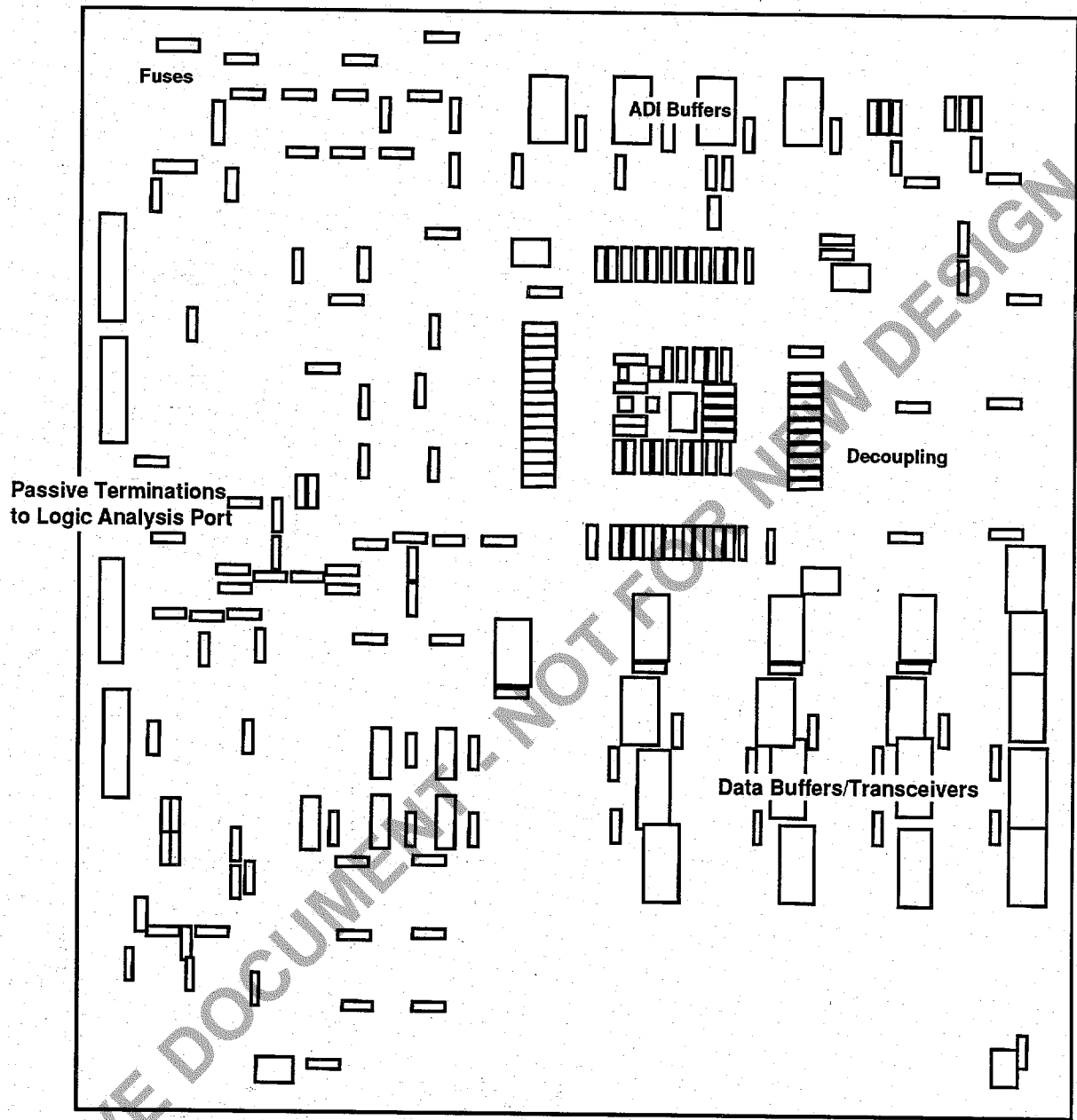
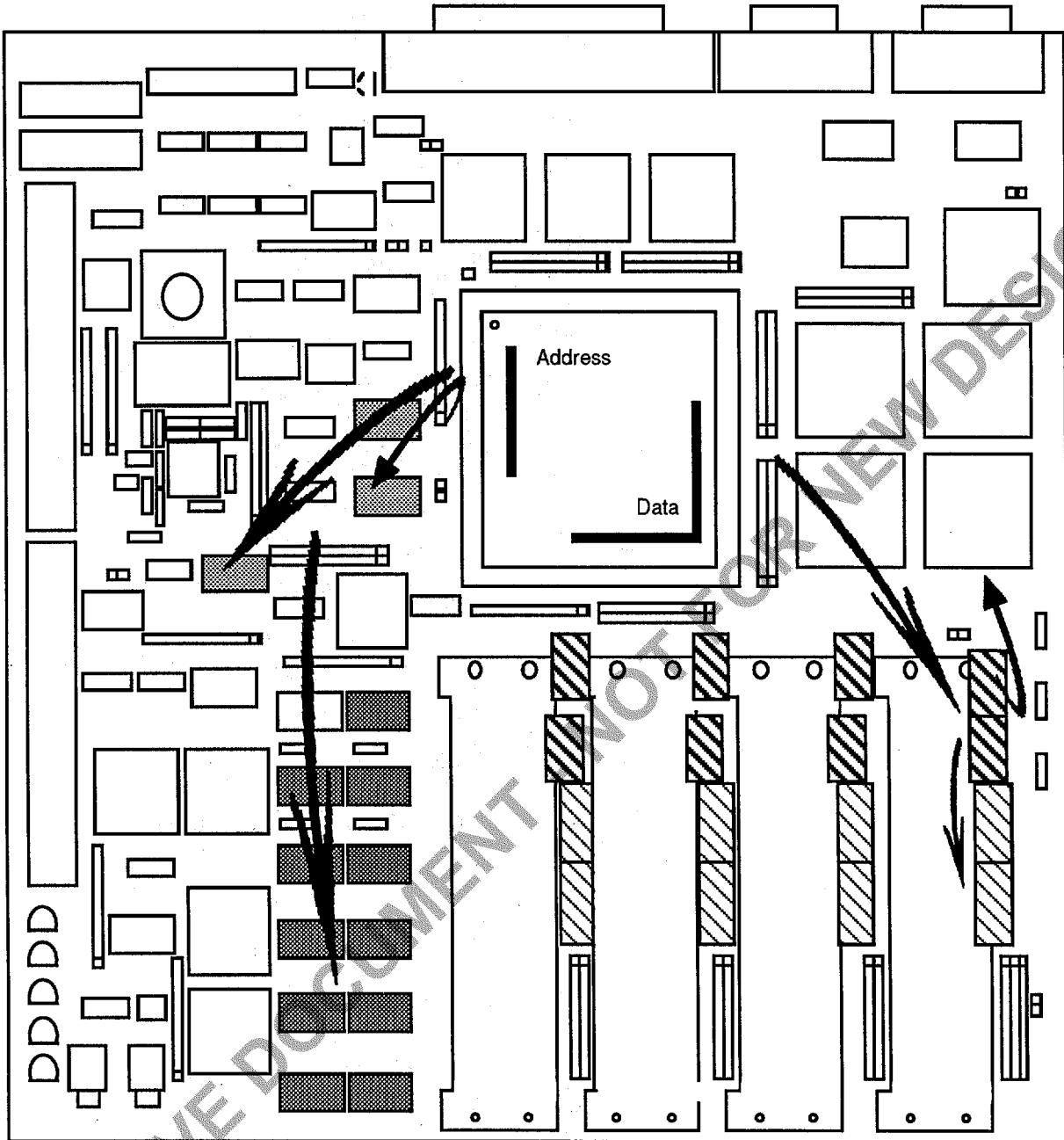


Figure B-2. Application Development System Layout (Bottomside)



- MC88110 Address Route Lengths
minimum 2.106 / average 3.306 / maximum 5.084
- MC88110 Data Route Lengths
minimum 5.055 / average 7.093 / maximum 9.320 (inches)





Legend	
Multiplexer	
Address Buffer	
Data Transceivers	
Data Latching Transceivers	

Figure B-3. Application Development System Layout (Address/Data Route Lengths)

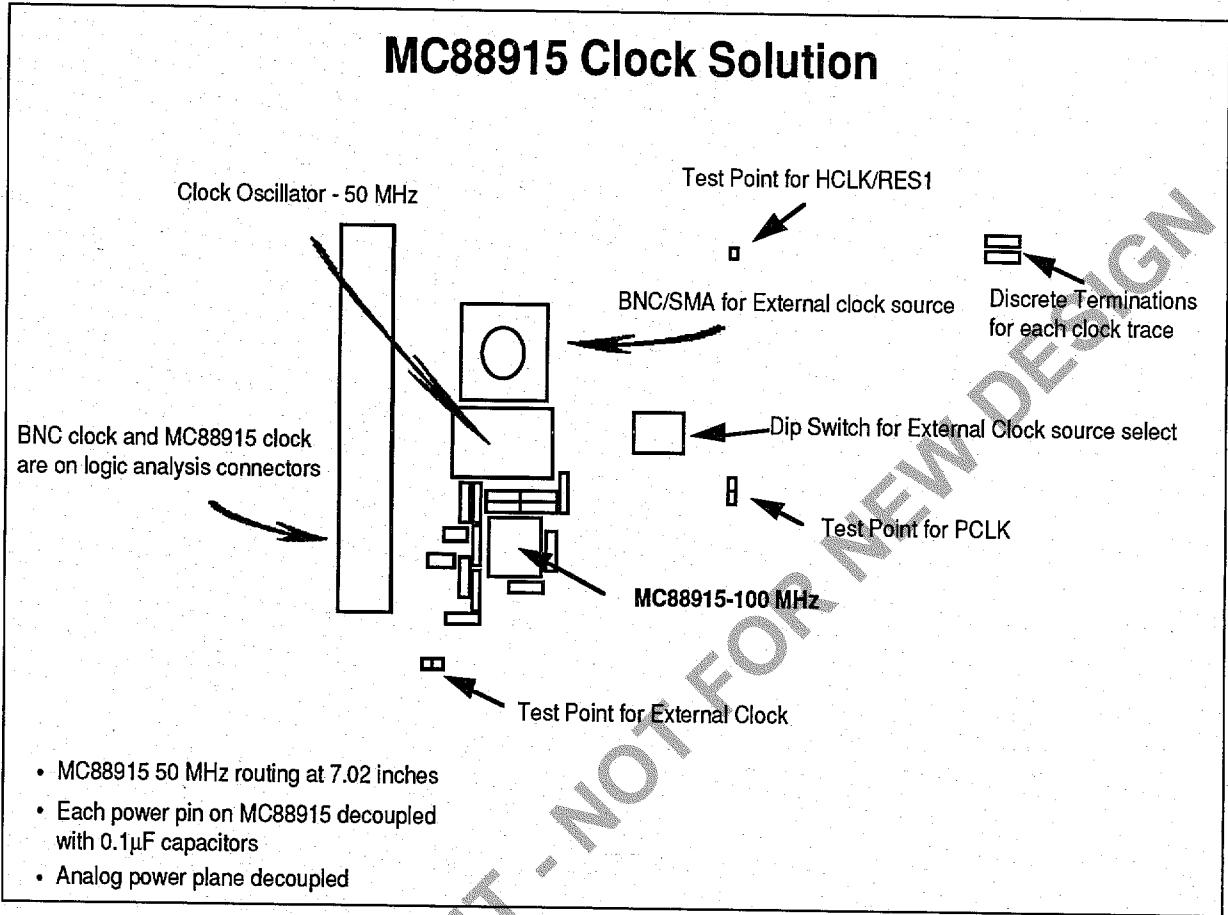
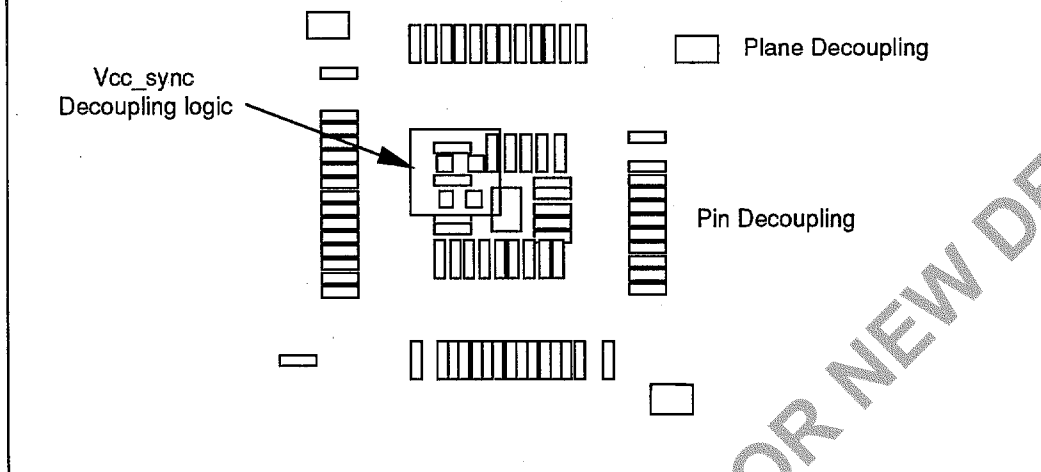


Figure B-4. Application Development System Layout (MC88915 Clock Solution)

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

MC88110 Decoupling



- Each Power pin independently decoupled with $0.1\mu\text{F}$ capacitors (66)
- Each power supply plane decoupled with $100\mu\text{F}$ and $22\mu\text{F}$ capacitors (10)

Figure B-5. Application Development System Layout (MC88110 Decoupling)

Appendix C
PAL Equations

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

```
module h1_address_decode;      flag '-r3';
title 'ADS_110'
```

C-2

```
h1_addr_decode device 'p2018';
```

```
"inputs
```

```
!watchdog_timeout    pin 1;
!TBST                pin 2;
!TA                  pin 3;
A31,A30,A29          pin 4,5,6;
!MC                  pin 7;
RW_                  pin 8;
!DBB                 pin 9;
!ODT                 pin 10;
remap                pin 23;
```

```
"outputs
```

```
"Combinatorial
```

```
!adi_port_CS        pin 16;
!dram_CS             pin 18;
!eprom_CE           pin 19;
!peripheral_CS0     pin 20;
!decode_led         pin 21;

!TEA                pin 22;      "Output only (ie. no feedback)
```

```
L,H,x,p,c,k,z      =      0,1,.x.,.p.,.c.,.k.,.z.;
e_decode            =      [A31,A30,A29];      "memory map decode bits
```

```
c_eprom            =      (e_decode == ^b000);      "memory map decoding
c_adi_port         =      (e_decode == ^b100);
c_counter          =      (e_decode == ^b110);
c_duart           =      (e_decode == ^b011);
c_off_board        =      (e_decode == ^b001);
c_control          =      (e_decode == ^b101);
c_reserved         =      (e_decode == ^b010);
c_dram             =      (e_decode == ^b111);
```

```
equations
```

```
adi_port_CS = c_adi_port & !ODT & DBB & MC #
"Begin on adi_port decode when all other peripherals
" have completed on the peripheral bus and DBB has
" asserted qualifying the address signals
" and a memory cycle

adi_port_CS & DBB;      "Hold for duration of data tenure
```

()

()

()

```

peripheral_CS0 = !(c_dram # c_reserved # (c_eprom & remap)) & !ODT &
                DBB & MC #
                peripheral_CS0 & DBB;

decode_led     = c_eprom & !ODT & DBB &          "Initially the decode_led will signal an access to the
                MC #                               " EPROM
                decode_led & DBB;

dram_CS        = c_dram & DBB & MC #             "Signal is used to enable the DRAM data path
                (c_eprom & remap) & DBB & MC #    "Note: ODT is not used to qualify the dram data
                dram_CS & DBB;                   " path because it is an independent data path
                                                " Only the peripheral data path could have
                                                " outputs still disabling when ODT is signalled

eprom_CE       = c_eprom & !ODT & !remap &      "Note: remap will decode eprom into dram
                DBB & MC & RW_ #

TEA            = c_reserved & DBB #             "Reserved memory space
                c_off_board & DBB #           "Off Board access
                watchdog_timeout & DBB #
                (DBB & c_eprom & !RW_ & !remap) # "Write to eprom when it is not remapped
                (TBST & !(c_eprom # c_dram) & DBB);
                                                "A burst into DUART/ADI will TEA that transaction
                                                "Note: This allows bursts into c_off_board
                                                " and prevents us from having to try to stop
                                                " the h1_per_ext state machine from providing
                                                " a PPTA
end h1_address_decode;

```

h1_byte_select device 'p2018';

C-4

"inputs

"outputs

!TBST pin 1;
TSIZ1 pin 2;
TSIZ0 pin 3;
!ABB pin 4;
A0 pin 5;
A1 pin 6;
A2 pin 7;
!CAS7_f pin 8;
!CAS6_f pin 9;
!CASen pin 10;
!refresh_req pin 11;

!CAS7 pin 15;
!CAS6 pin 22;
!CAS5 pin 16;
!CAS4 pin 17;
!CAS3 pin 18;
!CAS2 pin 19;
!CAS1 pin 20;
!CAS0 pin 21;

size = [TSIZ1,TSIZ0];

burst = (TBST == ^b1);
double = (size == ^b00);
word = (size == ^b01);
half = (size == ^b10);
byte = (size == ^b11);

equations

CAS0 = ((burst
double
word & !A2
half & !A2 & !A1
byte & !A2 & !A1 & !A0) & ABB & CASen)

()

()

()

```
CAS0 & CASen #
CASen & refresh_req;
```

```
CAS1 = ((burst
# double
# word & !A2
# half & !A2 & !A1
# byte & !A2 & !A1 & A0) & ABB & CASen) #
CAS1 & CASen #
CASen & refresh_req;
```

```
CAS2 = ((burst
# double
# word & !A2
# half & !A2 & A1
# byte & !A2 & A1 & !A0) & ABB & CASen) #
CAS2 & CASen #
CASen & refresh_req;
```

```
CAS3 = ((burst
# double
# word & !A2
# half & !A2 & A1
# byte & !A2 & A1 & A0) & ABB & CASen) #
CAS3 & CASen #
CASen & refresh_req;
```

```
CAS4 = ((burst
# double
# word & A2
# half & A2 & !A1
# byte & A2 & !A1 & !A0) & ABB & CASen) #
CAS4 & CASen #
CASen & refresh_req;
```

```
CAS5 = ((burst
# double
# word & A2
# half & A2 & !A1
# byte & A2 & !A1 & A0) & ABB & CASen) #
CAS5 & CASen #
CASen & refresh_req;
```

```
CAS6 = ((burst
# double
# word & A2
# half & A2 & A1
# byte & A2 & A1 & !A0) & ABB & CASen) #
CAS6_f & CASen #
CASen & refresh_req;
```

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

```
CAS7 = ((burst
# double
# word & A2
# half & A2 & A1
# byte & A2 & A1 & A0) & ABB & CASen) #
CAS7_f & CASen #
CASen & refresh_req;
```

```
end h1_byte;
```

