"S.P.U.D.S."

A STANDARDIZED PROGRAMMABLE USER DEVELOPMENT SYSTEM

by

William B. Baringer

Memorandum No. UCB/ERL M84/4

13 January 1984

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

"S.P.U.D.S."

A STANDARDIZED PROGRAMMABLE USER DEVELOPMENT SYSTEM

by

William B. Baringer

University of California

Berkeley, California

December, 1983

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, to partial satisfaction of the requirements for the degree of Master of Science, Plan II.

ABSTRACT

As the complexity of integrated circuits continues to increase, the sophistication of the equipment used to evaluate these circuits must keep pace. This report describes a programmable evaluation and development system that allows high-level language control and testing of circuits and systems.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to all of the Speech Group for their ideas and support. I would especially like to thank Professor Robert Brodersen for bringing me "on-board".

TABLE OF CONTENTS

CHAPTER 1

HIGH-LEVEL DESCRIPTION

1.1. Introduction

Current research interests include the implementation of signal processing algorithms into integrated circuits. This enables real-time processing of large quantities of information. As CAD tools and production costs make chip development more favorable to the engineer, high-speed dedicated processors have become more feasible to implement, and the compromises required in using general-purpose processors no longer exist. Because of the ease with which digital signal processors can be completely designed and laid out automatically with new CAD tools, most of the signal processing implementation is done with digital circuitry.

In the design and evaluation of complex systems and integrated circuits, simulation using MSI and SSI chips is no longer feasible. As an alternative, the designer may choose to have prototypes fabricated and then build test fixtures to exercise the chips. The testing and characterization of these chips is often done by converting analog control signals to digital signals to be used as inputs to the chip. The resulting digital outputs of the chip are then converted back to analog. Any data buffering, storage, conversion, or processing outside of the chip can be done by digital or analog means, but often require dedicated test fixtures that are rarely used more than once. The problems of noise, accuracy, and dynamic range can arise from the chain of data conversions. Many of these problems can be alleviated by the use of a standardized, microprocessor-based test fixture that uses digital I/O

and is programmable in a high level language.

This report describes "SPUDS", a Standardized Programmable User Development System. SPUDS is a compact, multi-purpose micro-computer, based on the Intel 80186 microprocessor and built on an Intel multibus board. It contains ROM, RAM, multibus interfacing, and a dual serial I/O port. The remaining two-thirds of the board is available for the development and testing of prototype systems and chips. The designer may then program the SPUDS board in a high-level language to control and monitor the prototype system. This can be done through the dual serial I/O port, communicating with a terminal and a host computer such as a VAX. Alternately, program development can be done on a host machine that supports a multibus card-cage. This enables a much closer link between signal processing chips and mainframe computers. Significant data processing can be done in the '186, or additional processing and data output can be done with the host computer and its printers and plotters.

Current trends in high-power computers have been towards the "mini-mainframes". These are stand-alone computer "workstations" that are configurable to the user's particular needs. We are using the Sun Microsystems 68000-based workstations. Each user has a complete computer running under the UNIX operating system, with optional disks, tape drives, graphics, etc. Several slots are then available in the multibus card cage in each Sun for SPUDS boards. In this way, the development of new I.C. signal processors leads immediately to custom boards available to any workstation user.

1.2. Architecture

The block diagram given in section 5.4 of this report shows the architecture of "SPUDS". Based on the Intel 80186 microprocessor, this development

board uses the Intel multibus board and protocol. The 32 kbyte area of EPROM is large enough to accommodate all necessary initialization routines and several non-trivial user programs. The 128 kbyte area of dynamic RAM has a reserved area in the lower address space for interrupt vectors, but otherwise is totally available to the user. The board is easily expandable to one-half megabytes of RAM at no additional cost of area. With the Intel 80186 operating at 10 MHz, the bandwidth of the RAM is 5 Mbytes/sec. The multibus interface contains a 16-bit data port, a 4-bit control/status port, and maskable interrupt facilities for high bandwidth. When SPUDS is used in a multibus card cage of a host computer, as in a Sun workstation, it appears as four memory addresses to the multibus, as decoded by the control unit. The dual channel serial I/O controller operates at a range of baud rates.

There are six major busses shown on SPUDS's block diagram. The AD bus is the '186's multiplexed address and data bus. The BAD bus is a buffered version of the AD bus. The A bus contains the most significant address nibble from the '186, and the BA bus is simply the buffered version. The LA bus carries the latched address for accessing memories. The MA bus is a multiplexed address bus that has first the least significant half of the address bus and then the most significant half.

The EPROM receives its address from both the LA and MA busses, and sends its data output onto the AD bus. The DRAM is addressed by the MA bus alone, and data is transferred over the AD bus. The AD bus has little current driving capability, so it is fairly limited in the amount and type of devices that can be attached to it. Therefore, most peripherals will connect to the BAD bus. The multibus control/ status and data ports are one example. The dual RS-232 port has a data transceiver that is also attached to the BAD bus.

1.3. Firmware

User programs are currently developed in the language "C" on a VAX 11/750 running under UNIX, although with the appropriate compiler, any language could be used. The user's program should be compiled and assembled into 80186 code so it can be shipped over the serial lines into SPUDS's dynamic RAM. The firmware in the EPROM is written so that a terminal may be connected to serial channel B and the VAX or other host processor to channel A. SPUDS then acts in a "terminal emulator" mode where characters sent from the terminal's keyboard are shipped to the VAX, and characters received from the VAX are sent back to the terminal's screen. However, if a certain string of control characters is received from the VAX, the following code is not sent to the terminal, but is stored in RAM. At the end of the transfer of the user's program, another set of control characters is sent as an end-of-text indicator, and program execution of the '186 commences at the beginning of the new program in RAM. A typical program loaded into RAM could then allow control of SPUDS via the terminal, and send data back to the terminal for examination. Data may also be sent to the VAX for further processing.

Use of the serial I/O lines has the inherent drawback of low bandwidth. Operating at 9600 baud yields less than one kilobyte per second of information transfer. Use of SPUDS in a multibus-based host computer allows data transfers of 2.5 megabytes per second (assuming the data is read from RAM and then written to the multibus in a DMA mode.) The current firmware does not support the necessary multibus drivers, and the Sun workstations do not yet have the complementary software drivers inserted in the Unix operating system, but these enhancements are under development.

CHAPTER 2

LOW-LEVEL HARDWARE DESCRIPTION

2.1. Eprom Hardware

SPUDS has been designed to accommodate any 24- or 28-pin EPROM chips, allowing from 2K words to 32K words of ROM. Two 8-bit EPROMs are addressed in parallel to achieve a 16-bit word. Two wire jumpers can be chosen such that address lines and power are appropriately wired for each EPROM type. (See Section 3.1, Alternative Configurations.) For example, with the currently used 2Kx8 EPROMs (2716), pins 23 and 26 are tied to Vcc. Pin numbers given are for the 28-pin socket, even when 24-pin chips are used. The 24-pin chips are inserted in the 28-pin sockets such that pins 1, 2, 27, and 28 are left open.

The '186 selects the EPROM through the '186's upper memory chip select line UCS*. The UCS* is wired to the EPROMs' chip enable CE*. The BRD* line (a buffered version of the pin OE*. The UCS* line is currently memory-mapped at FF000-FFFFFH with 3 wait states, by initializing the appropriate programmable chip select registers of the '186. Faster EPROMs will allow operation at 2 or even 1 wait state in every read cycle. Larger EPROMs will require the memory mapping to be changed; the 2732s will be mapped at FE000-FFFFFH, the 2764s at FC000-FFFFFH, etc. The EPROM must be mapped in the upper area of memory so that upon resetting the '186, the first instruction fetch, made at FFFF0H, is in ROM and is a valid instruction. Fewer wait states may be used with faster EPROMs.

The two EPROMs' data lines are connected directly to the low and high bytes of the '186's multiplexed address/data bus AD<15,0>. They are not connected to the buffered AD bus (BAD<15,0>) because the EPROMs do not have the current drive capability needed for that bus. The transceivers between the AD and BAD busses must not drive the AD bus during a read of EPROM.

The address lines to the two EPROMs must be latched and held during the entire read cycle, since the '186 puts first the address and then the data on the same bus during a single cycle. The operation for the dynamic RAM (DRAM) has a similar requirement for a latched address, so the two memories share a common address latching scheme. (See Section 2.2, Dynamic Ram Hardware) Address bits A8 through A0 of the EPROMs are connected to the latched address bus LA<9,1> and are valid for the entire read cycle. The remaining high order address bits of the EPROMs are wired to the multiplexed address bus MA<6,2> and are valid as high order address bits from the '186 after CAS* has gone low.