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

```

( )
( )
( )

module h1_column_mux; flag '-r3';
title 'ADS_110'

    h1_col_mux device 'p16r6';

    CLK,OE pin 1,11;

"inputs
A3 pin 3;
A4 pin 4;
A5 pin 5;
!CASen pin 6;
!short_node pin 7;
!DBB pin 8;
!RAS_refresh_en pin 9;

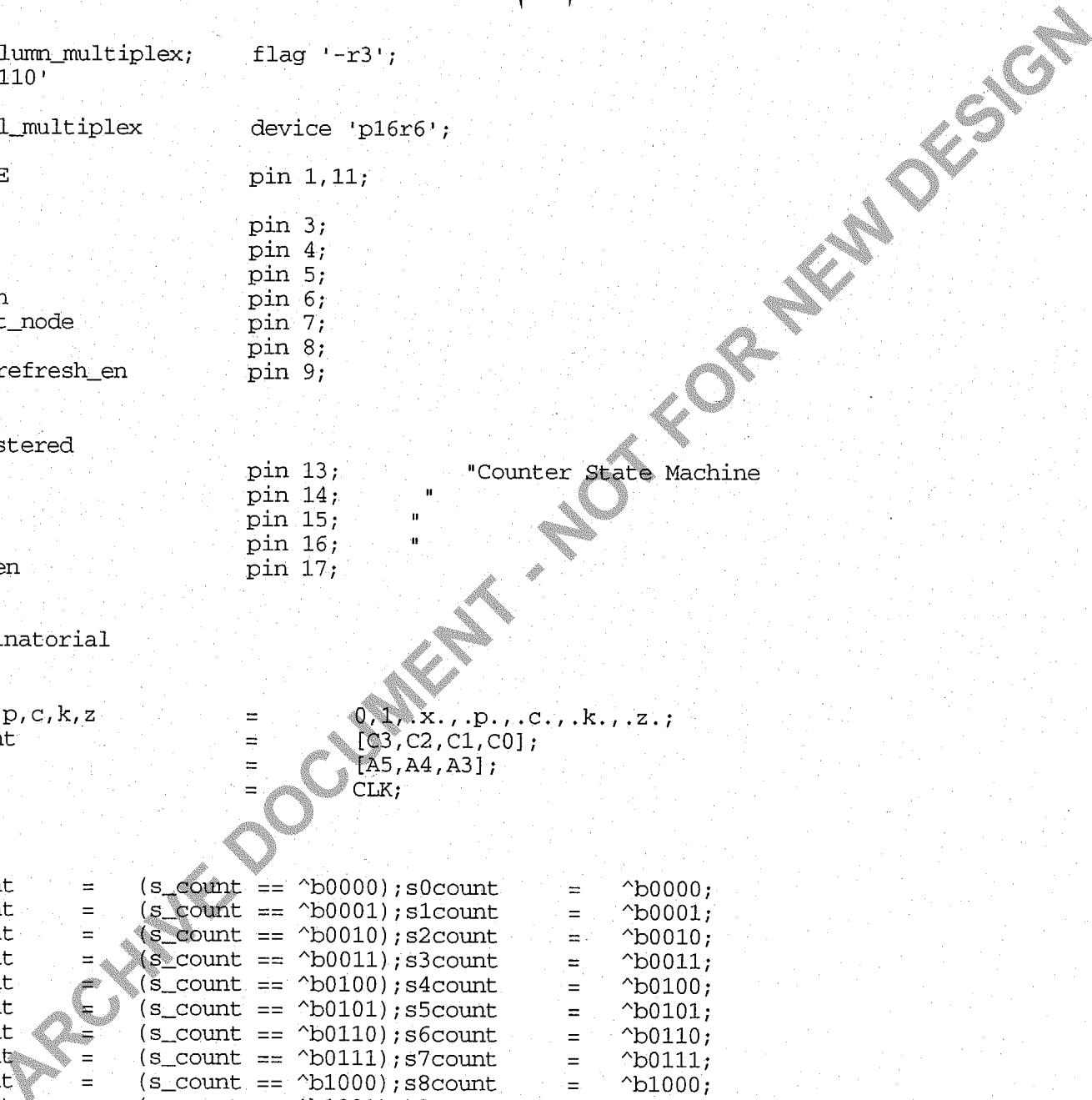
"outputs
"Registered
!C0 pin 13; "Counter State Machine
!C1 pin 14; "
!C2 pin 15; "
!C3 pin 16; "
l_CASen pin 17;

"Combinatorial

L,H,x,p,c,k,z = 0,1,.x.,.p.,.c.,.k.,.z.;
s_count = [C3,C2,C1,C0];
s_col = [A5,A4,A3];
clk = CLK;

S0count = (s_count == ^b0000); s0count = ^b0000;
S1count = (s_count == ^b0001); s1count = ^b0001;
S2count = (s_count == ^b0010); s2count = ^b0010;
S3count = (s_count == ^b0011); s3count = ^b0011;
S4count = (s_count == ^b0100); s4count = ^b0100;
S5count = (s_count == ^b0101); s5count = ^b0101;
S6count = (s_count == ^b0110); s6count = ^b0110;
S7count = (s_count == ^b0111); s7count = ^b0111;
S8count = (s_count == ^b1000); s8count = ^b1000;
S9count = (s_count == ^b1001); s9count = ^b1001;
S10count = (s_count == ^b1010); s10count = ^b1010;
S11count = (s_count == ^b1011); s11count = ^b1011;
S12count = (s_count == ^b1100); s12count = ^b1100;
S13count = (s_count == ^b1101); s13count = ^b1101;
S14count = (s_count == ^b1110); s14count = ^b1110;

```




```
State s11count:
  if(!DBB) then s0count
  else if(CASen & l_CASen & !RAS_refresh_en) then s12count
  else if(CASen & l_CASen & !RAS_refresh_en) then s8count
  else s11count;
State s12count:
  if(!DBB) then s0count
  else if(CASen & l_CASen & !RAS_refresh_en) then s13count
  else s12count;
State s13count:
  if(!DBB) then s0count
  else if(CASen & l_CASen & !RAS_refresh_en) then s14count
  else s13count;
State s14count:
  if(!DBB) then s0count
  else if(CASen & l_CASen & !RAS_refresh_en) then s15count
  else s14count;
State s15count:
  if(!DBB) then s0count
  else if(CASen & l_CASen & !RAS_refresh_en) then s12count
  else if(CASen & l_CASen & !RAS_refresh_en) then s0count
  else s15count;

end    h1_column_multiplex;
```

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

```
S15count = (s_count == ^b1111);s15count = ^b1111;
```

equations

```
short_node = 0;
```

```
l_CASen := CASen;
```

"state machines

```
-----  
state_diagram s_count  
  State s0count:  
    if(!DBB) then s0count  
    else if(s_col == ^b000) then s8count  
    else if(s_col == ^b001) then s9count  
    else if(s_col == ^b010) then s10count  
    else if(s_col == ^b011) then s11count  
    else if(s_col == ^b100) then s12count  
    else if(s_col == ^b101) then s13count  
    else if(s_col == ^b110) then s14count  
    else if(s_col == ^b111) then s15count  
  State s1count:  
    goto s0count;  
  State s2count:  
    goto s0count;  
  State s3count:  
    goto s0count;  
  State s4count:  
    goto s0count;  
  State s5count:  
    goto s0count;  
  State s6count:  
    goto s0count;  
  State s7count:  
    goto s0count;  
  State s8count:  
    if(!DBB) then s0count  
    else if(CASen & l_CASen & !RAS_refresh_en) then s9count  
    else s8count;  
  State s9count:  
    if(!DBB) then s0count  
    else if(CASen & l_CASen & !RAS_refresh_en) then s10count  
    else s9count;  
  State s10count:  
    if(!DBB) then s0count  
    else if(CASen & l_CASen & !RAS_refresh_en) then s11count  
    else s10count;
```

```

module hl_dram_access0;          flag '-r3';
title 'ADS_110'

    hl_dram_access_0            device 'p20r4';

    CLK,OE                      pin 1,13;
"inputs
    RW_                          pin 2;
    !TBST                       pin 3;
    !TEA                        pin 4;
    !TS                         pin 5;
    !PTA                        pin 6;

    !C0                         pin 8;          "Counter State Machine
    !C1                         pin 9;          "
    !C2                         pin 10;         "
    !C3                         pin 11;         "
    !C4                         pin 14;         "
    Clock                       pin 23;

"outputs
    "Registered
    !TS_latched                 pin 17;
    !TA                        pin 18;
    !rTS                       pin 19;
    !CAS_dram_en               pin 20;

    "Combinatorial
    !PPTA_dram                 pin 15;
    row_column                 pin 16;

    L,H,x,p,c,k,z             =      0,1,.x.,.p.,.c.,.k.,.z.;
    s_count                    =      [C4,C3,C2,C1,C0];
    clk                        =      CLK;

-----
" Counter states for DRAM accesses
-----
"
"
"
S0count = (s_count == ^b00000);s0count = ^b00000; "00 1F
S1count = (s_count == ^b00100);s1count = ^b00100; "04 1B
S2count = (s_count == ^b00110);s2count = ^b00110; "06 19
S3count = (s_count == ^b00111);s3count = ^b00111; "07 18

```

S4count	=	(s_count == ^b00101);	s4count	=	^b00101;	"05	1A
S5count	=	(s_count == ^b00001);	s5count	=	^b00001;	"01	1E
S6count	=	(s_count == ^b00011);	s6count	=	^b00011;	"03	1C
S7count	=	(s_count == ^b00010);	s7count	=	^b00010;	"02	1D
S8count	=	(s_count == ^b01010);	s8count	=	^b01010;	"0A	15
S9count	=	(s_count == ^b01000);	s9count	=	^b01000;	"08	17
S10count	=	(s_count == ^b01001);	s10count	=	^b01001;	"09	16
S11count	=	(s_count == ^b01011);	s11count	=	^b01011;	"0B	14
S12count	=	(s_count == ^b01111);	s12count	=	^b01111;	"0F	10
S13count	=	(s_count == ^b01110);	s13count	=	^b01110;	"0E	11
S14count	=	(s_count == ^b01100);	s14count	=	^b01100;	"0C	13
S15count	=	(s_count == ^b01101);	s15count	=	^b01101;	"0D	12
S16count	=	(s_count == ^b11101);	s16count	=	^b11101;	"1D	02
S17count	=	(s_count == ^b11100);	s17count	=	^b11100;	"1C	03
S18count	=	(s_count == ^b11000);	s18count	=	^b11000;	"18	07
S19count	=	(s_count == ^b11010);	s19count	=	^b11010;	"1A	05
S20count	=	(s_count == ^b11110);	s20count	=	^b11110;	"1E	01
S21count	=	(s_count == ^b11111);	s21count	=	^b11111;	"1F	00
S22count	=	(s_count == ^b11011);	s22count	=	^b11011;	"1B	04
S23count	=	(s_count == ^b11001);	s23count	=	^b11001;	"19	06
S24count	=	(s_count == ^b10001);	s24count	=	^b10001;	"11	0E
S25count	=	(s_count == ^b10101);	s25count	=	^b10101;	"15	0A
S26count	=	(s_count == ^b10111);	s26count	=	^b10111;	"17	08
S27count	=	(s_count == ^b10011);	s27count	=	^b10011;	"13	0C
S28count	=	(s_count == ^b10010);	s28count	=	^b10010;	"12	0D
S29count	=	(s_count == ^b10110);	s29count	=	^b10110;	"16	09
S30count	=	(s_count == ^b10100);	s30count	=	^b10100;	"14	0B
S31count	=	(s_count == ^b10000);	s31count	=	^b10000;	"10	0F

equations

```

CAS_dram_en := (S2count # S3count # S26count) & !TBST # "single beat
              (S2count # S3count # S5count # S6count #
              S8count # S9count # S11count #
              S12count # S26count) & TBST; "burst
              " Note: CAS_dram_en must come out two(2) clocks early
              " in order for CASen to assert at correct time

PPTA_dram = S4count & !TBST & RW_ # "single beat read
            (S2count & !TBST & !RW_) # "single beat write
            ((S4count # S7count # S10count # S13count) &
            TBST & RW_) # "burst read
            ((S2count # S5count # S8count # S11count) &
            TBST & !RW_); "burst write

```

"Note: TA is a registered PTA signal and therefore does NOT have to be generated
 " PPTA is generated 1 clock in front of PTA

```
TS_latched      := TS #
                  TS_latched & !(PTA # TEA);
                  "latch on TS
                  "hold until PTA or TEA is signalled

TA              := PTA;
                  "will provide a TA signal
                  " one clock after PTA

rTS            := TS;
                  "will provide a TS signal
                  " 3/6.5ns one clock after TS

row_column     =  S0count # S1count #
                  (S2count & Clock & row_column) #
                  S29count;
```

```
test_vectors
  ([clk,!TS] -> [!rTS])
```

```
-----
[ c,  x] -> [x];
[ c,  0] -> [0];
[ c,  0] -> [0];
[ c,  1] -> [1];
[ c,  0] -> [0];
[ c,  0] -> [0];
```

```
end      h1_dram_access0;
```

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

```

module h1_dram_access1;          flag '-r3';
title 'ADS_110'

    h1_dram_access_1          device 'p20r8';      "PAL changed to r8 to support slower speed dram
                                " refresh capability by expanding the refresh state
                                " machine;

```

```

"inputs
CLK,OE          pin 1,13;
!TS             pin 5;
!CAS_dram_en   pin 6;
!ref_timeout    pin 7;
!C0            pin 8;
!C1            pin 9;
!C2            pin 10;
!C3            pin 11;
!C4            pin 14;

```

"Counter State Machine

```

"outputs
"Registered
!R0            pin 16;
!R1            pin 17;
!R2            pin 18;
!RAS_refresh_en pin 19;
!CAS_en        pin 20;
!refresh_req   pin 21;

```

"Combinatorial

```

L,H,x,p,c,k,z = 0,1,.x,..p,..c,..k,..z.;
s_count        = [C4,C3,C2,C1,C0];
r_count        = [R2,R1,R0];
clk            = CLK;

```

" Counter states for DRAM accesses

```

S0count = (s_count == ^b00000);s0count = ^b00000; "00
S1count = (s_count == ^b00100);s1count = ^b00100; "04
S2count = (s_count == ^b00110);s2count = ^b00110; "06
S3count = (s_count == ^b00111);s3count = ^b00111; "07
S4count = (s_count == ^b00101);s4count = ^b00101; "05
S5count = (s_count == ^b00001);s5count = ^b00001; "01

```

```

S6count    = (s_count == ^b00011);s6count    = ^b00011;    "03
S7count    = (s_count == ^b00010);s7count    = ^b00010;    "02
S8count    = (s_count == ^b01010);s8count    = ^b01010;    "0A
S9count    = (s_count == ^b01000);s9count    = ^b01000;    "08
S10count   = (s_count == ^b01001);s10count   = ^b01001;    "09
S11count   = (s_count == ^b01011);s11count   = ^b01011;    "0B
S12count   = (s_count == ^b01111);s12count   = ^b01111;    "0F
S13count   = (s_count == ^b01110);s13count   = ^b01110;    "0E
S14count   = (s_count == ^b01100);s14count   = ^b01100;    "0C
S15count   = (s_count == ^b01101);s15count   = ^b01101;    "0D
S16count   = (s_count == ^b11101);s16count   = ^b11101;    "1D
S17count   = (s_count == ^b11100);s17count   = ^b11100;    "1C
S18count   = (s_count == ^b11000);s18count   = ^b11000;    "18
S19count   = (s_count == ^b11010);s19count   = ^b11010;    "1A
S20count   = (s_count == ^b11110);s20count   = ^b11110;    "1E
S21count   = (s_count == ^b11111);s21count   = ^b11111;    "1F
S22count   = (s_count == ^b11011);s22count   = ^b11011;    "1B
S23count   = (s_count == ^b11001);s23count   = ^b11001;    "19
S24count   = (s_count == ^b10001);s24count   = ^b10001;    "11
S25count   = (s_count == ^b10101);s25count   = ^b10101;    "15
S26count   = (s_count == ^b10111);s26count   = ^b10111;    "17
S27count   = (s_count == ^b10011);s27count   = ^b10011;    "13
S28count   = (s_count == ^b10010);s28count   = ^b10010;    "12
S29count   = (s_count == ^b10110);s29count   = ^b10110;    "16
S30count   = (s_count == ^b10100);s30count   = ^b10100;    "14
S31count   = (s_count == ^b10000);s31count   = ^b10000;    "10

R0count    = (r_count == ^b000);r0count     = ^b000;
R1count    = (r_count == ^b001);r1count     = ^b001;
R2count    = (r_count == ^b010);r2count     = ^b010;
R3count    = (r_count == ^b011);r3count     = ^b011;
R4count    = (r_count == ^b100);r4count     = ^b100;
R5count    = (r_count == ^b101);r5count     = ^b101;
R6count    = (r_count == ^b110);r6count     = ^b110;
R7count    = (r_count == ^b111);r7count     = ^b111;

```

equations

```

refresh_req := ref_timeout & S0count & !TS # "Refresh request signal
             ref_timeout & S29count #
             refresh_req & !R7count;

CAS_en      := CAS_dram_en #
             ((R0count & refresh_req & S0count) #
              R1count # R2count);    "CAS before RAS refresh cycle

RAS_refresh_en := (R1count # R2count # R3count # R4count);    "CAS before RAS refresh cycle

```

"Refresh state machines

state_diagram r_count
 State r0count:
 if(refresh_req & S0count) then r1count
 else r0count;
 State r1count:
 goto r2count;
 State r2count:
 goto r3count;
 State r3count:
 goto r4count;
 State r4count:
 goto r5count;
 State r5count:
 goto r6count;
 State r6count:
 goto r7count;
 State r7count:
 goto r0count;

end h1_dram_access1;

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN


```

module h1_sequence;      flag '-r3';
title  'ADS_110'

    U45_20R6_7          device 'p20r6';

    CLK,OE              pin 1,13;
"inputs
    remap               pin 2;
    !ABB                pin 3;
    !TBST               pin 4;
    RW_                 pin 5;
    !TS                 pin 6;
    !TS_latched         pin 7;
    !refresh_req        pin 8;

    Clock               pin 10;
    A31,A30,A29         pin 11,14,23;

```

```
"outputs
```

```

    "Registered
    !C0                  pin 16;          "Counter State Machine
    !C1                  pin 17;
    !C2                  pin 18;
    !C3                  pin 19;
    !C4                  pin 20;

```

```
"Combinatorial
```

```

    !RAS_dram_en        pin 15;
    !dram_access        pin 22;

```

```

    L,H,x,p,c,k,z      = 0,1,.x,..p,..c,..k,..z.;
    s_count             = [C4,C3,C2,C1,C0];
    e_decode            = [A31,A30,A29];      "memory map decode bits
    clk                 = CLK;

```

```

    c_eprom             = (e_decode == ^b000);  "memory map decoding
    c_adi_port          = (e_decode == ^b100);
    c_counter           = (e_decode == ^b110);
    c_duart             = (e_decode == ^b011);
    c_off_board         = (e_decode == ^b001);
    c_control           = (e_decode == ^b101);
    c_reserved          = (e_decode == ^b010);
    c_dram              = (e_decode == ^b111);

```

```

-----
" Counter states for DRAM accesses

```

()

()

()

```

S0count = (s_count == ^b00000);s0count = ^b00000; "00
S1count = (s_count == ^b00100);s1count = ^b00100; "04
S2count = (s_count == ^b00110);s2count = ^b00110; "06
S3count = (s_count == ^b00111);s3count = ^b00111; "07
S4count = (s_count == ^b00101);s4count = ^b00101; "05
S5count = (s_count == ^b00001);s5count = ^b00001; "01
S6count = (s_count == ^b00011);s6count = ^b00011; "03
S7count = (s_count == ^b00010);s7count = ^b00010; "02
S8count = (s_count == ^b01010);s8count = ^b01010; "0A
S9count = (s_count == ^b01000);s9count = ^b01000; "08
S10count = (s_count == ^b01001);s10count = ^b01001; "09
S11count = (s_count == ^b01011);s11count = ^b01011; "0B
S12count = (s_count == ^b01111);s12count = ^b01111; "0F
S13count = (s_count == ^b01110);s13count = ^b01110; "0E
S14count = (s_count == ^b01100);s14count = ^b01100; "0C
S15count = (s_count == ^b01101);s15count = ^b01101; "0D
S16count = (s_count == ^b11101);s16count = ^b11101; "1D
S17count = (s_count == ^b11100);s17count = ^b11100; "1C
S18count = (s_count == ^b11000);s18count = ^b11000; "18
S19count = (s_count == ^b11010);s19count = ^b11010; "1A
S20count = (s_count == ^b11110);s20count = ^b11110; "1E
S21count = (s_count == ^b11111);s21count = ^b11111; "1F
S22count = (s_count == ^b11011);s22count = ^b11011; "1B
S23count = (s_count == ^b11001);s23count = ^b11001; "19
S24count = (s_count == ^b10001);s24count = ^b10001; "11
S25count = (s_count == ^b10101);s25count = ^b10101; "15
S26count = (s_count == ^b10111);s26count = ^b10111; "17
S27count = (s_count == ^b10011);s27count = ^b10011; "13
S28count = (s_count == ^b10010);s28count = ^b10010; "12
S29count = (s_count == ^b10110);s29count = ^b10110; "16
S30count = (s_count == ^b10100);s30count = ^b10100; "14
S31count = (s_count == ^b10000);s31count = ^b10000; "10

```

equations

```

dram_access = c_dram # (c_eprom & remap) & ABB; "This signal is valid 22ns into S0
" which gives it 8ns to (S1count & !Clock)
" This is enough time since RAS_dram_en is
" a combinatorial output (ie. no setup)

RAS_dram_en = dram_access & S1count & !Clock # "set here
RAS_dram_en & (!TBST &
(S1count # S2count # S3count # S4count #
S26count # S27count # (S28count & Clock))) # "single beat

```

```

RAS_dram_en & (TBST &
(S1count # S2count # S3count # S4count # S5count #
S6count # S7count # S8count # S9count #
S10count # S11count # S12count # S13count # S26count # S27count #
(S28count & Clock)));
"burst

```

```

"This RAS enable signal or RAS_refresh_en*
" will result in generating RAS*(7:0) to the DRAM

```

```

"state machines
-----

```

```

state_diagram s_count

```

```

State s0count:

```

```

    if((TS # TS_latched) & !refresh_req) then s1count
    else s0count;

```

```

State s1count:

```

```

    if((c_dram # (c_eprom & remap)) & TS_latched) then s2count "added TS_latched qualifier for reduction
    else if (TS_latched) then s1count "hold here until this cycle completes
    else s0count;

```

```

" If the dram is chip select (even through the eeprom & remap) then there is no possibility of
" TEA being generated for this cycle so this state machine does not have to stop ever if it reaches s2count!!!!
" There is no need to check for writing to eeprom because if it is remapped then the write can occur into the DRAM

```

```

State s2count:

```

```

    if(!TBST & !RW_) then s26count
    else s3count;

```

```

State s3count:

```

```

    goto s4count;

```

```

State s4count:

```

```

    if(!TBST & RW_) then s28count
    else s5count;

```

```

State s5count:

```

```

    goto s6count;

```

```

State s6count:

```

```

    goto s7count;

```

```

State s7count:

```

```

    goto s8count;

```

```

State s8count:

```

```

    goto s9count;

```

```

State s9count:

```

```

    goto s10count;

```

```

State s10count:

```

```

    goto s11count;

```

```

State s11count:

```

```

    if(!RW_) then s26count
    else s12count;

```

```

"For the short 410 or 110 burst read

```

```

"      goto s12count;           For the long burst
State s12count:
      goto s13count;
State s13count:
      goto s28count;           "For the short 410 or 110 burst write
      goto s14count;
State s14count:
      goto s0count;
"      goto s15count;           For the long burst
State s15count:
      goto s0count;
"      goto s16count;           For the long burst
State s16count:
      goto s0count;
"      goto s17count;           For the long burst
State s17count:
      goto s0count;
"      goto s17count;           For the long burst
State s18count:
      goto s0count;
"      goto s19count;           For the long burst
State s19count:
      goto s0count;
"      goto s20count;           For the long burst
State s20count:
      goto s0count;
"      goto s21count;           For the long burst
State s21count:
      goto s0count;
"      goto s22count;           For the long burst
State s22count:
      goto s0count;
"      goto s23count;           For the long burst
State s23count:
      goto s0count;
"      if(!RW_) then s0count     For the 410 long burst
"      else s24count;
State s24count:
      goto s0count;
"      goto s25count;           For the 410 long burst
State s25count:
      goto s0count;
"      goto s26count;           For the 410 long burst Read it returns here regardless
State s26count:
      goto s27count;
State s27count:
      goto s28count;
State s28count:
      goto s29count;
State s29count:

```

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

```
    goto s0count;  
State s30count:  
    goto s0count;  
State s31count:  
    goto s0count;
```

C-20

```
end h1_sequence;
```

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

```

( )
module h1_peripheral_eprom;    flag '-r3';
title 'ADS_110'

    h1_per_eprom        device 'p20r6';

    CLK,OE              pin 1,13;

"inputs
!TS                    pin 2;
RW_                    pin 3;
!DBB                   pin 4;
!TS_latched            pin 5;
dram_access            pin 6;
A31,A30,A29           pin 7,8,9;
TCOD3                  pin 10;
TCOD2                  pin 11;
TCOD1                  pin 14;
TCOD0                  pin 15;

"outputs
"Registered
!P0                    pin 16;
!P1                    pin 17;
!P2                    pin 18;
!P3                    pin 19;
!P4                    pin 20;
LED                    pin 22;
"Counter State Machine

"Combinatorial
!PPTA_peripheral_0    pin 15;

L,H,x,p,c,k,z        =    0,1,.x...p...c...k...z.;
p_count               =    [P4,P3,P2,P1,P0];
e_decode              =    [A31,A30,A29];
clk                   =    CLK;
"memory map decode bits

c_eprom               =    (e_decode == ^b000);
c_adi_port            =    (e_decode == ^b100);
c_counter             =    (e_decode == ^b010);
c_duart               =    (e_decode == ^b011);
c_off_board           =    (e_decode == ^b001);
c_control             =    (e_decode == ^b101);
c_reserved            =    (e_decode == ^b110);
c_dram                =    (e_decode == ^b111);
"memory map decoding

```

```

-----
" Counter states for DRAM accesses
-----

```

```

P0count    = (p_count == ^b00000); p0count    = ^b00000;
P1count    = (p_count == ^b00100); p1count    = ^b00100;
P2count    = (p_count == ^b00101); p2count    = ^b00101;
P3count    = (p_count == ^b00111); p3count    = ^b00111;
P4count    = (p_count == ^b00110); p4count    = ^b00110;
P5count    = (p_count == ^b00010); p5count    = ^b00010;
P6count    = (p_count == ^b00011); p6count    = ^b00011;
P7count    = (p_count == ^b00001); p7count    = ^b00001;
P8count    = (p_count == ^b10001); p8count    = ^b10001;
P9count    = (p_count == ^b10011); p9count    = ^b10011;
P10count   = (p_count == ^b10010); p10count   = ^b10010;
P11count   = (p_count == ^b10000); p11count   = ^b10000;
P12count   = (p_count == ^b10100); p12count   = ^b10100;
P13count   = (p_count == ^b10101); p13count   = ^b10101;
P14count   = (p_count == ^b10111); p14count   = ^b10111;
P15count   = (p_count == ^b10110); p15count   = ^b10110;
P16count   = (p_count == ^b11110); p16count   = ^b11110;
P17count   = (p_count == ^b11111); p17count   = ^b11111;
P18count   = (p_count == ^b11101); p18count   = ^b11101;
P19count   = (p_count == ^b11100); p19count   = ^b11100;
P20count   = (p_count == ^b11000); p20count   = ^b11000;
P21count   = (p_count == ^b11001); p21count   = ^b11001;
P22count   = (p_count == ^b11011); p22count   = ^b11011;
P23count   = (p_count == ^b11010); p23count   = ^b11010;
P24count   = (p_count == ^b01010); p24count   = ^b01010;
P25count   = (p_count == ^b01011); p25count   = ^b01011;
P26count   = (p_count == ^b01001); p26count   = ^b01001;
P27count   = (p_count == ^b01000); p27count   = ^b01000;
P28count   = (p_count == ^b01100); p28count   = ^b01100;
P29count   = (p_count == ^b01101); p29count   = ^b01101;
P30count   = (p_count == ^b01111); p30count   = ^b01111;
P31count   = (p_count == ^b01110); p31count   = ^b01110;

```

equations

```

PPTA_peripheral_0 = P9count;      "PPTA is generated 1 clock before PTA which
                                   " is latched to generate TA* to the system bus

```

```

LED = TCOD3 & !TCOD1 & TCOD0;    "The LED decodes for Instruction Fetches both U/S"

```

"state machines

```

-----
state_diagram p_count
  State p0count:

```

()

()

()

```

    if(TS) then p1count
    else if(TS_latched) then p2count
    else p0count;
State p1count:
    goto p2count;
State p2count:
    if(c_eprom & DBB & RW_ & !dram_access) then p3count
    else if(TS_latched) then p2count
    else p0count;
State p3count:
    goto p4count;
State p4count:
    goto p5count;
State p5count:
    goto p6count;
State p6count:
    goto p7count;
State p7count:
    goto p8count;
State p8count:
    goto p9count;
State p9count:
    "EPROM_PPTA* generated
    goto p10count;
State p10count:
    goto p11count;
State p11count:
    goto p12count;
State p12count:
    "burst into EPROM will have DBB still asserted here
    if(DBB) then p2count
    else p18count;
State p13count:
    goto p0count;
State p14count:
    goto p0count;
State p15count:
    goto p0count;
State p16count:
    goto p0count;
State p17count:
    goto p0count;
State p18count:
    "begin ODT
    goto p19count;
State p19count:
    goto p20count;
State p20count:
    goto p21count;
State p21count:
    goto p22count;
State p22count:
    goto p23count;

```

"Note: This state machine stops if TEA is being generated for an illegal EPROM access"


```
State p23count:           "end ODT
  goto p0count;
State p24count:
  goto p0count;
State p25count:
  goto p0count;
State p26count:
  goto p0count;
State p27count:
  goto p0count;
State p28count:
  goto p0count;
State p29count:
  goto p0count;
State p30count:
  goto p0count;
State p31count:
  goto p0count;
```

```
end hl_peripheral_eprom;
```

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

```

module h1_peripheral; flag '-r3';
title 'ADS_110'

    h1_per_ext          device 'p20r8';

    CLK,OE              pin 1,13;

"inputs
!TS                   pin 3;
!TS_latched           pin 5;
RW_                   pin 7;
    A31,A30,A29       pin 8,9,10;
counter_3             pin 11;
counter_4             pin 14;

"
!DBB                  pin 4;          added for duart_CS* generation
"!ext_ODT             pin 2;

"outputs
    "Registered
!P0                   pin 15;          "Counter State Machine
!P1                   pin 16;          "
!P2                   pin 17;          "
!P3                   pin 18;          "
!P4                   pin 19;          "
!ODT                  pin 20;
!PPTA_peripheral_1   pin 21;
!duart_CS             pin 22;

L,H,x,p,c,k,z        =    0,1,.x,.p,.c,.k,.z.;
p_count              =    [P4,P3,P2,P1,P0];
e_decode             =    [A31,A30,A29];    "memory map decode bits
clk                  =    CLK;

c_eprom              =    (e_decode == ^b000);    "memory map decoding
c_adi_port           =    (e_decode == ^b100);
c_counter            =    (e_decode == ^b010);
c_duart              =    (e_decode == ^b011);
c_off_board          =    (e_decode == ^b001);
c_control            =    (e_decode == ^b101);
c_reserved           =    (e_decode == ^b110);
c_dram               =    (e_decode == ^b111);

```

```

-----
" Counter states for DRAM accesses

```

```

P0count    = (p_count == ^b00000);p0count    = ^b00000;
P1count    = (p_count == ^b00100);p1count    = ^b00100;
P2count    = (p_count == ^b00110);p2count    = ^b00110;
P3count    = (p_count == ^b00010);p3count    = ^b00010;
P4count    = (p_count == ^b00011);p4count    = ^b00011;
P5count    = (p_count == ^b00001);p5count    = ^b00001;
P6count    = (p_count == ^b00101);p6count    = ^b00101;
P7count    = (p_count == ^b00111);p7count    = ^b00111;
P8count    = (p_count == ^b10111);p8count    = ^b10111;
P9count    = (p_count == ^b10110);p9count    = ^b10110;
P10count   = (p_count == ^b10100);p10count   = ^b10100;
P11count   = (p_count == ^b10101);p11count   = ^b10101;
P12count   = (p_count == ^b10001);p12count   = ^b10001;
P13count   = (p_count == ^b10011);p13count   = ^b10011;
P14count   = (p_count == ^b10010);p14count   = ^b10010;
P15count   = (p_count == ^b10000);p15count   = ^b10000;
P16count   = (p_count == ^b11000);p16count   = ^b11000;
P17count   = (p_count == ^b11001);p17count   = ^b11001;
P18count   = (p_count == ^b11011);p18count   = ^b11011;
P19count   = (p_count == ^b11010);p19count   = ^b11010;
P20count   = (p_count == ^b11110);p20count   = ^b11110;
P21count   = (p_count == ^b11100);p21count   = ^b11100;
P22count   = (p_count == ^b11101);p22count   = ^b11101;
P23count   = (p_count == ^b11111);p23count   = ^b11111;
P24count   = (p_count == ^b01111);p24count   = ^b01111;
P25count   = (p_count == ^b01110);p25count   = ^b01110;
P26count   = (p_count == ^b01100);p26count   = ^b01100;
P27count   = (p_count == ^b01101);p27count   = ^b01101;
P28count   = (p_count == ^b01001);p28count   = ^b01001;
P29count   = (p_count == ^b01011);p29count   = ^b01011;
P30count   = (p_count == ^b01010);p30count   = ^b01010;
P31count   = (p_count == ^b01000);p31count   = ^b01000;

```

equations

```

ODT                := (P13count # P14count # P15count #
P16count # P17count) # "DUART's outputs disabling
!(counter_3 & counter_4); "EPROM output disabling

PPTA_peripheral_1  := P3count & c_adi_port #
P10count & c_duart & RW_ #
P11count& c_duart & !RW_;

duart_CS           := c_duart & (P2count # P3count # P4count #

```

P5count # P6count # P7count # P8count #
P9count # P10count # P11count # P12count);

"state machines

state_diagram p_count

```
State p0count:
    if(TS # TS_latched) then p1count
    else p0count;
State p1count:
    if((c_duart # c_adi_port) & TS_latched) then p2count
    else if(TS_latched) then p1count
    else p0count;
State p2count:
    goto p3count;
State p3count:
    goto p4count;
State p4count:
    goto p5count;
State p5count:
    if(!c_duart) then p0count           "TA* for ADI/Counter/Control here
    else p6count;
State p6count:
    goto p7count;
State p7count:
    goto p8count;
State p8count:
    goto p9count;
State p9count:
    goto p10count;
State p10count:
    goto p11count;
State p11count:
    goto p12count;           "TA* for duart(write) here

State p12count:
    if(!RW_) then p20count       "TA* for duart(read) here
                                "Then we don't have to wait for output disabling
                                " We just have to meet the CS* high (90ns) requirement
                                "Note: Since RW_ is only valid 2 clocks after TA then
                                "      a decision must be made when RW_ is still valid!

    else p13count;
State p13count:
    goto p14count;           "begin ODT for read cycles only
State p14count:
    goto p15count;
State p15count:
    goto p16count;
State p16count:
```

```
    goto p17count;
State p17count:
    goto p0count;
State p18count:
    goto p0count;
State p19count:
    goto p0count;
State p20count:
    goto p21count;
State p21count:
    goto p22count;
State p22count:
    goto p0count;
State p23count:
    goto p0count;
State p24count:
    goto p0count;
State p25count:
    goto p0count;
State p26count:
    goto p0count;
State p27count:
    goto p0count;
State p28count:
    goto p0count;
State p29count:
    goto p0count;
State p30count:
    goto p0count;
State p31count:
    goto p0count;

    "end ODT for read cyles

    "Read cycle duart_CS* high cycles
    "
    "

end h1_peripheral;
```