EPROMs are currently programmed by compiling and assembling a program, and then splitting it into even and odd bytes to be put into the two EPROMs separately. These even and odd hex files are the arguments of a unix routine called "blast" that ships the appropriately massaged code to the EPROM programmer in 140 Cory and then burns it into the new EPROM.

2.2. Dynamic Ram Hardware

SPUDS has been designed to accommodate the current generation of 64Kx1 (Hitachi HM4864P-2) dynamic RAM chips and the next generation of 256Kx1 DRAM. Two banks of eight chips each are addressed in parallel. Selection between the banks is controlled by the '186's byte high enable signal (BHE*) and the latched address bit LA<0>, resulting in a high or low byte access (an odd or even byte access). This yields a 64K word (or 128K byte) memory space with currently chips.

Selection of the DRAM by the '186 is achieved through the middle chip select lines MCS0-3*. These are currently memory-mapped at 00000-1FFFFH for MCS0*; 20000-3FFFFH for MCS1*; 40000-5FFFFH for MCS2*; and 60000-7FFFFH for MCS3*. Thus, the current 128K byte memory needs only MCS0* as an address decode signal. The 256Kx1 chips will require additional gating so that the memory banks are selected if any of MCS0-3* is active, given that the current memory access scheme is adhered to.

The DRAM is accessed in an "early-write" fashion; that is, the write enable control is valid before the row address strobe is active. This allows the data input and output lines of each chip to be tied together. The resulting data I/O lines are connected to the '186's multiplexed address/data bus AD<15,0> directly, since the DRAMs do not have the current drive capability necessary for the buffered AD bus (BAD<15,0>). Thus the transceivers between the AD and BAD busses must not drive the AD bus during a DRAM read.

Address lines from the '186 must be latched because of the time multiplexing of the address and data on the '186's AD bus. A similar requirement exists for the EPROM, so a common latching scheme is used. Latching of the

address is achieved with the '186's address latch enable signal ALE. The address is latched in two AMD29841 10-bit latches with tri-state outputs. The latch connected to the least significant address bits has its outputs enabled at all times. These outputs are the LA<9,1> bus that provides constant address information during the entire CPU cycle. The LA bus is fed to an AMD29827 10-bit tri-state buffer. A multiplexed address bus MA<9,1> is then derived from the output of this tri-state buffer and the tri-state output of the most significant address latch. The multiplexing control signal MUX* is connected to the tri-state output controls so that first the low order address bits are presented to the DRAM and latched internally, followed by the high order bits. The low order bits, or "row" bits of the address are strobed in by the row address strobe signal RAS*, active if either of the BRD* or BWR* lines are active. The high order address bits, or "column" bits, are latched in by a conditioned version of the column address strobe CAS* signal. CAS* is simply a delayed version of RAS*. The delay is set by a multi-tap digital delay line (DDL) (Belfuse 0447-0050-02) and is 40 nS. The MCAS* signal is derived by conditioning CAS* with MCS0*. Conditioning MCAS* with BHE* yields HICAS*, to access the high byte of the currently addressed word, and MCAS* conditioned with LA<0> gives the LOCAS* signal for the low byte.

A second tap off of the DDL generates the MUX* signal, to swap the address on the MA bus from low address to high address. This is again a delayed version of RAS*, delayed by 20 nS.

To summarize, the dynamic RAM operation will be described in a time-ordered sequence. For a DRAM read or write, the MCS0* line is active, the AD bus presents the address, and the ALE latches the address. MUX* is inactive, so the low address bits are presented to the DRAM. RD* or WR* become

active, causing RAS* to strobe the low address into the DRAM. Then MUX* becomes active, and the high order address is presented to the DRAM. Depending on whether the access is to an even address byte, an odd address byte, or an even address word, the LOCAS*, or HICAS*, or both LOCAS* and HICAS* lines will strobe the high address into the DRAM. (If the '186 accesses an odd address word, it automatically does an access to an odd address byte, followed by an access to an even address byte of the next word address.)

Notice that the MCS0* line need not be active to generate the RAS* signal. Sequential fetches from EPROM, for example, will strobe in the low order address to the DRAM on each fetch. This allows "refresh" of the DRAM by simply reading 128 NOPs from EPROM. This is important when initializing SPUDS from EPROM and needing to keep the DRAM's contents valid until actually jumping into operation in DRAM.

Refreshing the DRAM must occur at least every 2 mS to guarantee valid memory contents at all times over all operating temperatures. (It's actually possible to refresh as infrequently as once per second at room temperature.) The 2 mS timing is accomplished with the '186's internal timer #2 that is currently initialized to interrupt every 2 mS. It's interrupt service routine resets the internal 2 mS timer, executes 128 NOPs, resets the internal interrupt controller, and returns from interrupt.

2.3. Dual Serial I/O Channel Hardware

The Intel 8274 multi-protocol serial controller is a dual channel serial communications chip. It is also called a "dual SI/O", meaning serial I/O, or a "DART", for dual asynchronous receiver/transmitter. It can be operated in a polled, wait, interrupt driven, or DMA driven environment, in an asynchronous, bit-synchronous, or byte-synchronous mode. It is currently operated asynchronously, for standard RS-232 I/O, in a polled mode, although this can be changed through the programmable control registers of the 8274 and the '186's internal interrupt controller.

The chip has two internal byte-wide registers for each serial channel. One register transfers data bytes, and the other contains control capability and status information. Access to these registers is through the RD*, WR*, CS*, A0, and A1 lines. As noted in the schematics, these connect directly to the BRD*, BWR*, PCS4*, LA<1>, and LA<2> lines. PCS4* is initialized upon system reset to decode at 0200-027FH in I/O space, with two wait states. Two wait states are required to meet the timing specifications of the 8274 and '186. Therefore, a read or write to 0200H, 0202H, 0204H, and 0206H will access the channel A data, channel B data, channel A control/status, and channel B control/status bytes.

The byte-wide data bus of the 8274 is connected to the buffered address/data bus (BAD<7,0>) through an 8-bit transceiver ('LS245). Depending on the exact system configuration, this transceiver may be necessary for current drive considerations, but a larger constraint is placed by the timing specifications of the 8274. The delay between a read or write signal going inactive and the tri-state drivers of the 8274 going into a high-Z state is too long for direct connection to the BAD bus: the next address of the '186 is

present on the bus before the 8274 data is off of it. The direction of the data transfer is controlled by the '186's data transmit/receive signal DT/R*. The PCS4* signal conditioned by the '186's data enable signal DEN* enables the transceiver onto the bus.

The 8274 has an interrupt request line (INT*) that can be programmed to be active on a choice of conditions. For example, receiving a new character from the serial input can be programmed to send an interrupt to the '186. The chip also has the capability to issue a vector address in an interrupt acknowledge cycle, with the vector depending on the interrupting condition. However, the current SPUDS does not use this mode of interrupt. (See Section 3.3, Bugs.) Therefore, the interrupt acknowledge line INTA* of the 8274 is tied to Vcc. The INT* line must be pulled high with an external resistor and then inverted to match the interrupt input line specifications of the '186. The resulting interrupt signal is connected to the '186's INT2.

The serial transmit lines from each channel are fed to a 75150 RS-232 compatible line driver. The incoming serial data goes through a MC1489A RS-232 line receiver before connecting to the 8274. The line driver is powered by the +/- 12 volts available on the multibus.

The serial transmit and receive section of each channel of the 8274 require a baud rate clock of 16, 32, or 64 times the actual bit rate sent or received. (Operation in a x1 mode is not recommended.) These clock inputs are TxCa, TxCb, RxCa, RxCb. In the current configuration, all four clock inputs are fed from a common source, in a x16 mode. Timer 0 of the '186's internal counter/timer channels is used as the common baud rate clock. This timer is configured to divide the '186's 10 MHz clock by 64, giving a 156,250 Hz clock. When this clock is divided by 16 by the 8274, 9600 baud

rate communications is established. This clock allows the user to change the baud rate of either or both channels to 4800 or 2400 baud by changing the proper internal control registers of the 8274. Changing the '186's counter/timer clock rate changes the baud rate of both communications channels at once.