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

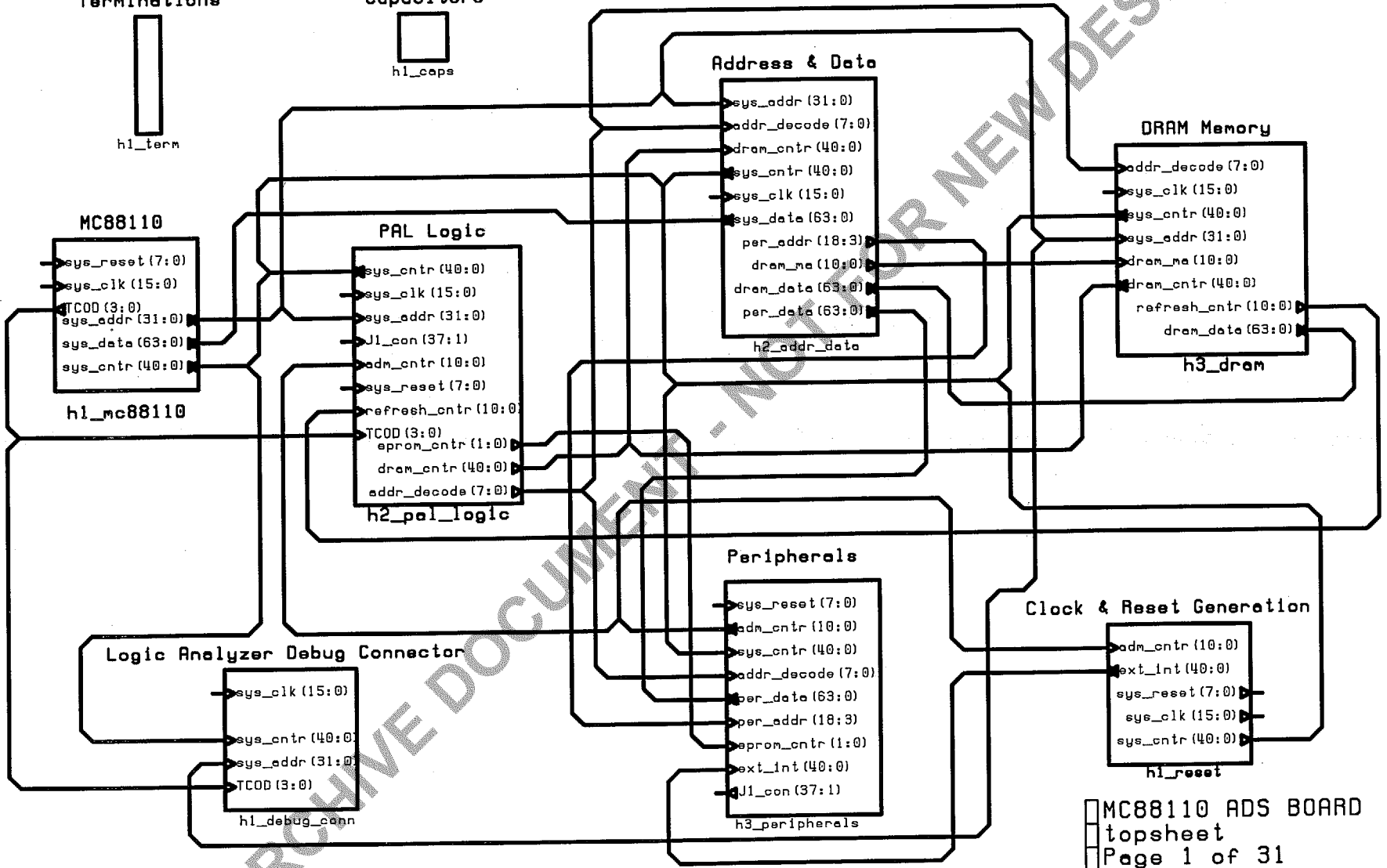
**Appendix D
Schematics**

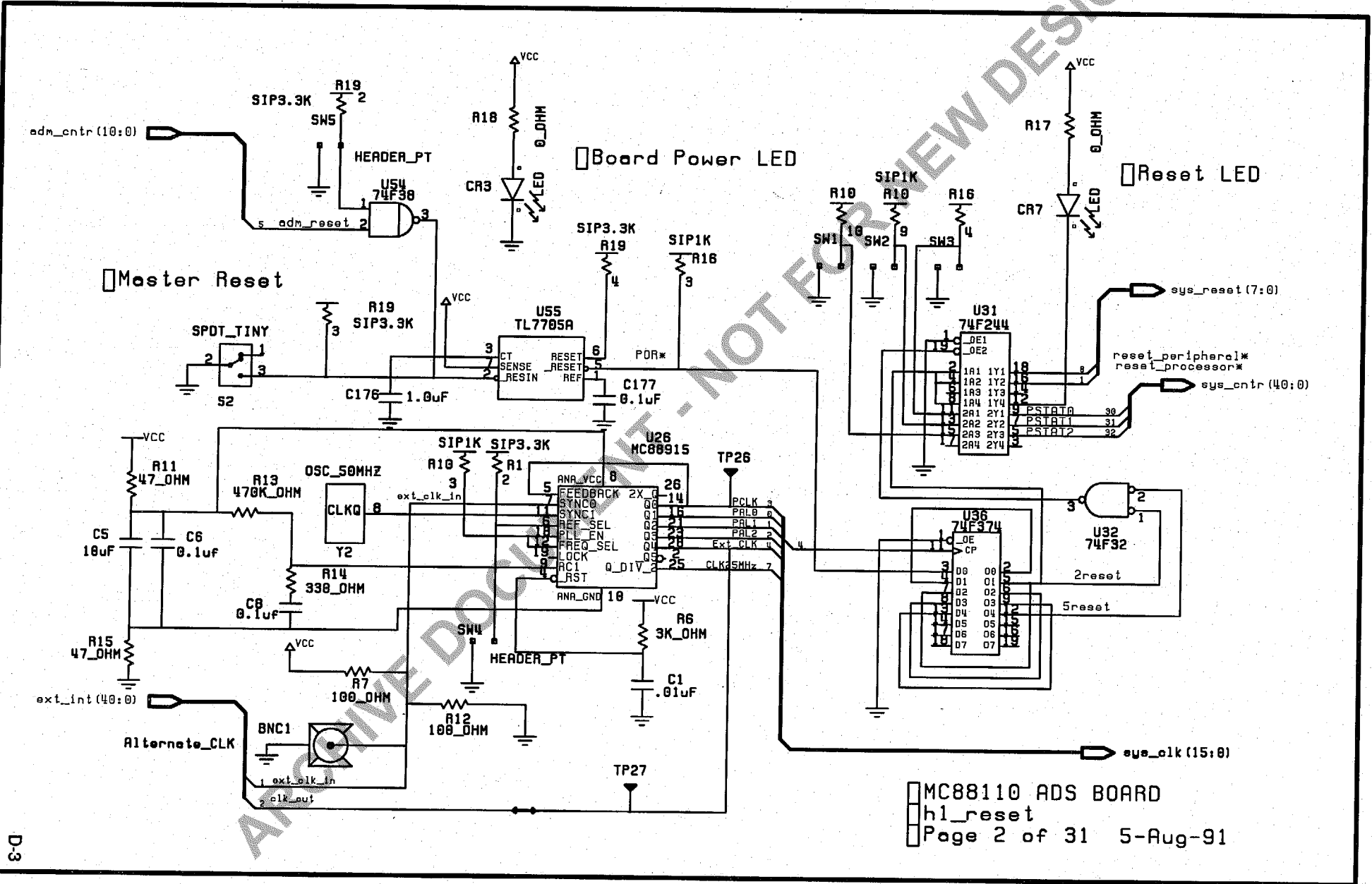
ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

Terminations



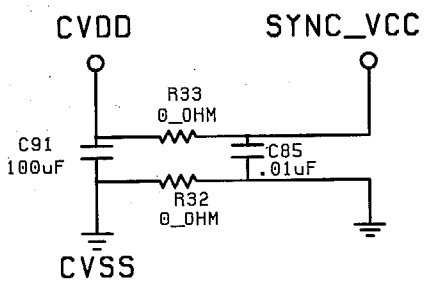
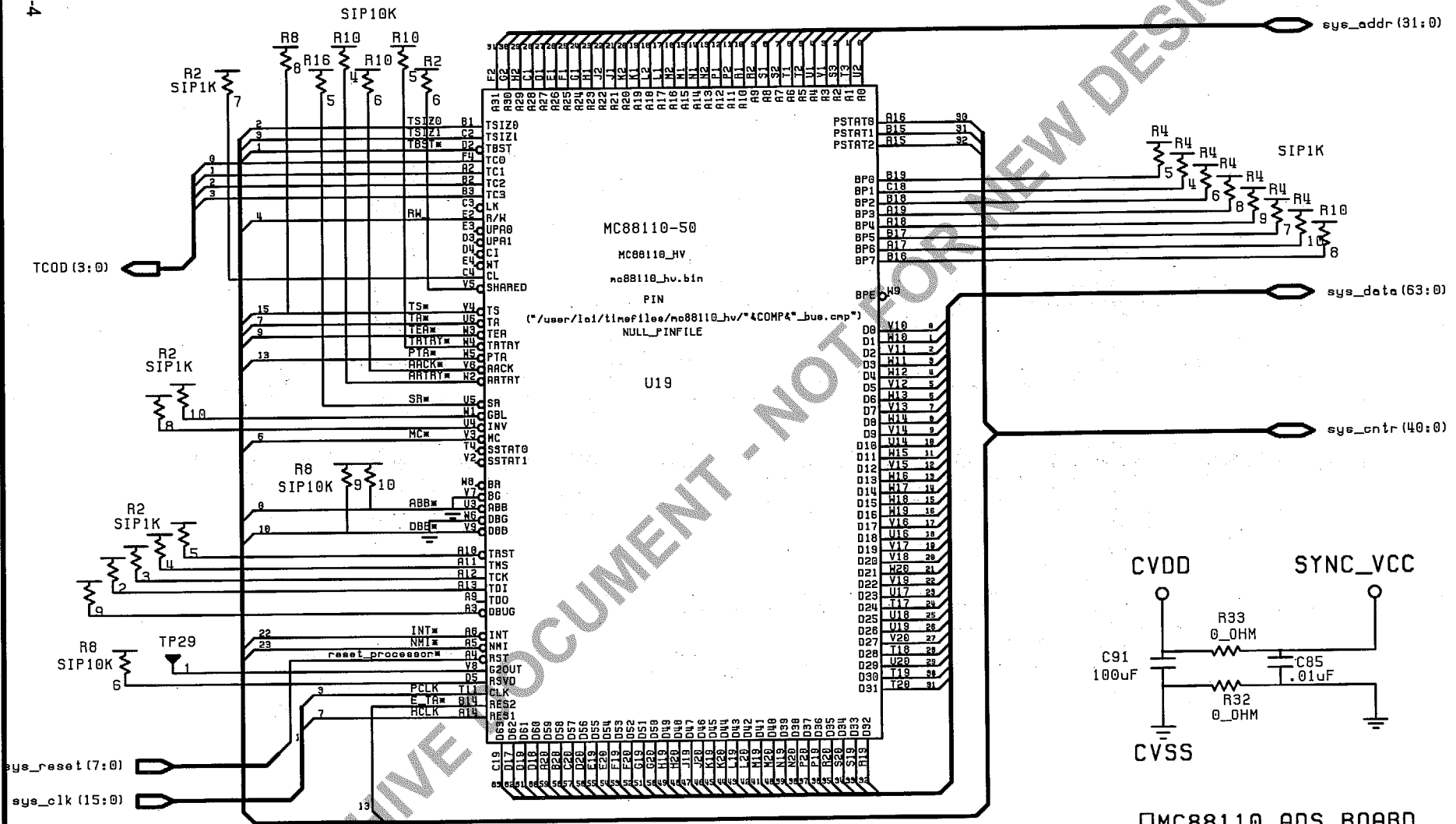
Capacitors





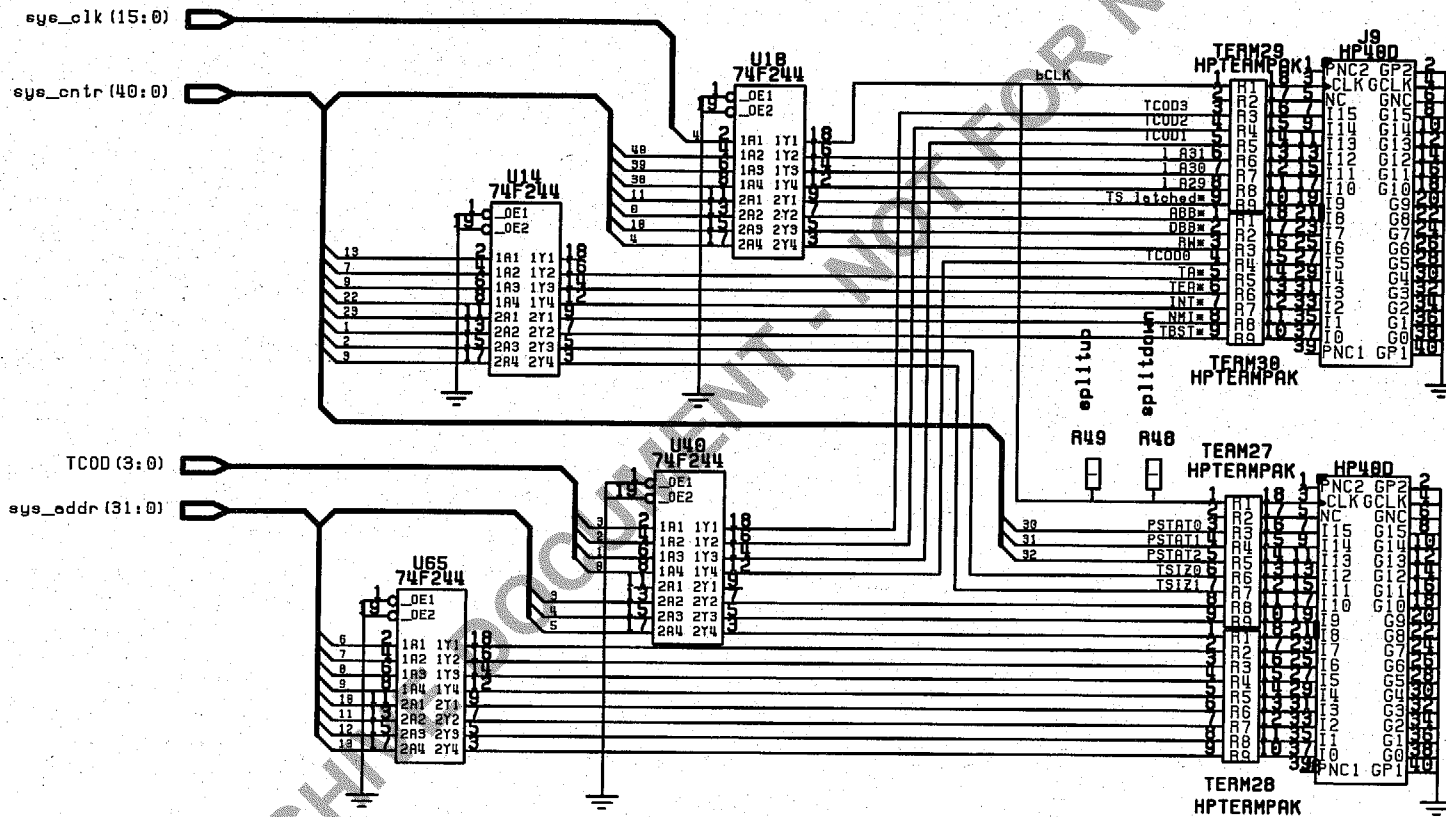
MC88110 ADS BOARD
 h1_reset
 Page 2 of 31 5-Aug-91

D-4



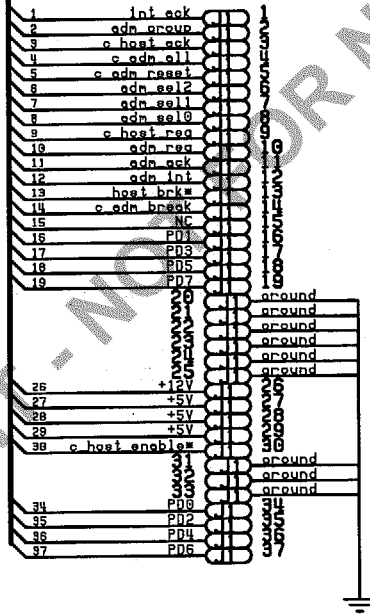
MC88110 ADS BOARD
 h1_mc88110
 Page 3 of 31 5-Aug-91

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN



MC88110 ADS BOARD
 h1_debug_conn
 Page 4 of 31 5-Aug-91

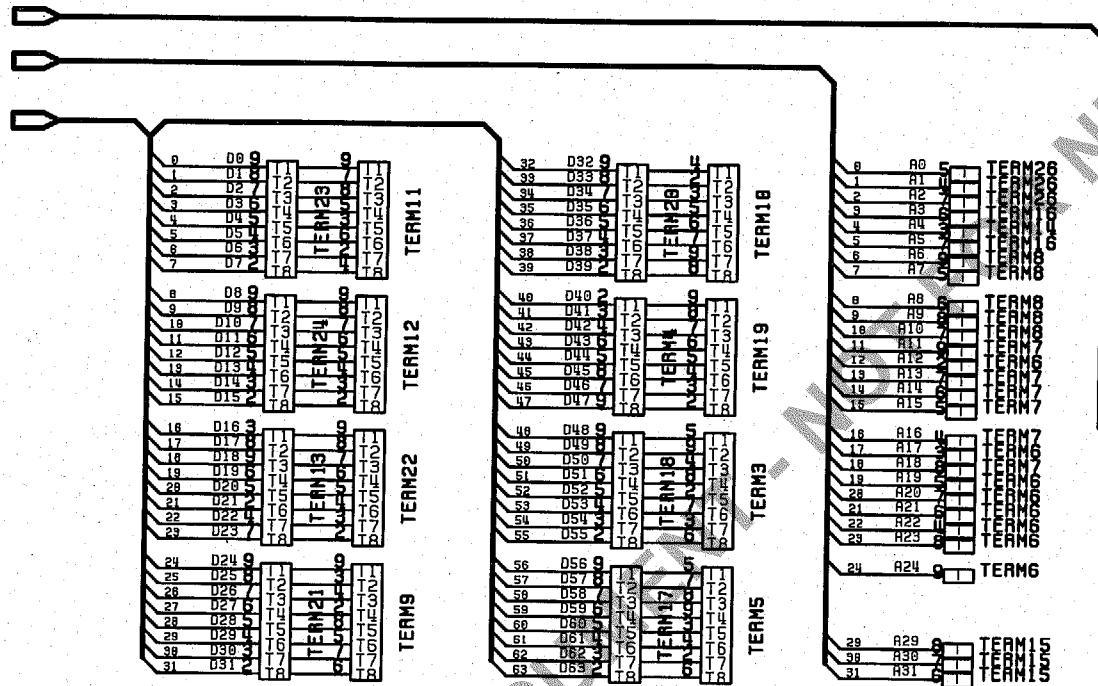
J1_con (37:1)



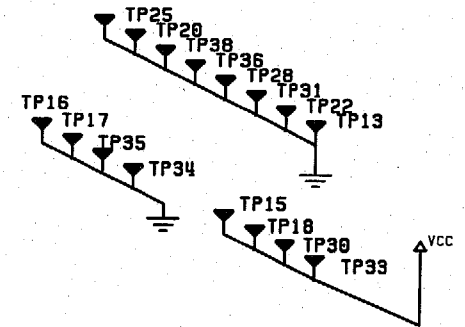
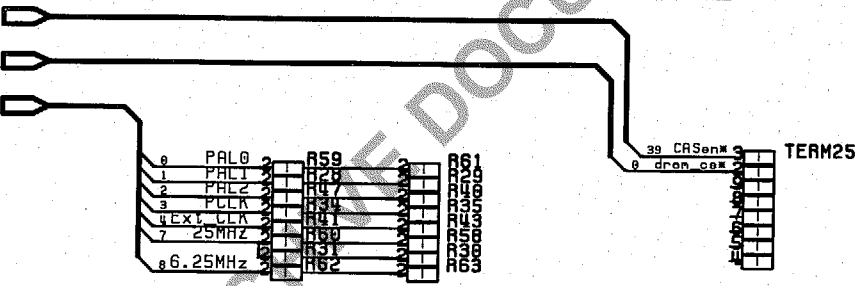
ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

MC88110 ADS BOARD
h1_adi_conn
Page 5 of 31 5-Aug-91

eye_entr (40:0)
 eye_addr (31:0)
 eye_data (63:0)

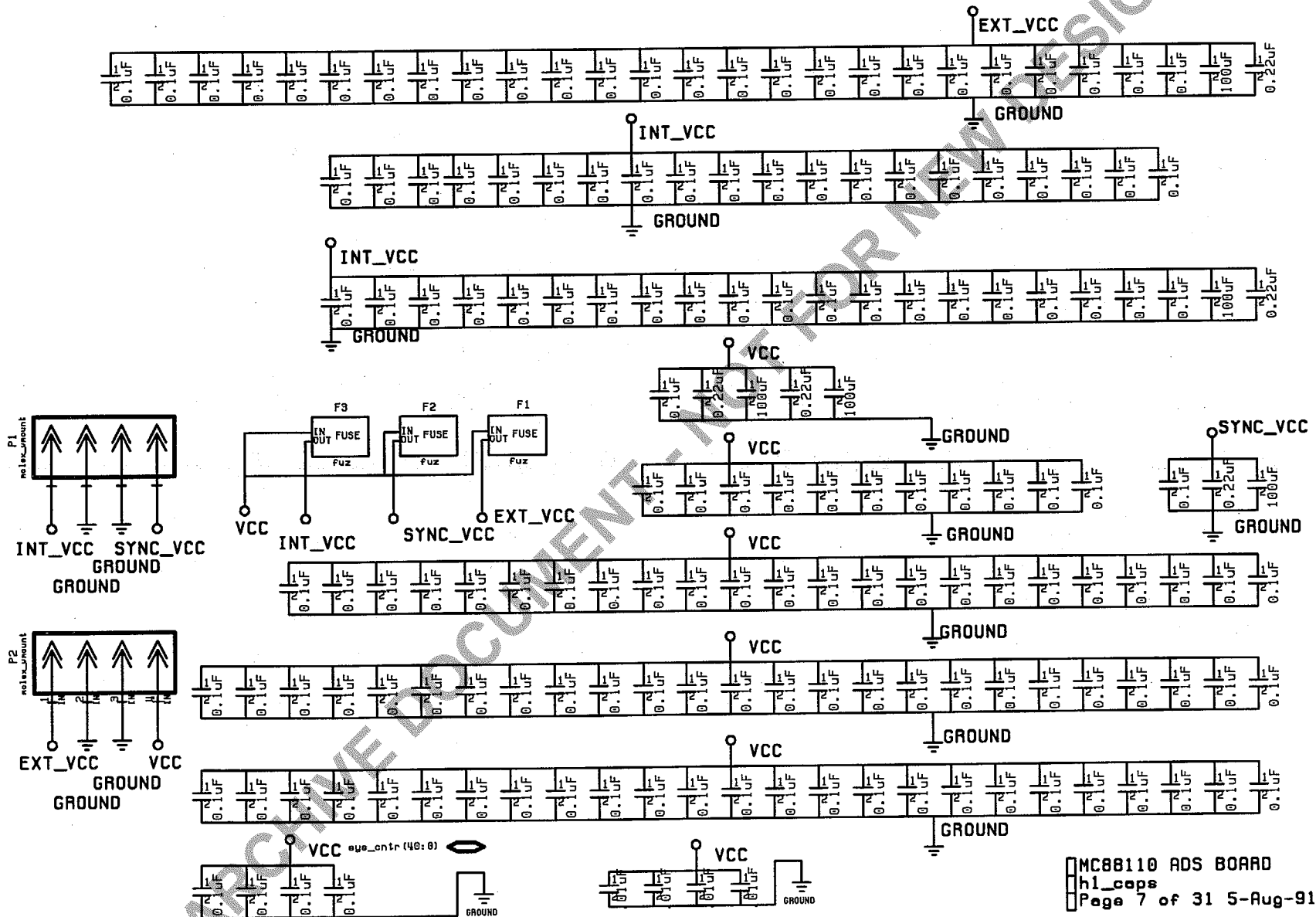


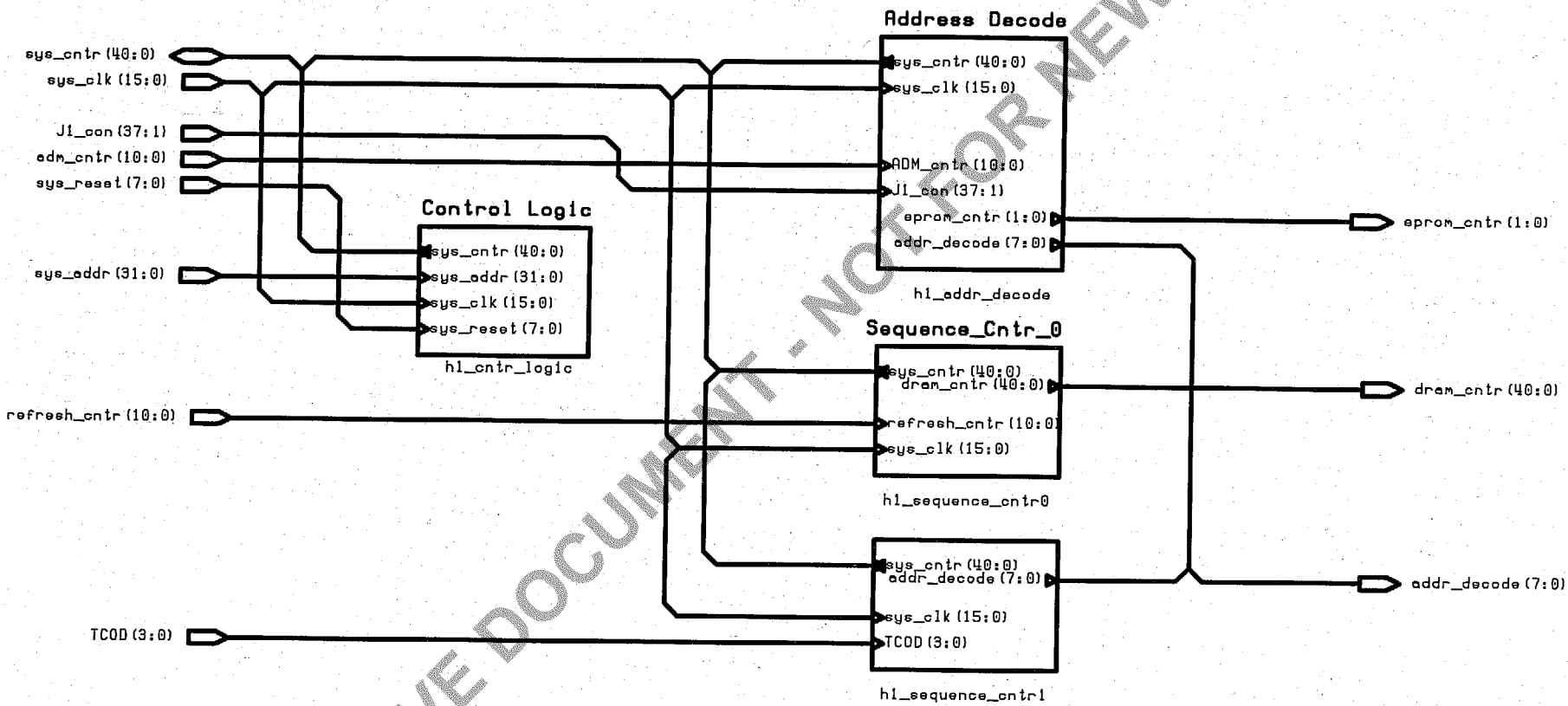
dram_entr (40:0)
 addr_decode (7:0)
 eye_clk (15:0)



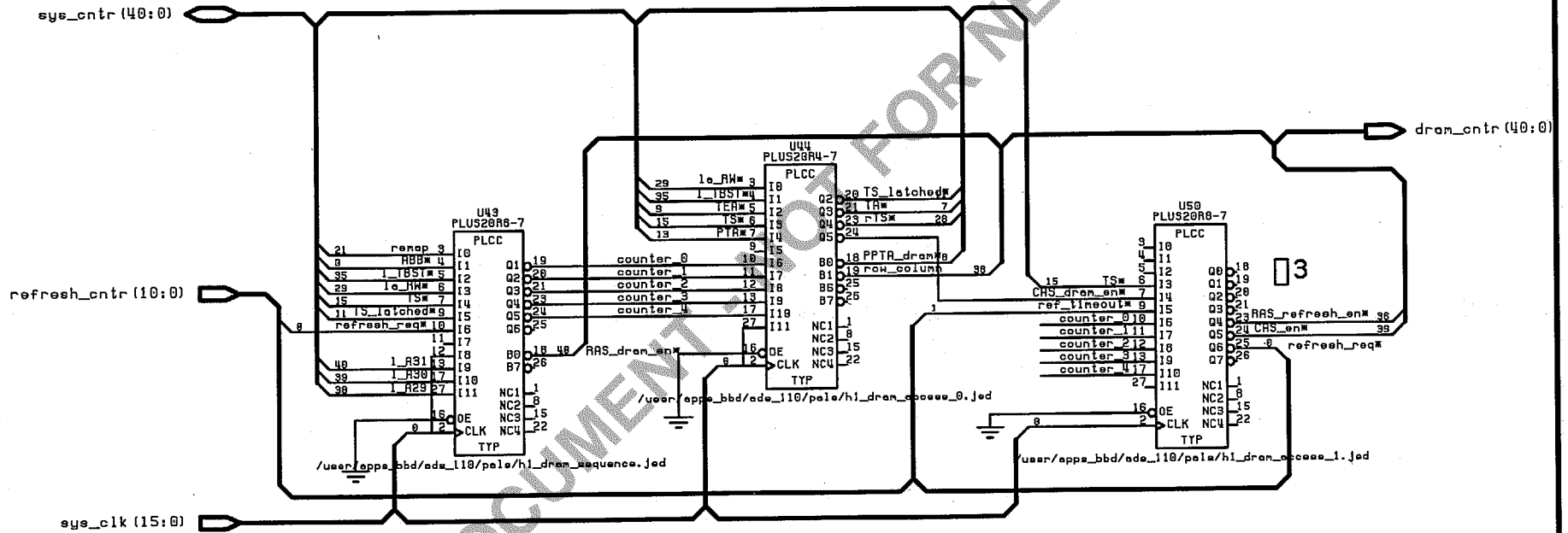
Discrete terminations on clock signals

MC88110 ADS BOARD
 h1_term
 Page 6 of 31 5-Aug-91

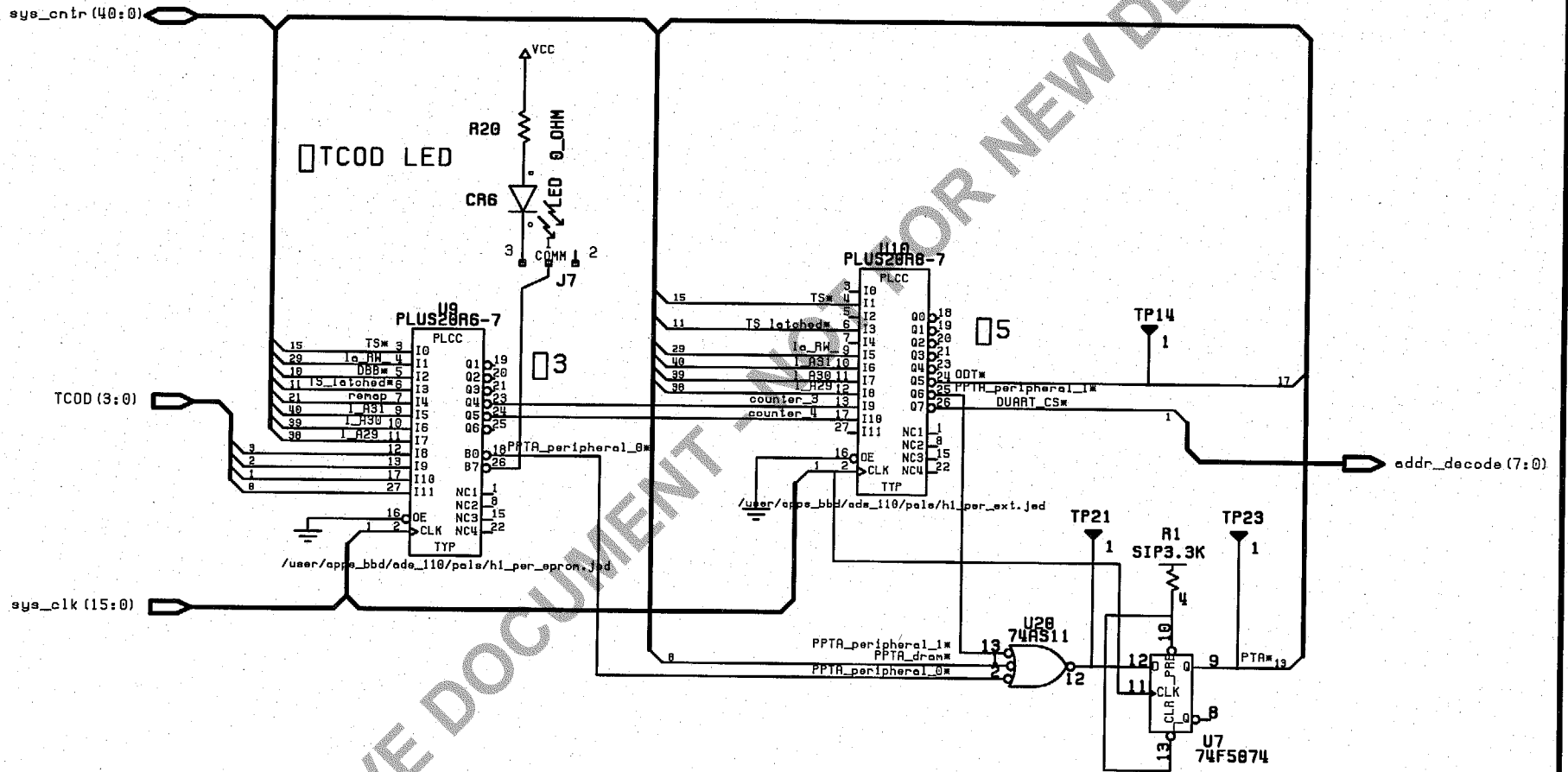




ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

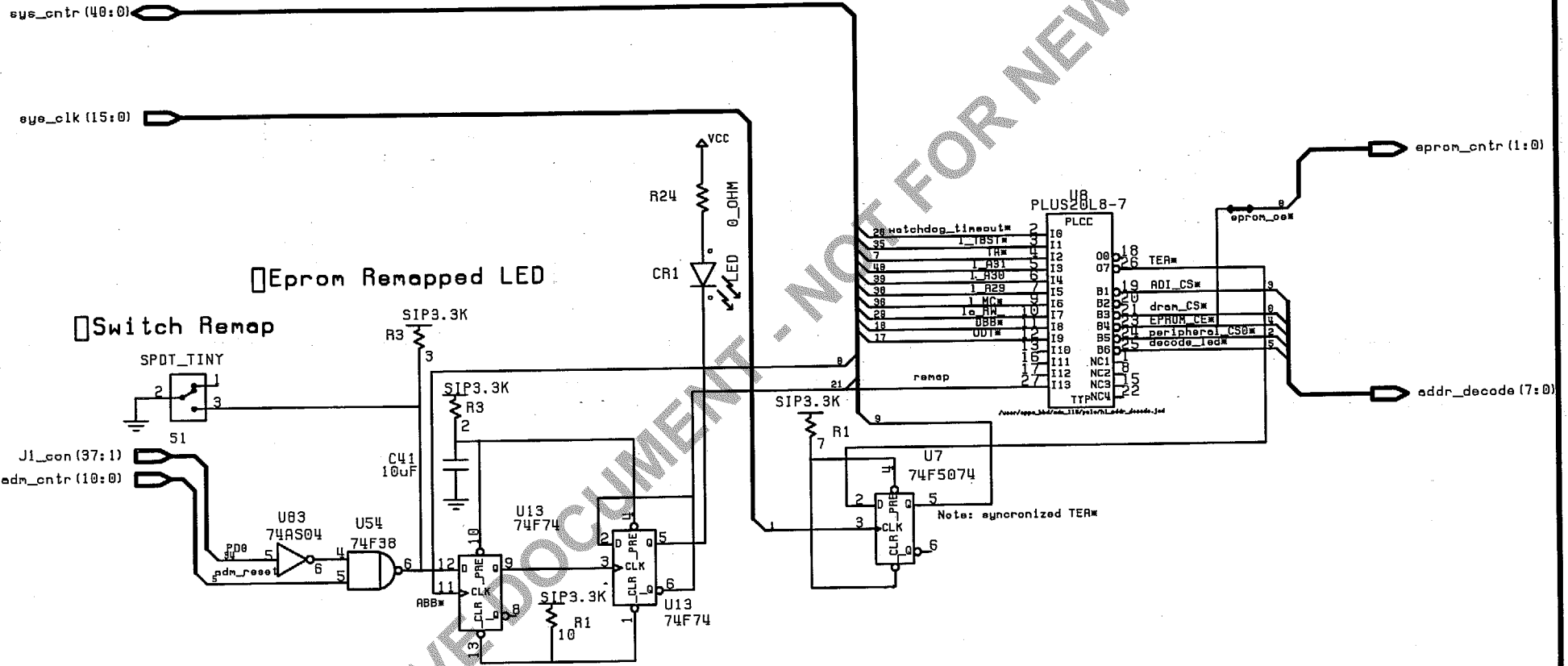


ARCHIVE DOCUMENT FOR NEW DESIGN



ARCHIVE DOCUMENT FOR NEW DESIGN

MC88110 ADS BOARD
 h1_sequence_cntr1
 Page 10 of 31 5-Aug-91



Eprom Remapped LED

Switch Remap

MC88110 ADS BOARD
 h1_addr_decode
 Page 11 of 31 5-Aug-91

ARCHIVED DOCUMENT - NOT FOR NEW DESIGN

sys_cntr (40:0)

sys_addr (31:0)

sys_clk (15:0)

sys_reset (7:0)