In addition to the baud rate clocks, the 8274 needs an independent clock (CLK) to run its internal system. This is chosen to be a 2.5 MHz clock, derived by dividing the '186's 10 MHz clock by 4 with an external 4-bit counter (LS163). This counter also provides clocks at 5 MHz, 1.25 MHz, and 625 kHz.

The 8274 has a number of lines used for communication "hand-shaking"; that is, connections to a modem or other communications device that determine and send status of the devices on each end of the link. All of these lines have been wired in a default mode, so that the 8274 believes that the link is good. The 8274's DMA request lines are also not used.

The 8274 can be reset or initialized by its RESET* line. This is connected to SPUDS's SYSRESET*, originating at the '186. Upon reset of the '186, the 8274 is reset, and the '186 then initializes the internal control registers of the 8274. The current mode of operation is: divide the external baud rate clock by 16; use 1.5 stop bits; do not use parity; disable interrupts; and use 7 bits/char in both transmit and receive modes.

2.4. Multibus Interface Hardware

Two bidirectional ports are available for communication between the multibus and the CPU. One is a 16-bit port intended for data transfer. The other port is designed as a control and status register (CSR).

From the CPU's perspective, the data port is I/O-mapped at address 0004H, and the CSR is at address 0006H. From the multibus's perspective, the data word is at 080,002H and the CSR is at 080,000H. (These latter addresses are mapped in the Sun's virtual memory at 180,002H and 180,000H.)

The data port consists of two AMD2953(A) inverting bidirectional latches with tri-state outputs. This "bi-port" is equivalent to two LS374s back-to-back, in a single 24-pin package. Data can be clocked in and read out independently from either direction.

The CSR has four bits for arbitrary control and status information; their use is left to the designer of the multibus software drivers, from both the multibus's and CPU's perspective. Two other bits in the CSR control and monitor the interrupt machinery of the multibus interface.

When a write to the data port is made from the multibus, a flip-flop is set. If the CPU has enabled this interrupt mode, by setting bit <1> of its CSR high, then the '186 will be automatically interrupted by this write to the data port (on the '186's INT0 line). When the '186 reads this data from the port, the flip-flop is reset and the interrupt is disabled. If the multibus has enabled its interrupt for this mode, by setting bit <8> of its CSR high, then the multibus will be automatically interrupted by the '186's read of the data port(on the multibus's INT4* line). Similarly, when the on the multibus's CSR<9>, and when the multibus reads from the data port, the multibus

interrupt is cleared and the '186 may be interrupted, if its CSR<0> is set. Therefore, writing to CSR bits <1,0> from the '186, or CSR<9,8> from the multibus, sets or resets mask bits for different interrupt modes.

Reading from the CSR<0> by the '186 will monitor the status of data flow in the direction of CPU to multibus. If the multibus has read the data word, this bit will be set. If the '186 has written data in, this bit will be reset. The multibus may monitor the same information, with opposite polarity logic, when reading from its CSR<9>.

Reading from the CSR<1> by the '186 will monitor the status of data flow in the direction of multibus to CPU. If the multibus has written to the data word, this bit will be set. If the '186 has read the data word, this bit will be reset. Again, the multibus may monitor the same information, with opposite logical polarity, when reading from its CSR<8>.

The CSR consists of two LS174 hex flip-flops, and two LS367 hex tri-state buffers. Four of the bits from each chip are wired in a bi-port configuration, and the other bits are connected to achieve the above described operation.

An LS109 dual J/K* flip-flop is used to monitor the state of the last read or write to the data port. An LS51 AND-OR-INV provides the necessary random logic, along with an inverter, and a tri-state buffer that simulates an O.C. driver for the INT4* line.

Decoding of the multibus address bus is accomplished with an 8-bit comparator (AMD 25LS2521). This decodes ADR13H* - ADRBH* so that the signal BA* (base address) is active for any multibus address 80,000H to 80,7FFH. (As mentioned in the section on "alternate designs", another 8-bit comparator could be used to further decode the address of the board.) BA* then enables a 3-to-8 decoder (LS163), which has the multibus signals MRDC*, MWTC*, and

ADR1* as its inputs. The outputs of this decoder are the four signals DATRD*, DATWR*, CSRRD*, and CSRWR*, for data and control/status word reads and writes.

The multibus requires a transfer acknowledge signal for its communications protocol. This tri-state signal, XACK*, must be pulled low any time after data is presented on the bus during a multibus read, and must be off the bus no later than 65 nS after the MWTC* or MRDC* signals are inactive. If either MRDC* or MWTC* are active, a "memory access" signal MA* becomes active. If BA* is also active, indicating a memory access to this address space, then a "enable XACK*" signal ENX* becomes active. This enables the tri-state driver of the XACK* signal. ENX* is also fed to a DDL (digital delay line, Belfuse 0447-0050-02) to delay XACK* until the data is actually on the bus on a multibus read. However, since the ENX* line is not delayed, XACK* will be off the bus in time.

2.5. Installation of Integrated Circuits and Systems

There exist many options available to the user in the connection of I.C.s or systems to SPUDS. This section will discuss some of the possible interfacing schemes.

The most important decision in connecting to SPUDS is the choice of data and address busses. For most uses, the BAD bus will provide the necessary data path, and addressing is rarely needed. For example, if a single I.C. is to be controlled and tested, tri-state data buffers can be connected to the BAD bus from the I.C., or transceivers can be connected if data flow is in both directions. The BAD bus requires a significant amount of current drive, depending on system configuration, so that most NMOS I.C.s will not be able to drive it directly. Once buffers or transceivers are added, a new bus has been created with less critical current drive demands, and several peripherals can share it. Use of the data transmit/receive line DT/R*, data enable line DEN*, and the BRD* and BWR* lines may be necessary to control the data flow.

If an address bus is required, use of BA<19,16> and BAD<15,0> is recommended. This provides 20 bits of addressing, valid during t.1 of the CPU's cycle. If additional RAM is needed for the '186, short of replacing the 64k x 1 DRAM chips with 256k x 1 versions, the MA bus may be used in conjunction with MCS1,2, or 3*. The LA bus provides latched addresses available through the entire CPU cycle, making it very useful for conditioning the peripheral chip select lines. The ALE line allows latching of the address information from the BAD bus during t.1. Using the BHE* may also be necessary, if distinction between high and low bytes is necessary (or whenever access to words at odd addresses is needed).

Using 6, 8, or 10 bit latches, driven from the BAD bus, can provide necessary control of the devices. Different modes of testing or operation could then be established until the next write to the control register.

In the current SPUDS, the PCS0*, BRD*, BWR*, and LA<3,1> are used to create 8 read and 8 write strobes. Two of the read lines and two of the write lines are used for accessing the multibus data and control/status ports. The other 12 lines are available for use by peripherals. They can be used to enable tri-state buffers, read FIFOs, clear flip-flops, write data words, set control bits, etc. As an example, it is helpful to understand their use with the multibus interface.

Currently, only PCS0* and PCS4* are in use. Peripheral chip selects 4, 5, and 6 are programmed for two wait states to be inserted in every read or write cycle. PCS<3,0> use zero wait states. This should be considered when connecting peripherals with different access times to the other peripheral chip select lines.

If more than two wait states are needed, use of the ARDY and SRDY ready signals may be in order. Under certain applications, the TEST* and HOLD lines may be used. The user is referred to the 80186 manuals.

The counter/timer channel 1, both of the DMA request lines, and the interrupt input lines 2 and 3 are all available for use by the peripherals. (Interrupt line 1 is also available if operation of the DART chip in an interrupt mode is not needed.) Initialization and control of these lines is done by internal accesses to the 80186.

The '186's CLKOUT is fed to a 4-bit counter that provides 5 MHz, 2.5 MHz, 1.25 MHz, and 625 kHz clocks, also available to the user.

Resetting the '186 will cause the SYSRESET* line to become active. This can be used by the peripherals to reset the entire system to an initial state.

Since many I.C.s now developed will eventually go into microcomputer-operated systems, it is to the user's advantage to include a simple microcomputer interface on the chip. This may include large current pad drivers, tri-state outputs, ready signals or interrupt requests, clock input, reset input, on-chip FIFOs, and/or on-chip control ports for programmability with address, chip select, read, and write line inputs. However, the required interfacing components can be easily implemented with just a few external chips if it is not desired to include them in the I.C., especially during testing of initial versions.