U89
74AS94

U83
74AS84

U41
74F373

U2
74F74

R1
SIP3.3K

R1
SIP3.3K

R10
SIP1K

R16
SIP1K

R10
SIP1K

U21
74F579

TP24

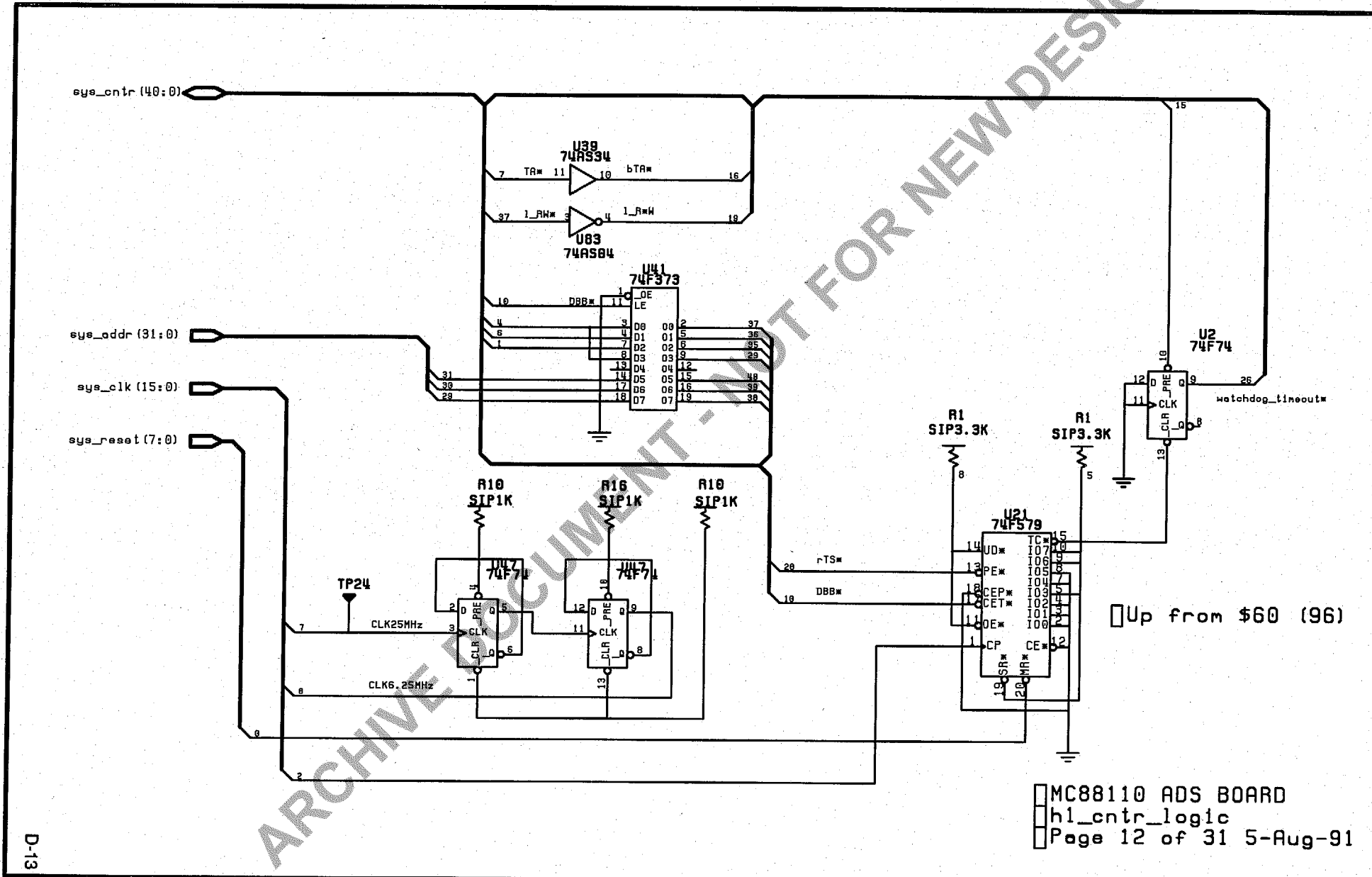
CLK25MHz

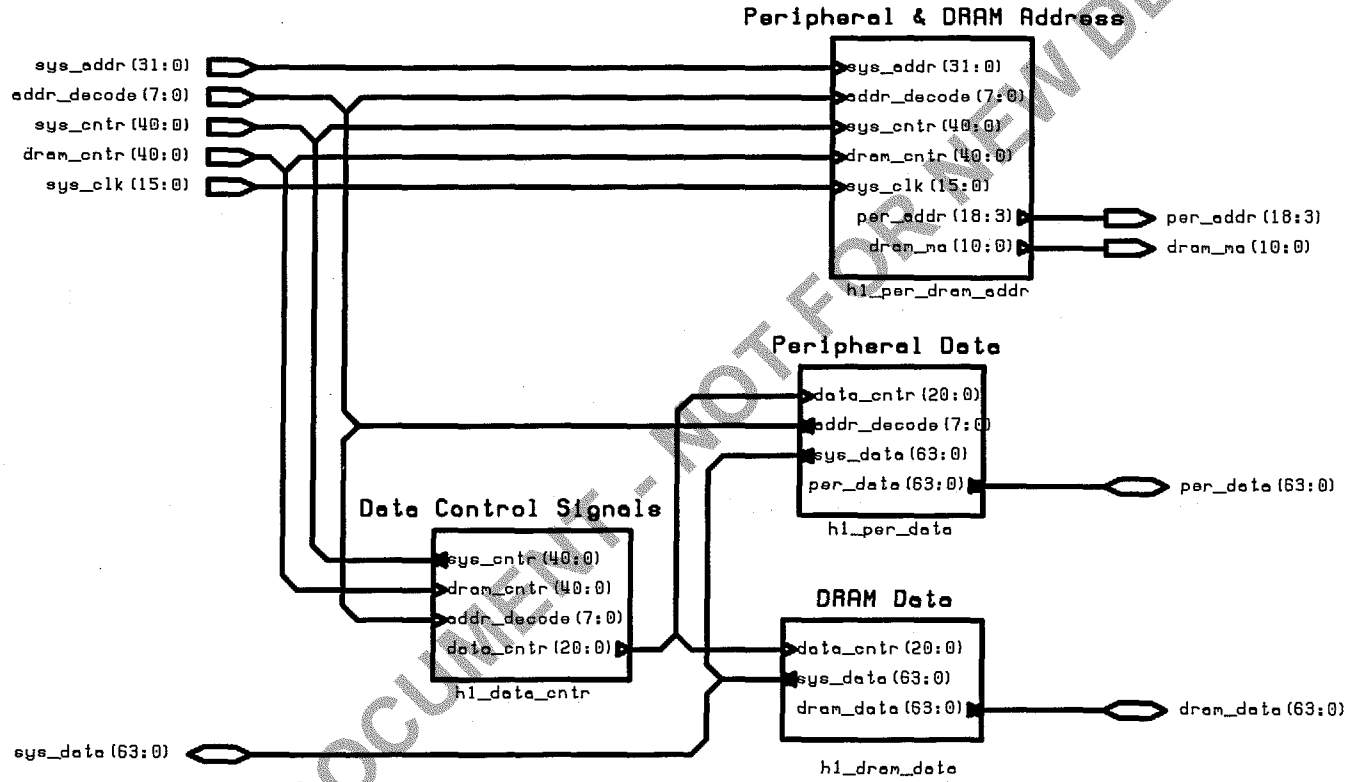
CLK6.25MHz

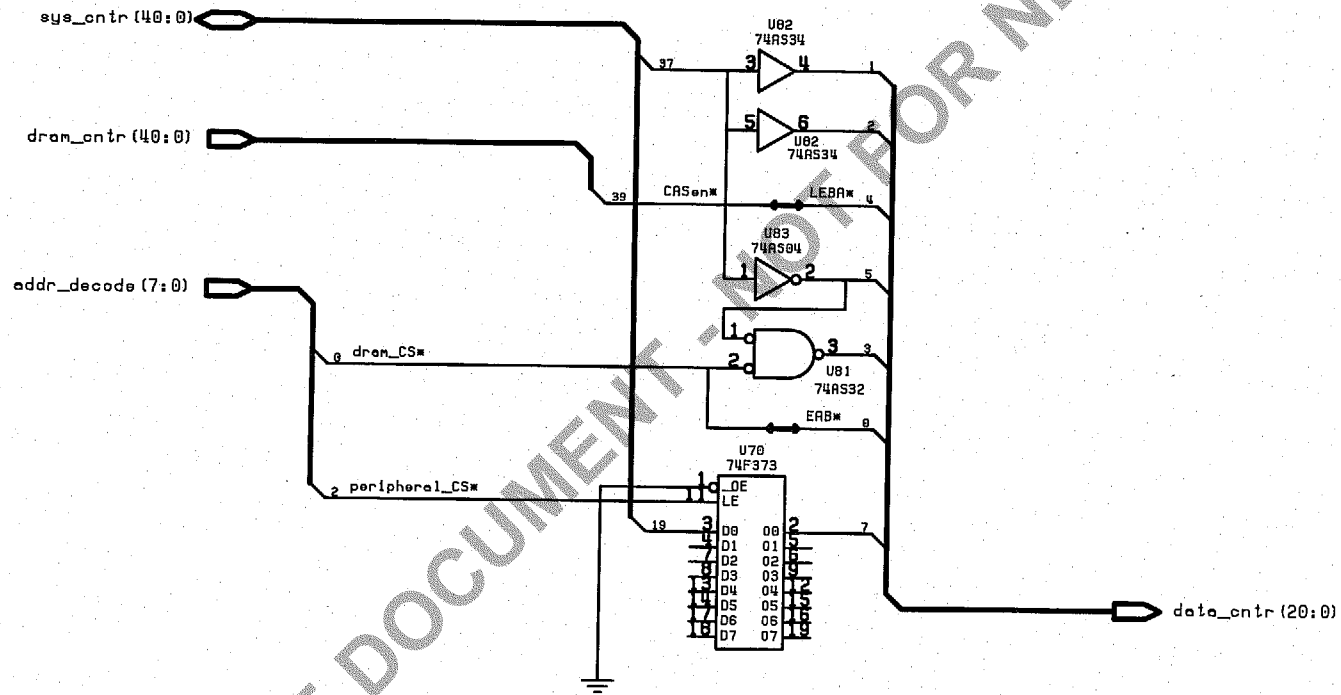
Up from \$60 (96)

MC88110 ADS BOARD
hl_cntr_logic
Page 12 of 31 5-Aug-91

ARCHIVED DOCUMENT - NOT FOR NEW DESIGN







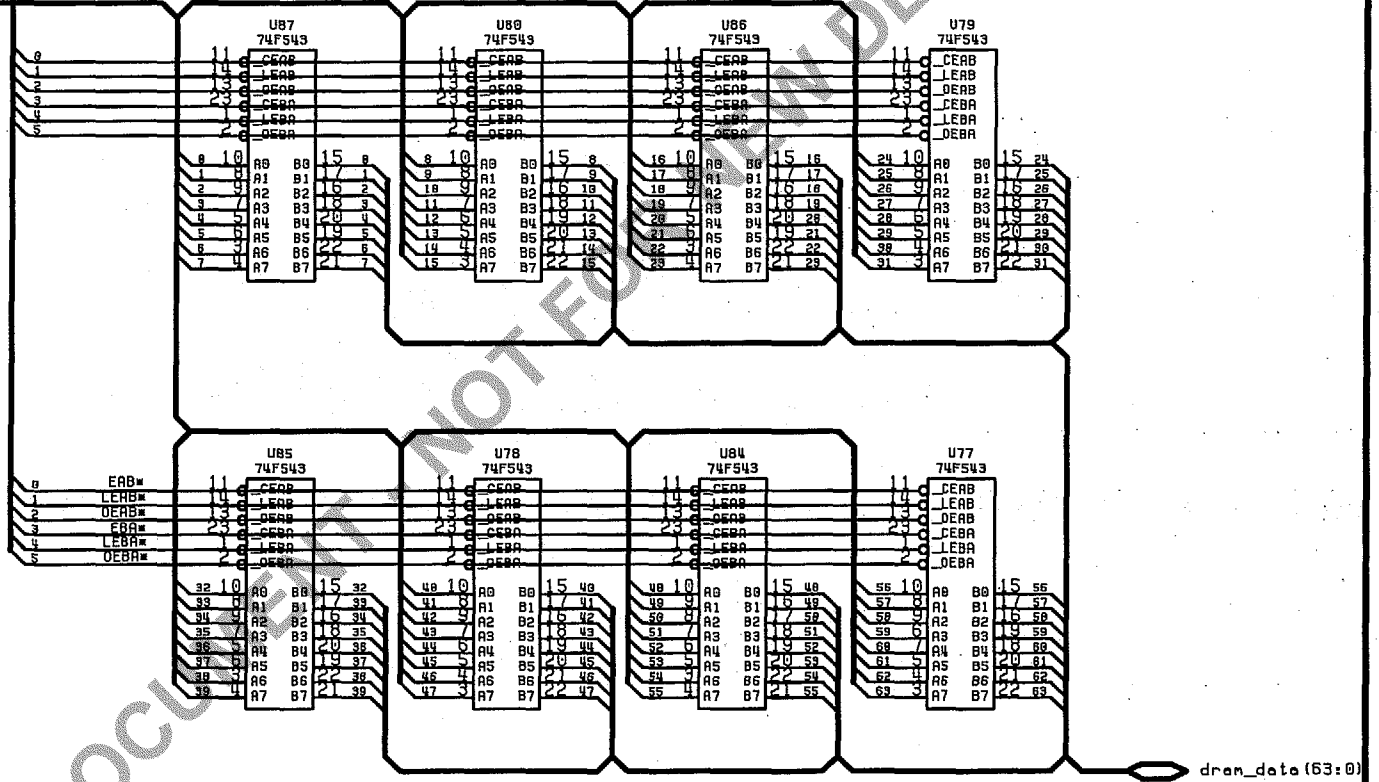
ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

MC88110 ADS BOARD
 h1_data_cntr
 Page 14 of 31 5-Aug-91

D-16

data_cntr (20:0)

sys_data (63:0)



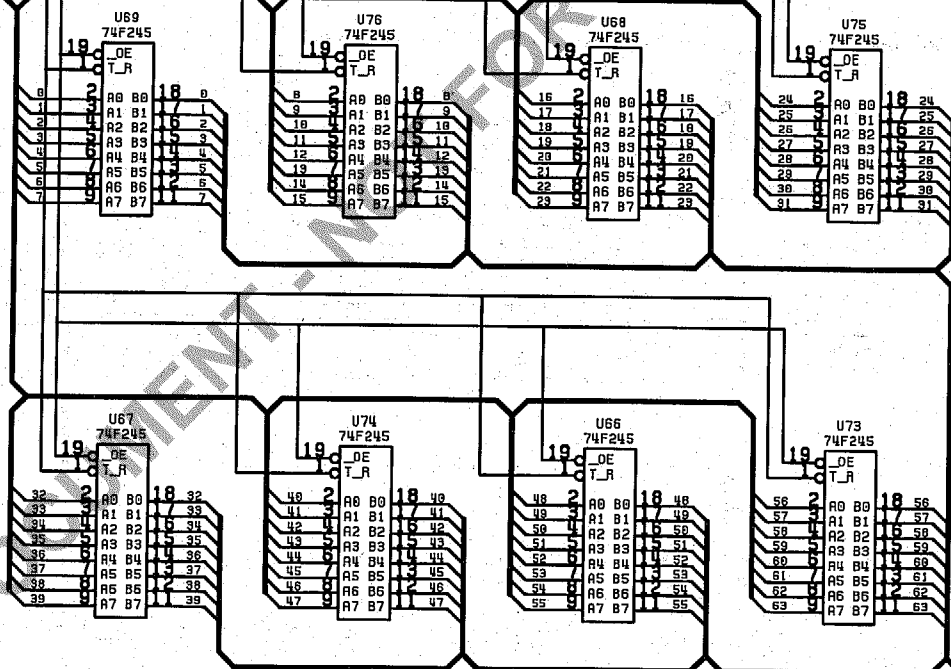
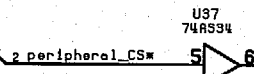
ARCHIVE DOCUMENT NOT FOR DESIGN

MC88110 ADS BOARD
 h1_dram_data
 Page 15 of 31 5-Aug-91

data_cntr (20:0)

addr_decode (7:0)

sys_data (63:0)

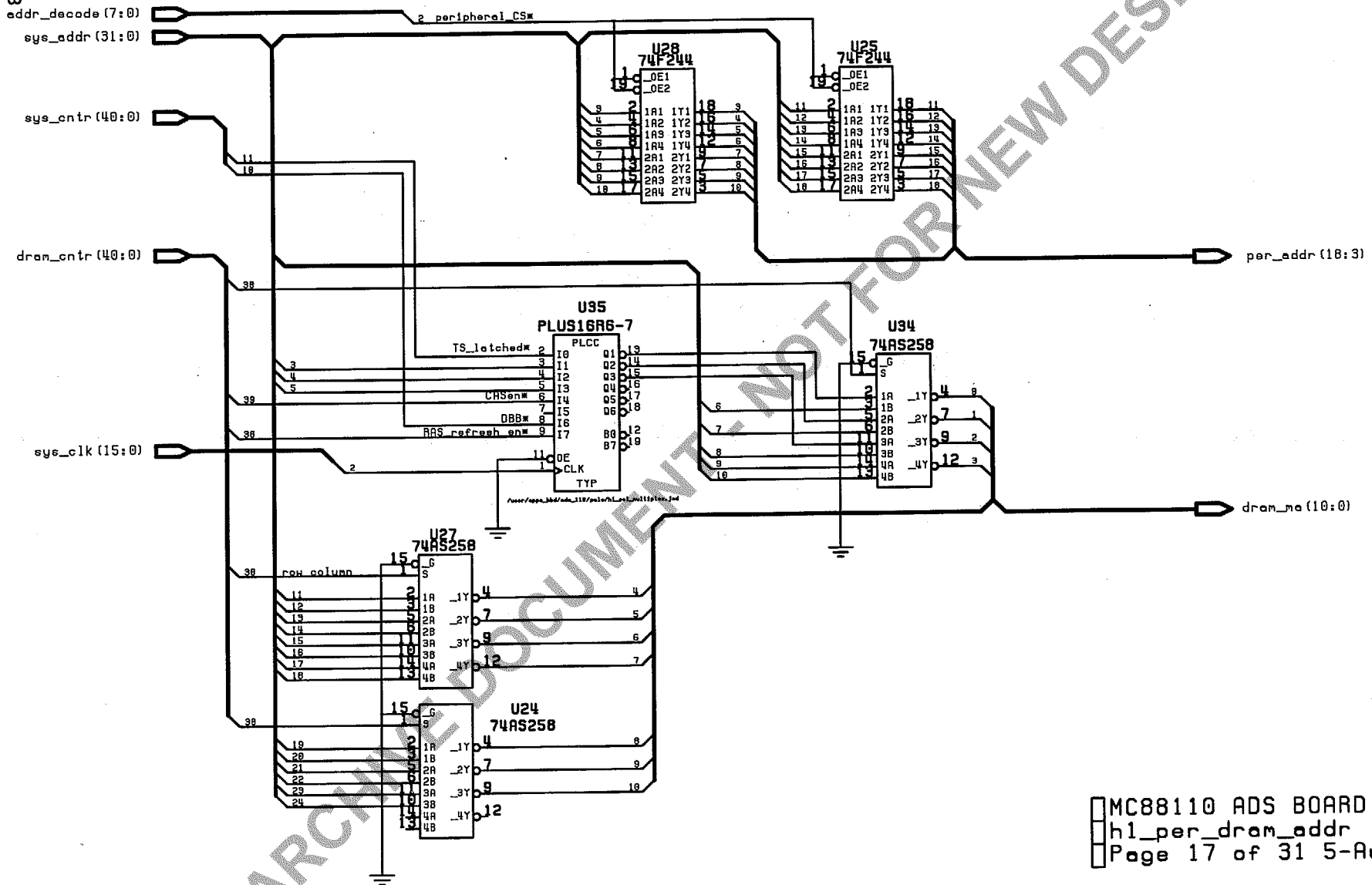


per_data (63:0)

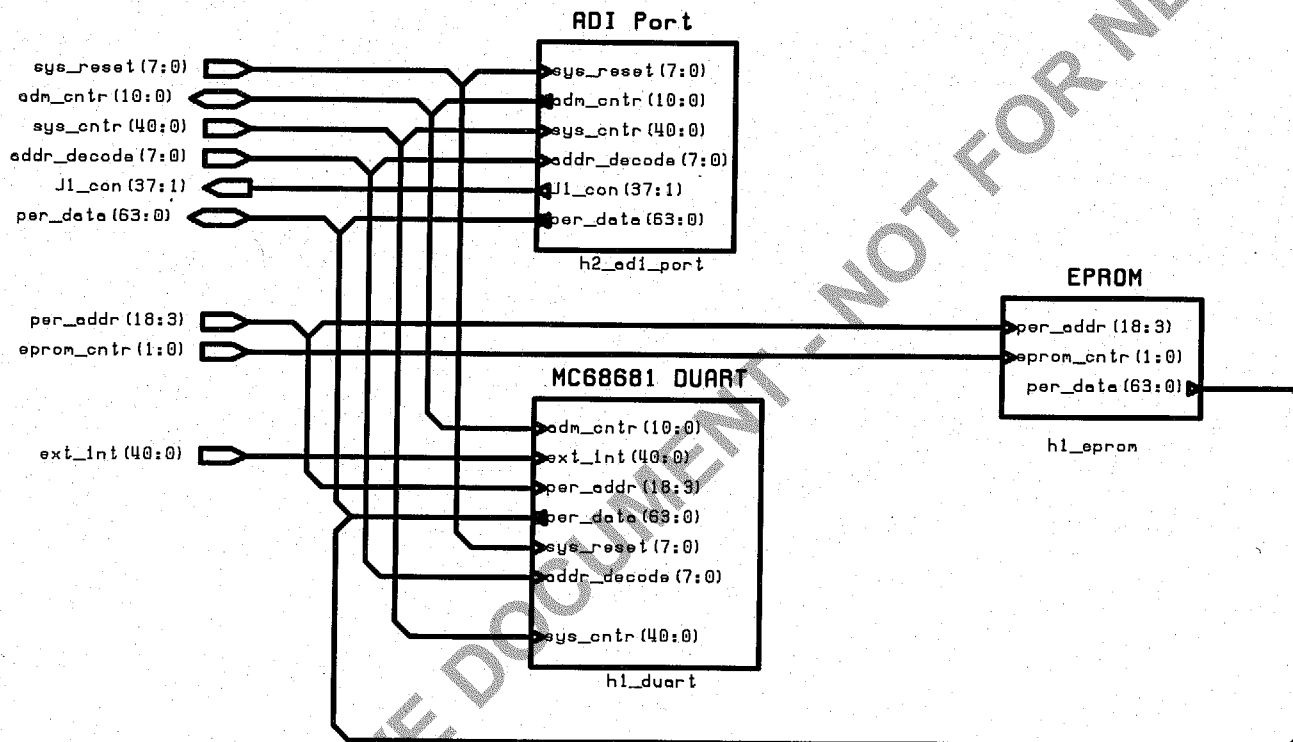
ARCHIVE DOCUMENT FOR NEW DESIGN

MC88110 ADS BOARD
 h1_per_data
 Page 16 of 31 5-Aug-91

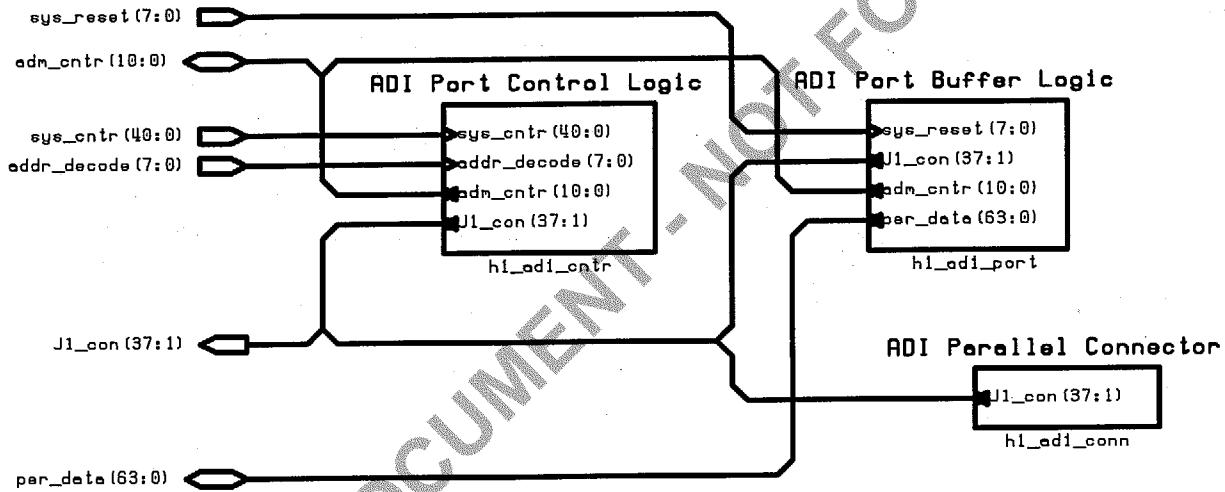
D-18



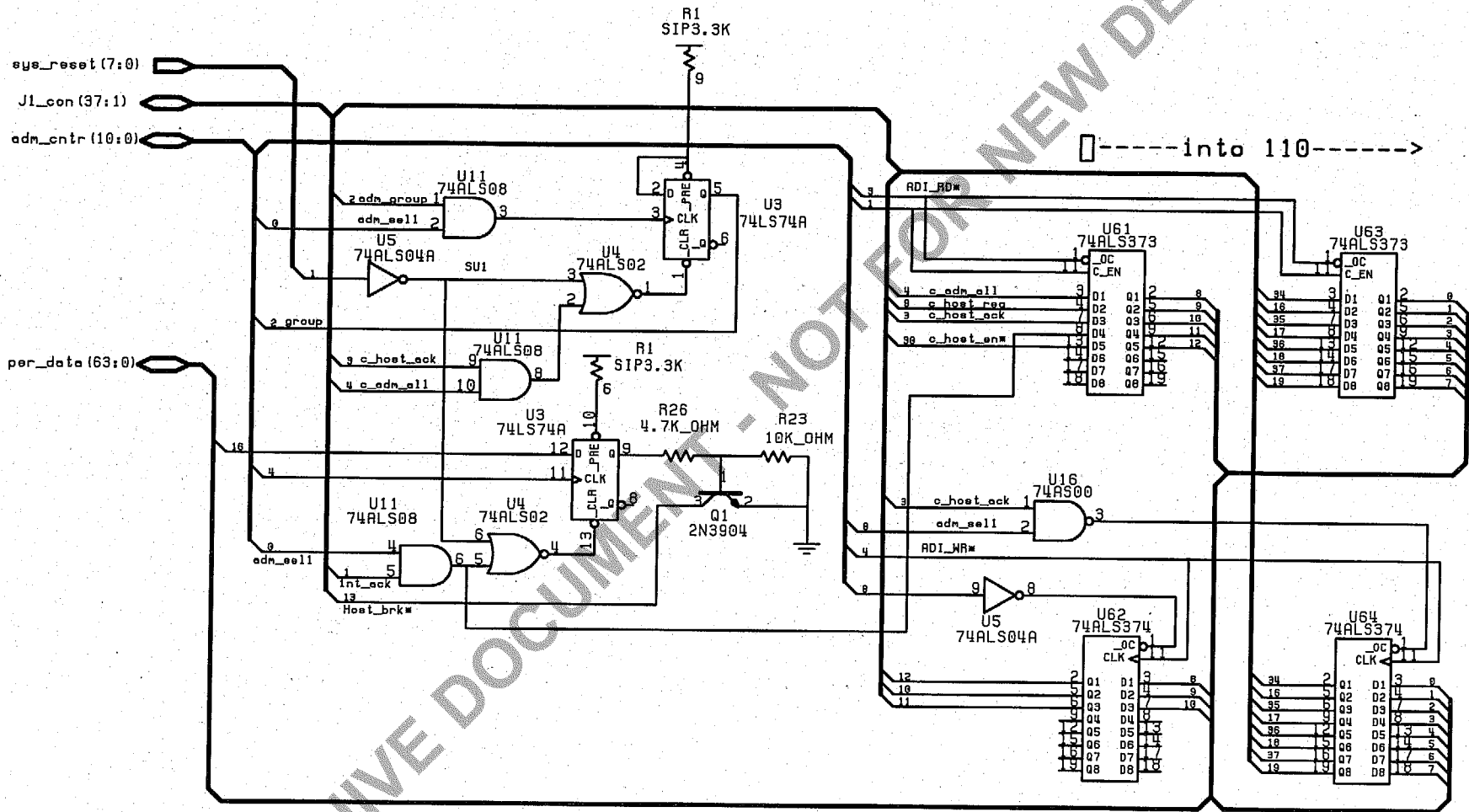
MC88110 ADS BOARD
 h1_per_dram_addr
 Page 17 of 31 5-Aug-91



ARCHIVE DOCUMENT - NOT FOR NEW DESIGN



ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

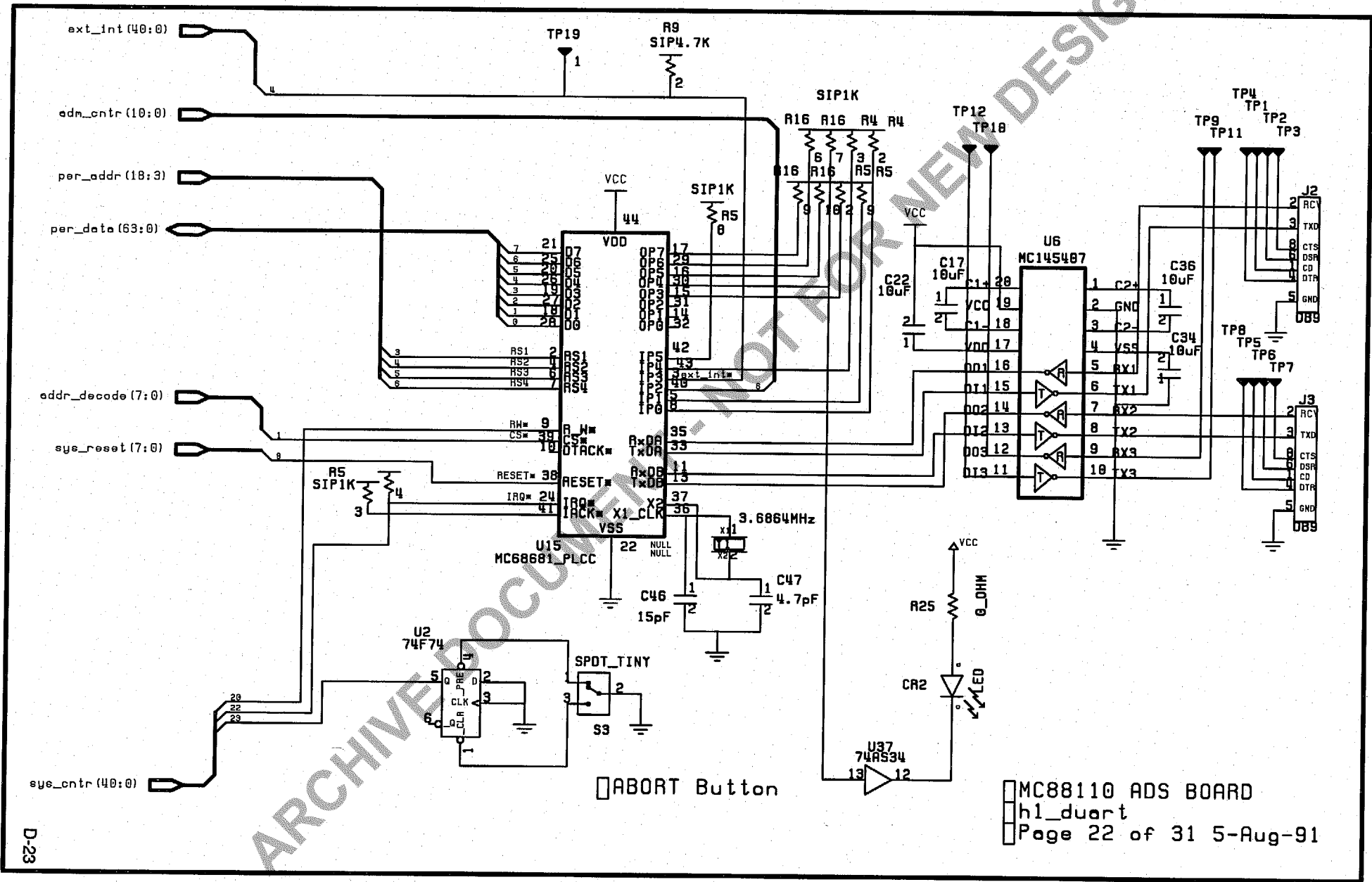


into 110

out of 110

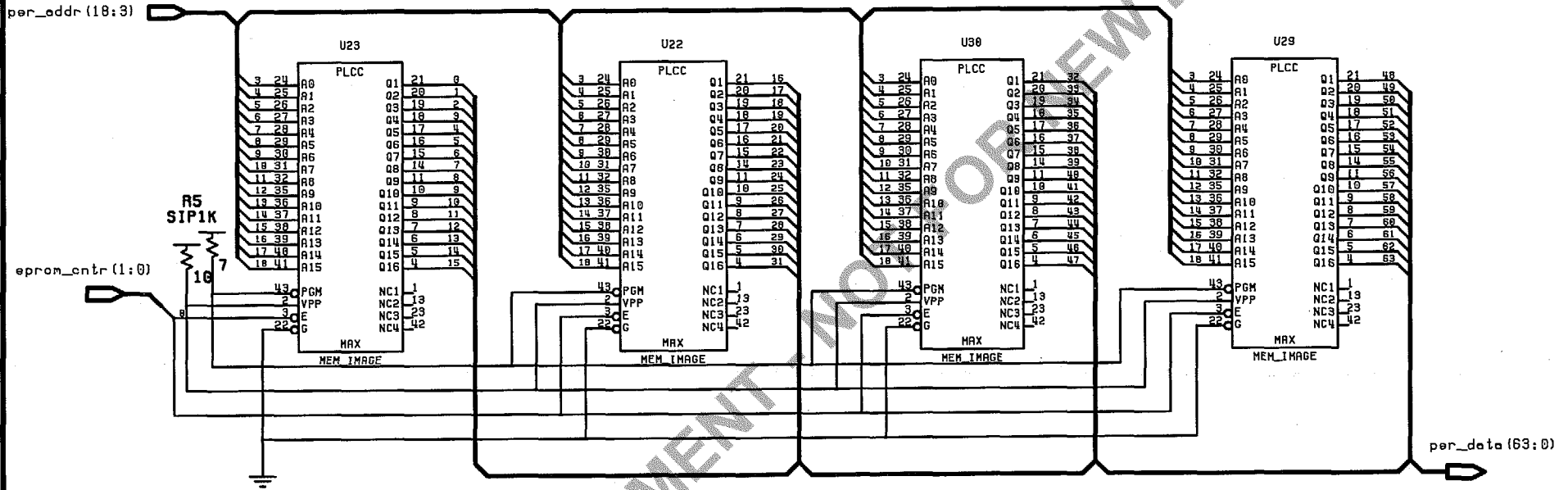
MC88110 ADS BOARD
 h1_adi_port
 Page 20 of 31 5-Aug-91

ARCHIVE DOCUMENT - NOT FOR NEW DESIGN



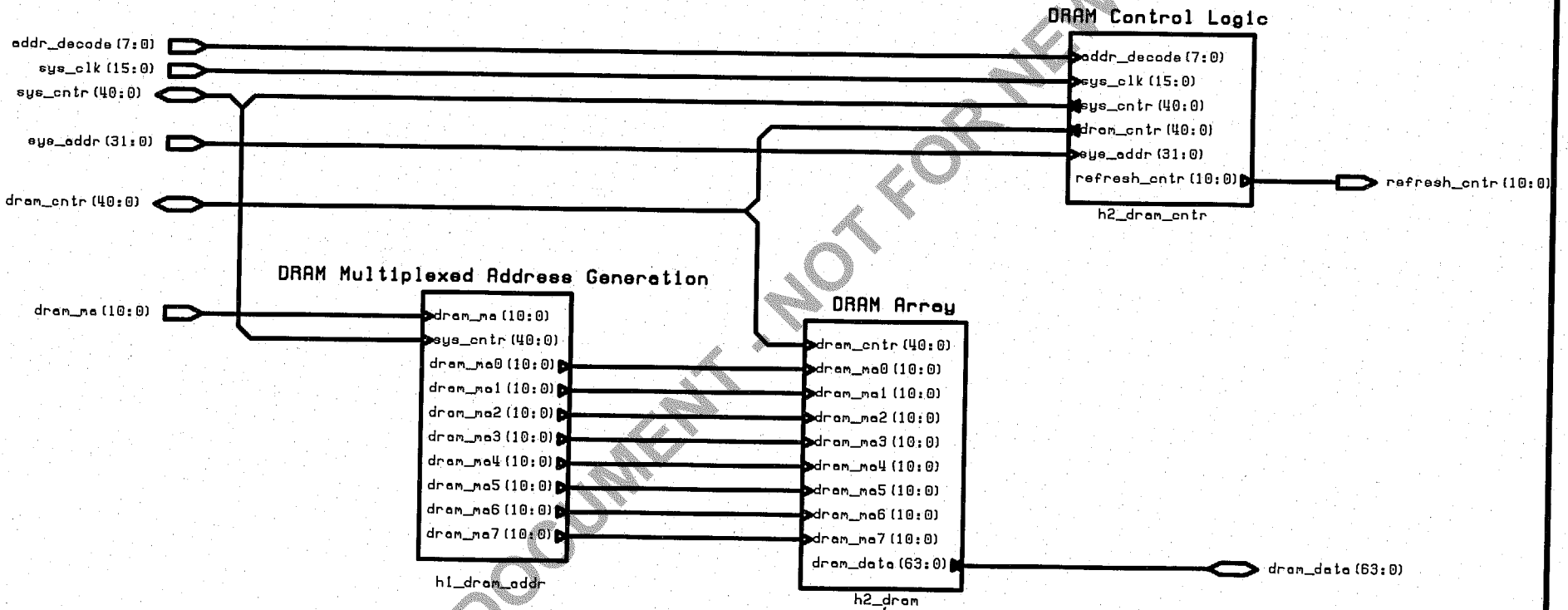
ABORT Button

MC88110 ADS BOARD
 h1_duart
 Page 22 of 31 5-Aug-91



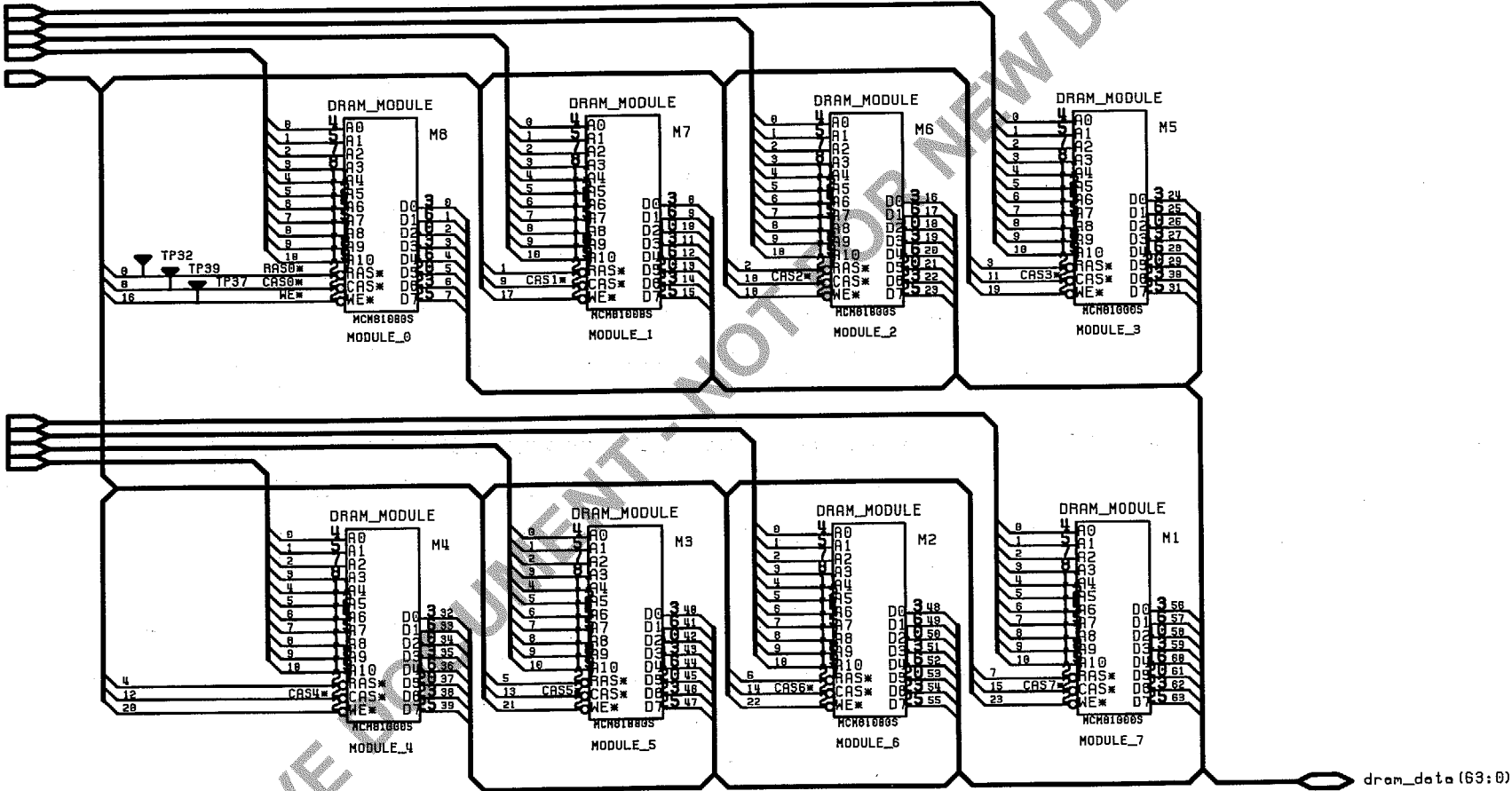
MC88110 ADS BOARD
 h1_eprom
 Page 23 of 31 5-Aug-91