As an example, the block diagram of the connection of a speech recognition system to SPUDS is shown in Section 5.4 of the Appendices. Two of Berkeley's custom integrated circuits are utilized in this system. One is a 16-channel filter-bank chip, and the other is a parallel processing "time-warp" chip to do dynamic programming.

The filter bank takes its input from digitized speech signals. The resulting Fourier coefficients are loaded into FIFOs, to be read by the CPU in response to an interrupt signal. These FIFOs are connected to the BAD bus. The A/D and FIFOs will be included on the next generation of this filter-bank chip.

The dynamic programming chip shown requires two banks of RAM for efficient operation. The T.P. or template dynamic RAM contains 256 kbytes of memory, connected in a dual-ported fashion with the CPU. Address lines from the CPU are taken from the BA and BAD busses. The LA bus provides additional address information needed by the T.P. address control circuitry.

The D.P. or dynamic programming memory has 96 kbytes of RAM for use by the D.P. chip only. Its data bus is shared with another set of FIFOs that are connected to the BAD bus.

Control signals to this speech recognition system are taken from the 12 available decoded strobes, the CPU's DEN*, BWR*, RAS*, DT/R*, and a 10-bit latch attached to the BAD bus.

2.6. Support Hardware

The '186 can be reset by several different means. If the multibus's INIT* line becomes active, or if an on-board pushbutton is depressed, the RES* line to the '186 becomes active. If the board is addressed through the multibus at any memory address between 80,000H and 80,7FFH and multibus address bit 6 is active, (as in 80,040H, 80,080H, 80,0B0H, etc.) then the RES* will also become active. This allows the master computer controlling the multibus (in this case, the Sun workstation) to selectively reset this board without resetting any other multibus boards.

When the '186 is reset by pulling RES* low, it then strobes its RESET output line high. This is inverted to become SYSRESET*, and is used to reset the rest of the peripherals on SPUDS. Both the multibus control/status port and the dual serial I/O chip need this SYSRESET* signal. Other peripherals may also use it.

Although the '186's AD bus has twice the current drive capacity of the previous family of 8086 chips, because of the number and type of devices transferring data or needing addresses, it is necessary to buffer the bus. Additional peripherals can then be added to the buffered AD bus (BAD). Two 8-bit transceivers ('LS245) buffer the AD<15,0> to the BAD<15,0> bus, and one 'LS367 buffers the A<19,16> to become the BA<19,16>.

The '186 has an internal programmable chip select unit that strobes an output line if a selected area of I/O or memory is addressed. However, many peripherals do not have chip enable or chip select inputs, so that conditioning of the '186's read and write signals with these peripheral chip select lines is necessary. Two 3-to-8 decoders (LS138) are used to accomplish this, with one enabled on BRD* and PCS0* active, and the other with BWR* and PCS0*.

The latched address lines LA<3,1> are fed to the A, B, C inputs of the decoders, so that eight sequential addresses are decoded to become strobes to the peripheral components. Four of these strobes are used for the multibus interface ports. The other 12 are available for use by other peripherals.

CHAPTER 3

FUNCTIONAL CONSIDERATIONS

3.1. Alternative Configurations

Several options of operation of SPUDS are available to the user. Some of these options can be realized by physically changing the wiring on SPUDS, and others will require alteration of programs in the RAM or EPROM.

SPUDS was designed to be compatible with the Sun multibus workstation. Therefore, the most significant data byte to the multibus is on DAT<7,0>, and the least significant byte is on DAT<15,8>, in agreement with the Motorola 68000 used in the Sun. (This is not in accordance with the multibus specification.) Using SPUDS in a different multibus system means swapping all wires connecting to multibus pins 67 through 74 with those of pins 59 through 66.

Interrupting the multibus (in this case, the Sun) is done on the multibus interrupt request line INT4*. This choice of interrupt is easily changed by moving the output of the tri-state driver (wired to simulate an O.C. driver) to another interrupt request pin. However, the multibus specifications list the INT0* line as having the highest priority and the INT7* line the lowest, while the Sun's multibus explicitly uses the opposite priority scheme.

The available address space in the Sun workstation's multibus is between 040,000H and 0C0,000H, depending on system configuration. Currently, the board is decoded at 080,YXXH, where Y is any address from 0 to 7, and X is any 0 through F. Thus, the "base address" of the board is con-

sidered as 080,000H, and this can be changed by rewiring the pins on the 8-bit comparator. The multibus address lines are active-low, so decoding a "0" in the address space means comparing to a "1" or +5 volts on the comparator. The ADR13H* address line is currently wired to the comparator's active-low enable input line Ein*, so it is being decoded to a "1" in the address space. Changing the state of any of the "B" inputs of the 8-bit decoder will change the base address of the board.

The CPU can be reset by addressing any of 080,YWXH, where W is equal to 4, 5, 6, 7, C, D, E, or F. In other words, setting ADR6* true while being in the decoded address space will send a reset signal to the '186, which will reset the entire SPUDS. Rewiring any ADR<AH>* through ADR<2>* in place of ADR<6>* would work just as well to reset the board at the corresponding address space. This is not true if another 8-bit address decoder is used, in which case only ADR<2>* could be used. (ADR<1>* is used to choose between data and control words.)

SPUDS is designed to be flexible in its capacity for size of EPROMs used. The twenty-eight pin sockets used can accommodate any EPROMs from the 2 kbyte 2716 to the 32 kbyte 27128. All pin numbers used are for the 28-pin socket, even for the 24-pin chips. Listed below are the wiring changes necessary for each size of EPROM:

EPROM type	size	Pin 23		Pin 26
2716	2 k	+5	+5	
2732	4 k	LA<12>		+5
2764	8 k	LA<12>		+5
27128	16 k	LA<12>		LA<13>

SPUDS can utilize 256Kx1 DRAM chips instead of 64Kx1 DRAMs with a slight change in the multiplexed address generation. This provides 512 kbytes of scratchpad RAM to the '186. The BAD<9,0> lines remain on the 10-bit latch as wired. BAD<9> is then removed from pin 6 of the other 10-bit latch, and BAD<17> replaces it. It is then necessary to connect MA<9> from the 10-bit buffer to pin 1 of the DRAM chips. The MCS* signal is no longer equal to the '186's MCS0*, but must be active if any of MCS0*, MCS1*, MCS2*, or MCS3* are active. This requires the addition of the equivalent of 3 AND gates. Any differences in timing requirements of the 256Kx1 chips must be considered carefully, and are most likely resolved by changing the RAS*, MUX*, and CAS* timing on the digital delay line. The RAM refresh interrupt service routine would also need modification; adding another 128 NOPs is probably sufficient, depending on the RAM. (Some RAM manufacturers may require 512 NOPs to refresh every 4 mS.) It is the programmer's option to have something useful done in place of a string of NOPs, as long as that routine is done without interruption or waiting.

As mentioned in the section on the 8274 dual serial I/O chip, DMA operation is an option. One or both of the transmit or receive DMA request lines for either communication channel.

Operation of the 8274 with interrupt acknowledge cycles to jump to different service routines depending on the interrupting condition requires connection of the 8274's INTA* input to the '186's INTA1* line.

3.2. Alternative Designs

Different applications may require slight modifications to the given design. For example, if a scratch-pad RAM area of no greater than 128 kbytes is needed, one could replace two of the 24-pin chips with 20-pin chips. These would be the 10-bit BAD to MA register and the 10-bit LA to MA buffer, that could be replaced by 8-bit versions. The three AND gates used to create MCS* could then be removed. Another option is to include additional address decoding of the multibus. This may be accomplished with another 8-bit comparator in cascade with the original. PALs may be used to save board area by replacing random logic gates and the 3-to-8 decoder used on the multibus read, write, and address lines.

Differences in original SPUDS:

The original SPUDS constructed used a more complicated RS-232 line receiver than the final version. With a constant effort to conserve board area, an 8-pin 75141 dual line receiver was used. Two resistors and four diodes preceded the receivers, functioning as signal limiters for the incoming +/- 15 volts. Although the 4-channel MC1489A line receiver is a larger chip, it is designed to receive RS-232 level signals and thus does not require discrete components.

3.3. Bugs

As Intel developed the 80186 microprocessor, early versions of the chip were released for development purposes. These chips were not completely functional, and errata sheets accompanied the chips as documentation. The "step A-1" version of the '186 had a non-functional timer unit and an internal interrupt controller unit that only worked in non-cascade mode. This means that the interrupt controller could not be used in the iRMX 86 compatibility mode. The chip's recommended Vcc limits were 4.0 to 4.8 volts for CPU and DMA operation.

The step "B-1" version of the '186 resolves some of the problems that the A-1 had, but it has its own errata sheets. The known bugs that may affect the operation of SPUDS now or in the near future are:

1) DMA registers: Any read of the upper 4 bits of the 20-bit pointer registers in the integrated DMA controller will yield all zeros. The DMA controller will continue to operate correctly if these registers are read. This does not prevent the DMA controller from responding properly with all 20 bits of the DMA memory location when a DMA cycle is run. The upper 4 bits must still be programmed with their correct value. The given errata sheet proposes the following solution: If the content of these upper 4 bits is required, it can be determined by reading the DMA count register to determine the number of DMA transfers which have occurred and adding this to the value with which the register was programmed.

2) Queue status: This problem will only affect users of the 8087 in conjunction with the 80186.

3) Improper interrupt vectoring: This problem will also affect only those

users of the 8087.

4)Non-contiguous INTA cycles: When using DMA and the internal interrupt controller (cascaded, nested, fully nested or RMX86 modes) it is possible to get a DMA cycle in between the two INTA cycles. Intel proposes a solution: If it is recognized that an interrupt is coming in, external logic should be used to block the DMA request lines until after the first INTA cycle has been completed. This will allow the second INTA cycle to run before the DMA request is recognized. The user should be positive that interrupts are enabled (STI instruction), otherwise DMA may never be serviced.

Currently, no INTA cycles are used for operation of any of SPUDS's peripherals. However, in the future it may be desirable to use the dual serial I/O chip in a conditional vectored interrupt mode. At this point, either the above mentioned additional hardware will have to be added, or purchase of the "step C" 80186 must be pursued, assuming that this bug will be fixed in that version.

5) String move instruction: Essentially, if a string move instruction (MOVS, INS, and OUTS) is fetched but its execution not begun, and a HOLD request is received, (from the internal DMA controller in our case, since no external HOLD is currently done) the '186 will not properly begin the string move instruction after regaining control of the bus. Apparently, this failure will only occur if the last bus operation performed before the HOLD is acknowledged was either a memory or I/O read cycle. The recommended solutions to the problem that apply to SPUDS are either to compare the destination string with the source string after a move to insure that proper execution has occurred, or to insert a write cycle or an instruction fetch cycle immediately before the string instruction. For example the code sequence

"pop AX ; rep movs" is replaced by "pop AX ; jmp A ; A: rep movs". Of course, a simpler solution to the problem would be to just not use the DMA controller or the string move instructions, if that is possible to do.

CHAPTER 4

CONCLUSIONS

This report has described a powerful and compact evaluation and development system that affords a wide variety of potential uses. Using high-level language development tools and providing a flexible digital interface to integrated circuits allows sophisticated single-board systems to be developed. A complete speech recognition system was presented as one example of the uses of this board. Another board is currently being designed to perform character (hand-writing) recognition, while a third is planned for speech synthesizer and/or vocoder integrated circuits. Other boards will be distributed to various I.C. designers for use as programmable test fixtures. Used in conjunction with the new generation of "mini-mainframe" user-configurable host computer workstations, a completely new definition of "user interface" can be established.

CHAPTER 5

DOCUMENTATION / APPENDICES

5.1. Memory and I/O Maps

The upper memory chip select line UCS* is currently mapped at:

UCS*: FF000 - FFFFFH

Using the 2732 EPROMs will require mapping of UCS* at FE000 - FFFFFH.
Using the 2764's will require mapping of UCS* at FC000 - FFFFFH.
Using the 27128's will require mapping of UCS* at F8000 - FFFFFH.

The middle memory chip select lines MCS<3,0>* are mapped at:

MCS0*: 00000 - 1FFFFH
MCS1*: 20000 - 3FFFFH
MCS2*: 40000 - 5FFFFH
MCS3*: 60000 - 7FFFFH

The lower memory chip select is not used.

The bottom 400H words of memory are reserved for use by the 80186.

The peripheral chip select lines are mapped as follows:

PCS0*: 000 - 07FH
PCS1*: 000 - 07FH
PCS2*: 100 - 17FH
PCS3*: 100 - 17FH
PCS4*: 200 - 27FH
PCS5*: 200 - 27FH
PCS6*: 300 - 37FH

The upper area of I/O space, from FF20 - FFFFH are reserved for use by the '186.

Programmable I/O lines of the 80186 are currently used as follows:

Interrupt lines:

INT0	Multibus data transfer interrupt.
INT1	Optional: dual serial I/O channel.
INT2/INTA0*	Available for use by peripherals.
INT3/INTA1*	(Future INTA for dual serial I/O channel.)

Counter/timer lines:

TMR0	Dual serial I/O baud rate clock generator.
TMR1	Available for use by peripherals.
TMR2	Refresh dynamic RAM through interrupts.

Peripheral chip selects:

PCS0*-PCS3* are initialized to operate with zero wait states.
- PCS0* Further decoded with LA<3,1> to provide 8 read and 8 write chip selects.
- PCS1* Not used.
- PCS2* Not used.
- PCS3* Not used.

PCS4*-PCS6* are initialized to operate with two wait states.
- PCS4* Dual serial I/O chip. LA<2,1> decoded internal to chip.
- PCS5* Not used.
- PCS6* Not used.

5.2. Boot Listing / Initialization Code

This section contains the 80186 assembly code representation of the current contents of the EPROMs. A thin vertical line "|" has comments to the right of it. The user is free to make additions to this code by programming new EPROM chips.

Eproms can be programmed in the electronics support shop, room 140 Cory Hall, or on any other prom programmer connected to the unix system. In 140 Cory, one must secure the personality module appropriate for the particular eprom chip being programmed, and then attach the unix connection and switch. With the switch in the "unix" position, login and execute a "blast" [arg], where arg is the previously compiled and assembled hex code that has been split into even and odd addresses. "Blast" can be found in /lc/cad/bin. "Blast" gives all necessary instructions.

```
| 186 ROM code for 2716 PROMS  (2K x 16 bits)

        .globl  _main
        .globl  _etext
        .globl  _edata
        .globl  _end
        .globl  _downloa

VAXCSR  = 0x0204
VAXDATA = 0x0200
CONCSR  = 0x0206
DARTIN      = 01
REFCNT  = 5000
        .text
|       Upon reset, the 80186 sets its UCS* line to 3 wait states and then
|       does a read at FFFF0H.  The first 4 OUTs initialize the '186's
|       programmable chip select lines.
|
start:                          | These next 3 instructions are done
                                | in upper memory.
|       mov   dx,#0xFFA0        | Set chip select registers:
|       mov   ax,#0xFF3F        | Upper Memory CS is 4k bytes for 2716,
|       outw                    | with 3 wait states. (Two waits @ 8 MHz.)
|
        mov   dx,#0xFFA8        | Middle Memory CS is 256K words for RAM,
```

```
mov   ax,#0xC0BE  | and I/O is I/O Mapped; 7 PCS lines &
                  | A1 A2 not latched.
outw              | Two wait states for PCS4-6. (One wait
                  | state @ 8MHz CPU operation.)

mov   dx,#0xFFA4  | Peripheral CS is I/O Mapped at 0.
mov   ax,#0x003C  | 0 wait for PCS0-3.
outw

mov   dx,#0xFFA6  | Offset for Middle Memory CS is 0.
mov   ax,#0x01F8  | No wait states for Middle Memory CS.
outw
```

The next three OUTs initialize the '186's internal counter/timer channel 0, used as a baud rate clock generator.

```
mov   dx,#0xFF52  | Set up timer 0 for 156.25 kHz, 50% duty
mov   ax,*8       | cycle. Used for 16 * baud rate.
outw
mov   dx,#0xFF54
outw
mov   dx,#0xFF56
mov   ax,#0xC003
outw

nop; nop; nop; nop; | wait a while
```

The following code initializes the Intel 8274 dual serial I/O chip, or "DART".

```
mov   dx,#VAXCSR  | Initialize DART CHIP channel A.
mov   al,#0x04
out
mov   al,#0x48    | Use x16 clock, 1.5 stop bits, no parity.
out
mov   al,#0x01
out
mov   al,#0x00    | Disable waits, interrupts disabled.
out
mov   al,#0x05
out
mov   al,#0x28    | Transmit 7 bits/char, enable transmit.
out
mov   al,#0x03
out
mov   al,#0x41    | Receive 7 bits/char, enable receiver.
out
mov   al,#0x02
out
mov   al,#0x00    |
out

mov   dx,#CONCSR  | Initialize DART CHIP channel B.
mov   al,#0x04
```

```
out
mov   al,#0x48    | Use x16 clock, 1.5 stop bits, no parity.
out
mov   al,#0x01
out
mov   al,#0x00    | Disable waits, interrupts disabled.
out
mov   al,#0x05
out
mov   al,#0x28    | Transmit 7 bits/char, enable transmit.
out
mov   al,#0x03
out
mov   al,#0x41    | Receive 7 bits/char, enable receiver.
out
```

End of DART initialization.