ARCHIVE DOCUMENT NO. 07 NEW DESIGN



ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

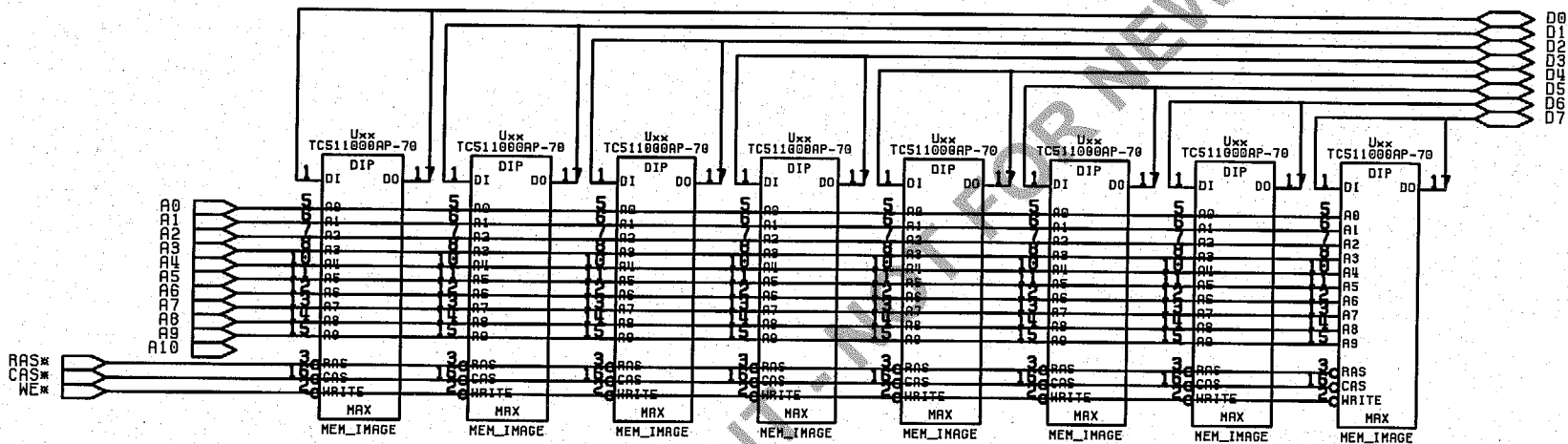
dram_ma3 (10:0)
dram_ma2 (10:0)
dram_ma1 (10:0)
dram_ma0 (10:0)
dram_cntr (40:0)



dram_ma7 (10:0)
dram_ma6 (10:0)
dram_ma5 (10:0)
dram_ma4 (10:0)

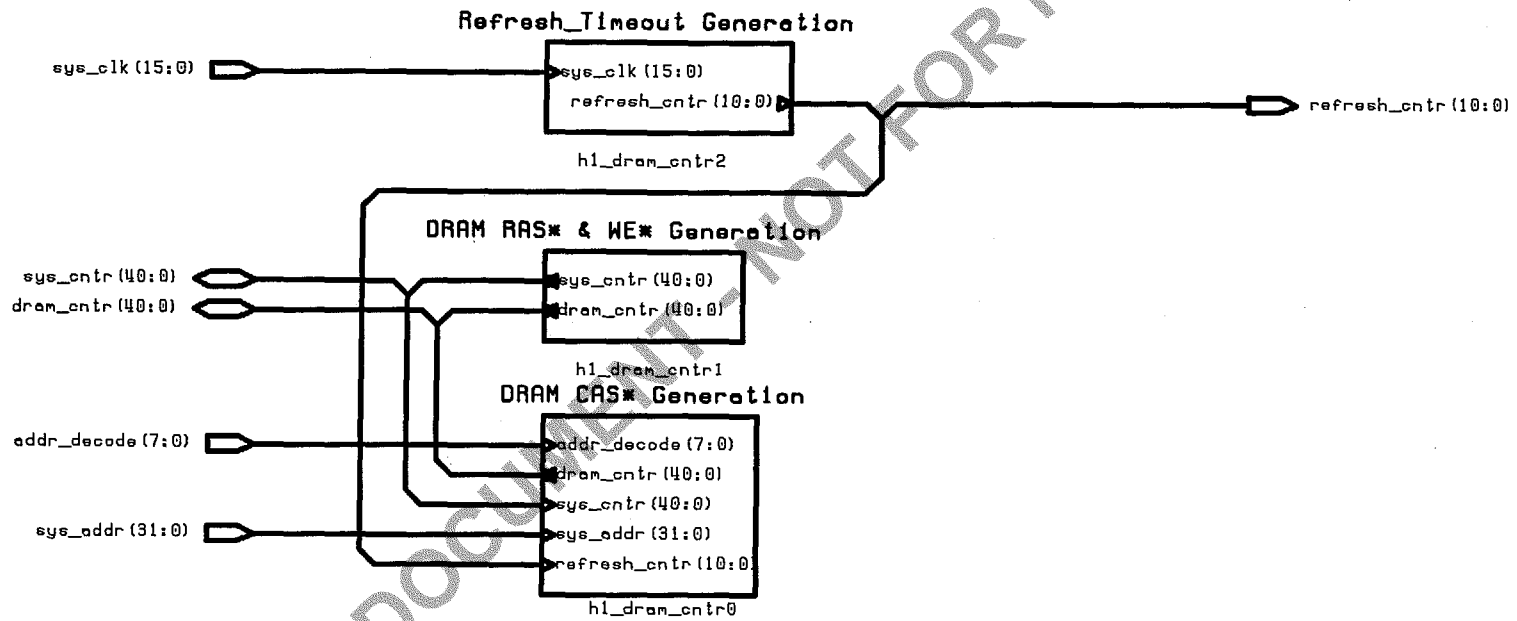
dram_data (63:0)

MC88110 ADS BOARD
h2_dram
Page 25 of 31 5-Aug-91



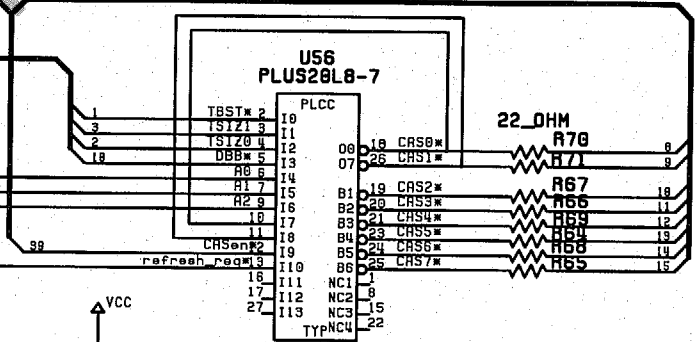
ARCHIVE DOCUMENT FOR NEW DESIGN

MC88110 ADS BOARD
 h1_dram_module
 Page 26 of 31 5-Aug-91

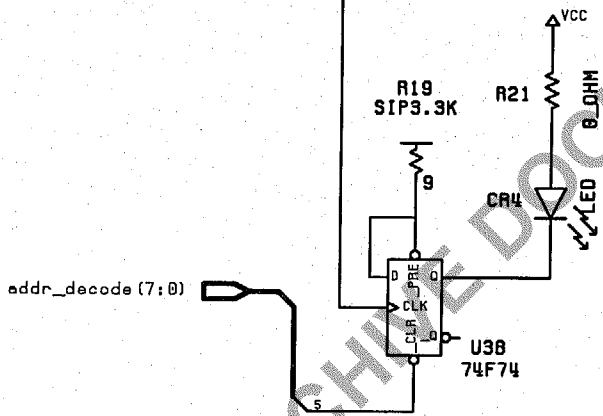


ARCHIVE DOCUMENT - NOT FOR NEW DESIGN

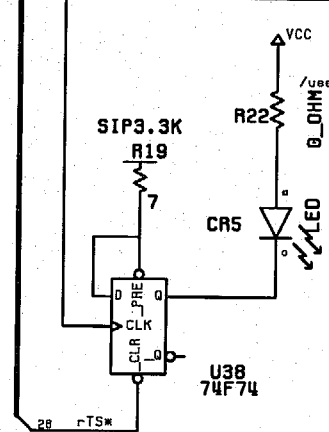
dram_cnr (40:0)
 sys_cnr (40:0)
 sys_addr (31:0)
 refresh_cnr (10:0)



22_OHM
 R70
 R67
 R66
 R69
 R64
 R68
 R65

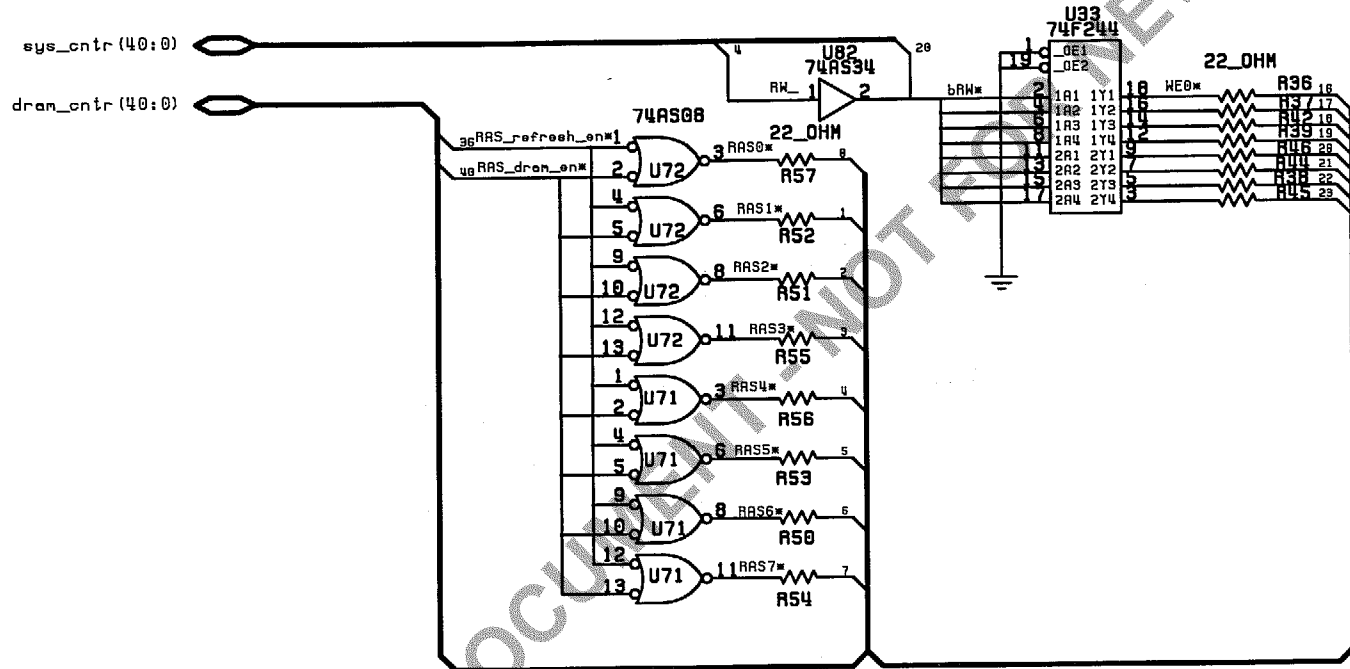


Decode LED

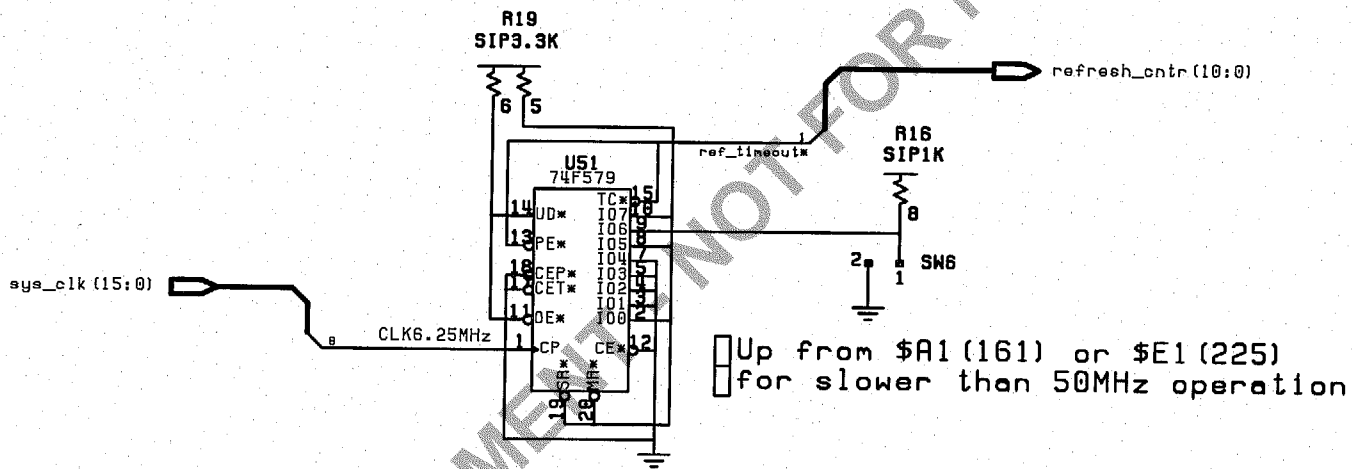


Heart Beat LED

MC88110 ADS BOARD
 h1_dram_cnr0
 Page 28 of 31 5-Aug-91

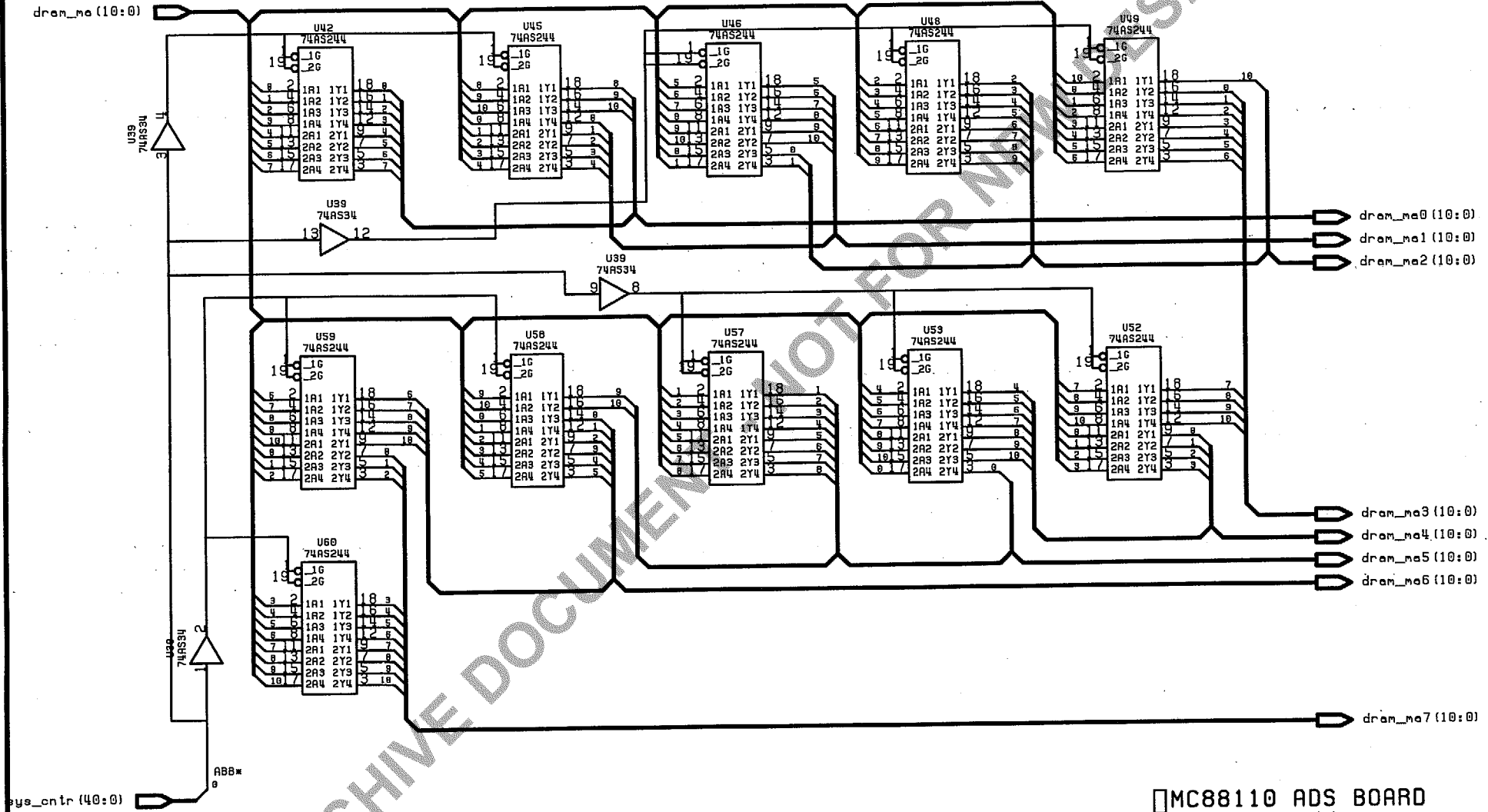


ARCHIVE DOCUMENT NOT FOR REUSE DESIGN



ARCHIVE DOCUMENT NOT FOR NEW DESIGN


MC88110 ADS BOARD
 h1_dram_cntr2
 Page 30 of 31 5-Aug-91



MC88110 ADS BOARD
 h1_dram_addr
 Page 31 of 31 5-Aug-91

ARCHIVE DOCUMENT NOT FOR REPRODUCTION

PRELIMINARY DOCUMENT - NOT FOR NEW DESIGN

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.

EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.

ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.



MOTOROLA

1ATX31329-0 PRINTED IN USA 5/93 IMPERIAL LITHO 91366 18,000 MPU YGABAA



AN1217/D