Set up top of stack at top of memory.

```
mov   ax,#0x040   | Stack and data segments start of 0x00400
mov   ss,ax
mov   ds,ax
mov   sp,#0xFFFE  | Stack is at top
mov   ax,#0       | Extra segment is at 0
mov   es,ax
```

This initializes the '186's internal counter/timer channel 2
and internal interrupt controller to serve as a dynamic
RAM refresh machine.

```
mov   bx,#0x004C  | Set up refresh interrupt vector from timer 2
mov   ax,#ref_int | to jump to PROM refresh routine
mov   es:0(bx),ax
mov   ax,cs
mov   es:*2(bx),ax
mov   dx,#0xFF32  | Initialize timer 2 interrupt control register
mov   ax,#0x0000  | to priority 0, un-masked.
outw
mov   dx,#0xFF62  | Set up timer 2 to interrupt after counting
mov   ax,#REFCNT  | 5000 internal events (2.5 MHz clock pulses)
outw              | thus interrupting every 2 mS to refresh
                  | the DRAM.
mov   dx,#0xFF66  | Set timer 2 mode word to stop and interrupt
mov   ax,#0xE000  | after max count
outw
```

End of refresh timer/interrupt initialization.

```
mov   ax,ds       | copy C data from PROM to RAM
mov   es,ax
mov   cx,#_data-_test+1
repz              | string move
```

```
        seg     es              | use code segment
        movb
        sti                     | start refresh

        call    _main           | call the C Main program.
        jmp     start
|
|
| Refresh Interrupt Routine
|
ref_int:
        push    dx
        push    ax

        mov     dx,#0xFF66      | Restart counter/timer 2 by writing E000 to
        mov     ax,#0xE000      | FF66H.
        outw
|
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
        nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
|
        mov     dx,#0xFF22      | Write end-of-interrupt to interrupt
        mov     ax,#0x8000      | controller.
        outw
        pop     ax              | pop and return
        pop     dx
        iret
|
|
| Download routine, does not use any RAM except to put downloaded data.
|
_downloa:
        cli
        mov     dx,#VAXCSR      | reprogram channel A for 8 bit data
        mov     al,#0x05
        out
        mov     al,#0x68        | Transmit 8 bits/char.
        out
        mov     al,#0x03
```

```
        out
        mov   al,#0xC1      | Receive 8 bits/char.
        out
        sti
srchS:                      | search for S
        mov   dx,#VAXCSR
t2:     inw
        and   ax,#DARTIN
        jz    t2
        mov   dx,#VAXDATA
        inw
        and   ax,#0x7F
        cmp   ax,*83
        bne   srchS
        mov   dx,#VAXCSR    | get lsb of length
t1:     inw
        and   ax,#DARTIN
        jz    t1
        mov   dx,#VAXDATA
        inw
        and   ax,#255
        mov   si,ax
        mov   dx,#VAXCSR    | get msb of length
t3:     inw
        and   ax,#DARTIN
        jz    t3
        mov   dx,#VAXDATA
        inw
        mov   cx,*8
        sal   ax,cl
        add   si,ax
        mov   di,*0
        mov   cx,*0         | clear checksum
        mov   ax,#0x0040    | download into memory starting at 0x0400
        mov   ds,ax
dldloop:                    | move in the data
        mov   ax,si
        dec   si
        or    ax,ax
        beq   dlddone             | check if done (len = 0)
        mov   dx,#VAXCSR
t4:     inw
        and   ax,#DARTIN
        jz    t4
        mov   dx,#VAXDATA
        inw
        mov   bx,di
        inc   di
        movb  (bx),ax
        add   cx,ax         | for checksum
        br    dldloop
dlddone:
        mov   dx,#VAXCSR
```

```
t5:     inw
        and     ax,#DARTIN
        jz      t5
        mov     dx,#VAXDATA
        inw
        cmpb    cx,ax
|       jnz     start           | if checksum doesn't match then restart
        jmpi    0,0x0040        | jump to new code at 0x00400

|
|       at FFFF0 put a long jump to FF000 (code segment is FF00, address 0)
|       this is put in by 86makeproms
```

5.3. Signal Descriptions

AD<15,0>
: Multiplexed address and data bus from '186. To DRAM, EPROM, and BAD 8-bit transceivers. Low current drive bus, not intended for use by additional peripherals.

BAD<19,0>
: Buffered multiplexed address and data bus, from two 8-bit transceivers from AD<15,0>. To LA bus latches, DART's data transceiver, multibus data port, and multibus control/status register. Transceivers are disabled during read of EPROM or DRAM. Transceiver direction controlled by BRD*. High current drive bus, intended for additional peripherals.

A<19,16>
: Address bus from '186. To 4-bit bus buffer.

BA<19,16>
: Buffered address bus from 4-bit bus buffer from A<19,16>. To MA bus latches. Always enabled.

LA<9,0>
: Latched address bus from one 10-bit latch. To EPROM, DART, 3-8 address decoders, MA bus buffer, and low byte enable for DRAM. Valid while ALE low, and while ALE high and address valid on BAD bus.

MA<9,1>
: Multiplexed address bus. From 10-bit latch from BAD bus and 10-bit buffer from LA bus. To DRAM and EPROM. MA<9,1>=LA<9,1> circa RAS fall; MA<7,2>=BAD<15,10>, MA<8>=BA<16>, and MA<9>=BA<18> circa CAS fall. MA<1>=BAD<9> circa CAS fall for 64k x 1 DRAM chips; MA<1>=BA<17> circa CAS fall for 256k X 1 chips.

BD<7,0>
: Buffered data bus, from 8-bit transceiver from BAD<7,0>, to DART only. Direction of transceiver controlled by DT/R* signal, and transceiver enabled by PCS4* conditioned with DEN*.

ADR<13H,0>*
: Multibus address bus, 20 bits wide. (<13H,0> is in hex.) ADR<13H,BH>* and ADR<1>* to address decoder consisting of 8-bit comparator and 3-8 decoder. ADR<6>* to reset decoder.

DAT<F,0>
Multibus data bus. To 16-bit data port, and 6-bit control/status port.

BRD*
Buffered RD*. Tri-state buffer, always enabled. To: 3-8 chip decoder; DART's RD*; EPROM's OE*; in conjunction with BWR* forms RAS*; direction control of BAD bus transceiver; and in conjunction with MEM* forms enable control RDMEM to BAD bus transceiver.

BWR*
Buffered WR*. Tri-state buffer, always enabled. To 3-8 chip decoder, DART's WR*, DRAM's WE*, and in conjunction with BRD* forms RAS*.

RAS*
Row address strobe for DRAM. Active if either BWR* or BRD* are active. To 16 DRAMs and DDL.

MUX*
Multiplexer signal, enables high or low address onto MA bus. When active, MA has high address, preparing for CAS* to strobe it in. Delayed version of RAS*, by 20nS. To output enable of high address latch/tri-state.

MUX
Inverted MUX*. To output enable of low address tri-state. When active, low address on MA bus, ready for RAS* to strobe address in.

CAS*
Column address strobe for DRAM. Delayed version of RAS*, by 40nS, from DDL.

WE*
Write enable of DRAMs.

ALE
'186 address latch enable. When low, latches address from BAD in LA and MA bus latches.

BHE*
'186 byte high enable. With CAS* forms HICAS*, enabling high byte of dynamic RAM memory.

PCS0*
'186 peripheral chip select 0. In conjunction with LA<3,1> and BRD* or BWR* generates decoded read and write signals for multibus data port and multibus control/status port.

PCS4*

'186 peripheral chip select 4. To DART's CS*; in conjunction with '186 DEN* becomes enable for DART's data transceiver (DEN4*).

LCS*

'186 lower memory chip select. Not used.

UCS*

'186 upper memory chip select. To EPROM's CE*, and in conjunction with MCS* and BRD* becomes enable control for BAD bus transceivers.

MCS*

Currently equals '186 MCS0*. When 256k x 1 DRAM chips available, MCS*=MCS0* x MCS1* x MCS2* x MCS3*. In conjunction with CAS* becomes MCAS*; in conjunction with UCS* and BRD* becomes enable control for BAD bus transceivers.

INT0

'186 interrupt input 0. From multibus interface hardware, indicating that the multibus data port has been read out or has new data in, depending on programmable mask bits.

INT1

'186 interrupt input 1. From inverted DARTINT* of DART chip. Programmable to be active on a certain set of conditions.

INT2/INTA0*

'186 interrupt input 2. Available for use by peripherals.

INT3/INTA1*

'186 interrupt input 3. Available for use by peripherals, or can be used as INTA to DART chip.

MRDC*

Multibus memory read control. To multibus 3 to 8 address/read/write decoder and to XACK* generation circuit.

MWTC*

Multibus memory write control. To multibus 3 to 8 address/read/write decoder and to XACK* generation circuit.

BA*

Base address of multibus address decoding. Output of eight-bit comparator, to decode multibus address bus and enable the multibus interface. In conjunction with MC* forms ENX*. Enables 3-to-8 decoder to form DATRD*, DATWR*, CSRRD*, and CSRWR*.

MC*

Multibus-memory access signal. Active if either MWTC* or MRDC* are active.

ENX*

Enable XACK* tri-state signal. Active if both BA* and MC* are active, indicating a multibus memory access to the decoded address space.

XIN*

Delayed version of ENX*, to drive input of XACK* tri-state buffer. Delay timing is chosen so that XACK* falls after data from data port is on multibus.

XACK*

Multibus data acknowledge. Tri-state. Enabled by ENX*, driven by XIN*.

DATWR*

Multibus data port write decoded strobe. Active if BA* and MWTC* and ADR1* active, by a 3-8 decoder. Strobes CPR line on multibus data port to write data in from multibus.

DATRD*

Multibus data port read decoded strobe. Active if BA* and DR1* and ADR1* active, by a 3-8 decoder. Pulls OEAS* line low on multibus data port to read word to multibus.

CSRWR*

Multibus control port write decoded strobe. Active if BA* and MWTC* and ADR1 active, by a 3-8 decoder. Strobes clock line on multibus control/status port to write data from multibus.

CSRRD*

Multibus control port read decoded strobe. Active if BA* and DR1* and ADR1 active, by a 3-8 decoder. Pulls OE* low on multibus control/status port to read to multibus.

INT4*

Multibus interrupt line 4.
Tri-state buffer, enabled by INT4REQ*,
with input tied low, as an O.C. equivalent.

MBDRD*

Decoded read strobe of I/O port 0004H.
Active if PCS0* and BRD* and LA<1>* and LA<2>
and LA<3>* active. Pulls OEBR* line low on multibus
data port for read of word to '186.

MBDWR*

Decoded write strobe of I/O port 0004H.
Active if PCS0* and BWR* and LA<1>* and LA<2>
and LA<3>* active. Strobes CPR line of multibus
data port for write of word from '186.

MBCRD*

Decoded read strobe of I/O port 0006H.
Active if PCS0* and BRD* and LA<1> and LA<2>
and LA<3>* active. Pulls OE* line low on multibus
control/status port for read of data to '186.

MBCWR*

Decoded write strobe of I/O port 0006H.
Active if PCS0* and BWR* and LA<1> and LA<2>
and LA<3>* active. Strobes clock line of multibus
control/status port for write of data from '186.

RES*

Reset signal to '186. If on-board push-button
is pressed, or if multibus INIT* line becomes
active, or if BA* and ADR6* are active, then
RES* is active.

RESET

Active high reset signal from '186. To
inverter to create SYSRESET*.

SYSRESET*

Active low reset from inverter from RESET.
To DART, and multibus control/status port.

RD*

'186 read signal. To tri-state buffer.

WR*

'186 write signal. To tri-state buffer.

CLKOUT

From '186. Input clock's or crystal's
frequency divided by 2. Currently equals
10 MHz. To 4-bit counter.

DARTCK
: From 4-bit counter, that divides CLKOUT by 2, 4, 8, or 16 continuously on 4 output pins. The resulting 2.5 MHz clock is fed to the DART. Other taps available for peripherals.

TMR 0 IN
: '186 internal counter/timer unit's timer 0 input. Tied high, this timer used as a baud rate generator.

TMR 0 OUT
: Output of '186's timer 0. Used as a baud rate clock for DART.

TMR 1 IN
: '186 internal counter/timer unit's timer 1 input. Available for use by peripherals.

TMR 1 OUT
: Output of '186's timer 1. Available for use by peripherals.

NMI
: '186 non-maskable interrupt input. Tied low, not used.

HOLD
: '186 hold input. Tied low, not used.

HLDA
: '186 hold acknowledge output. Not used.

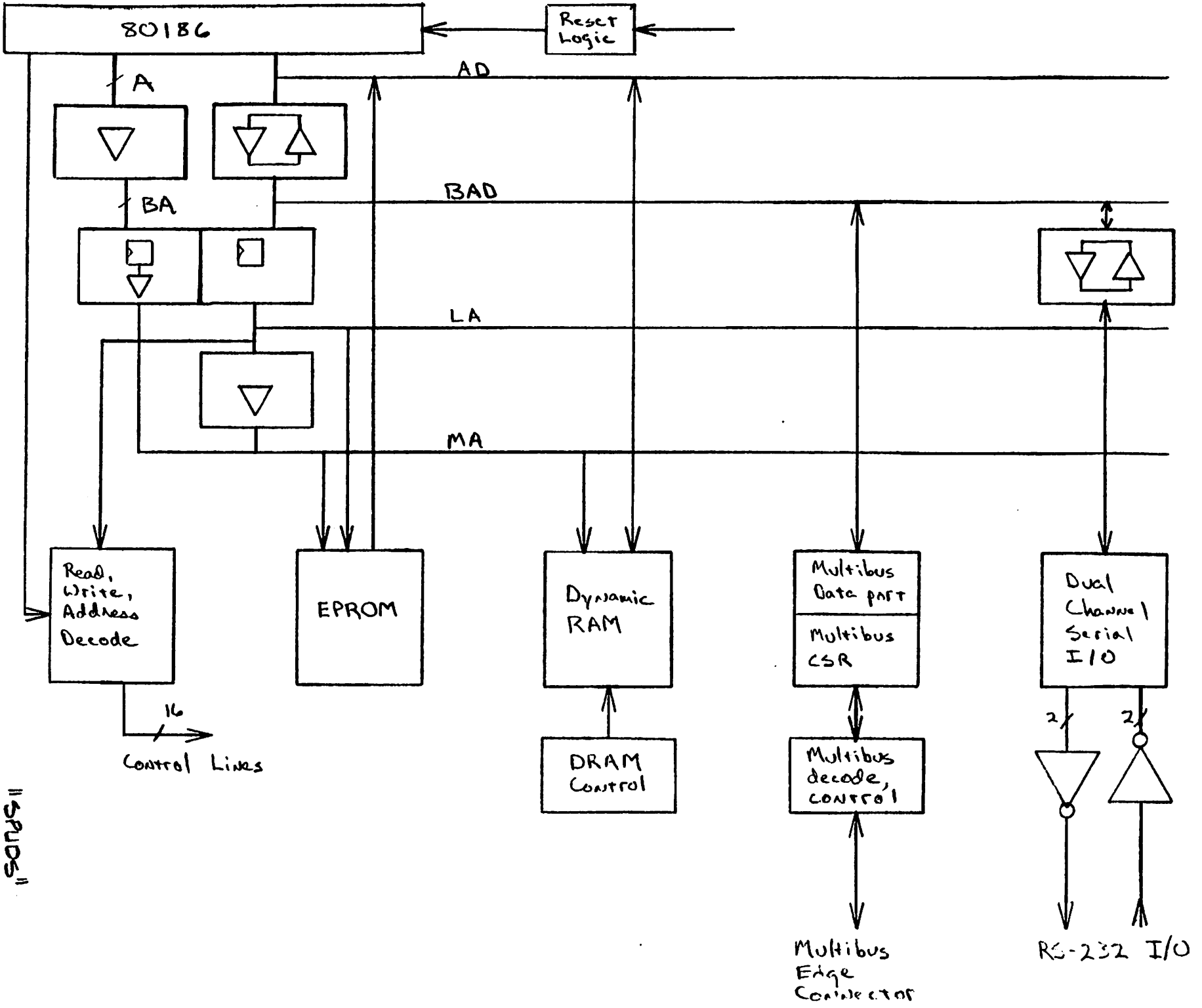
TEST*
: '186 test input. Tied low, not used.

SRDY
: '186 synchronous ready input. Tied low, not used.

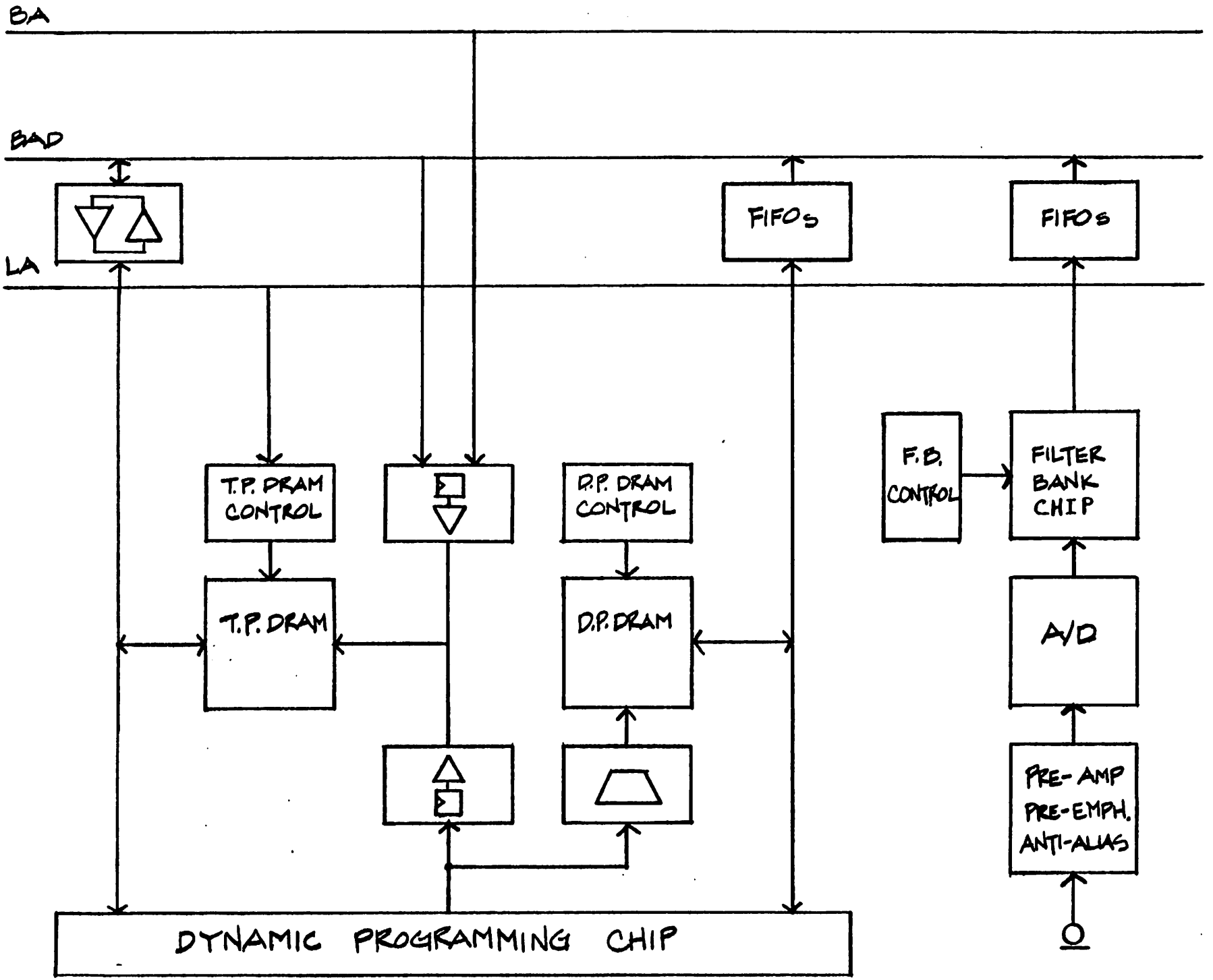
ARDY
: '186 asynchronous ready input. Tied high, not used.

LOCK
: '186 lock input. Tied low, not used.

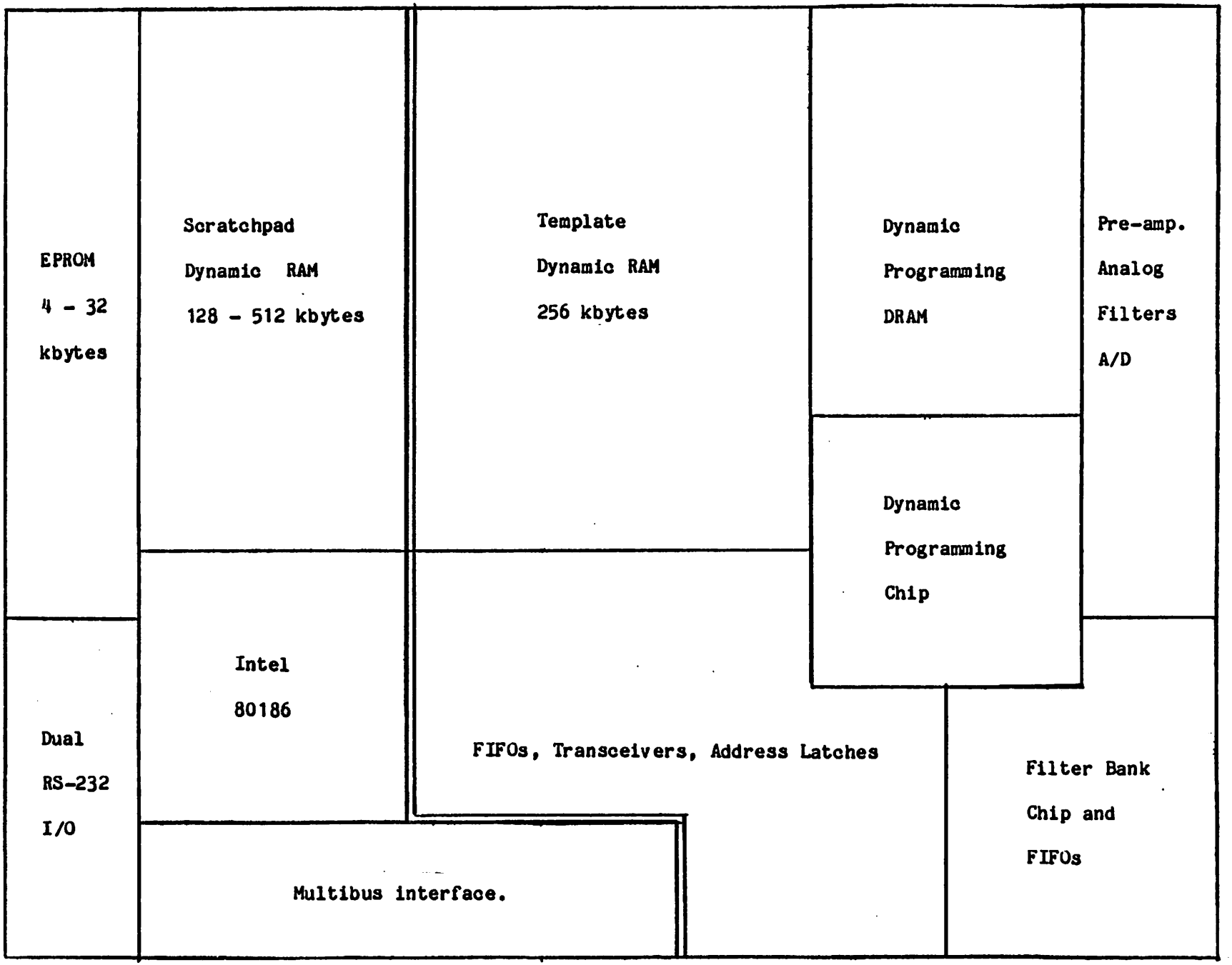
S0*-S2*
: '186 status outputs. Not used.



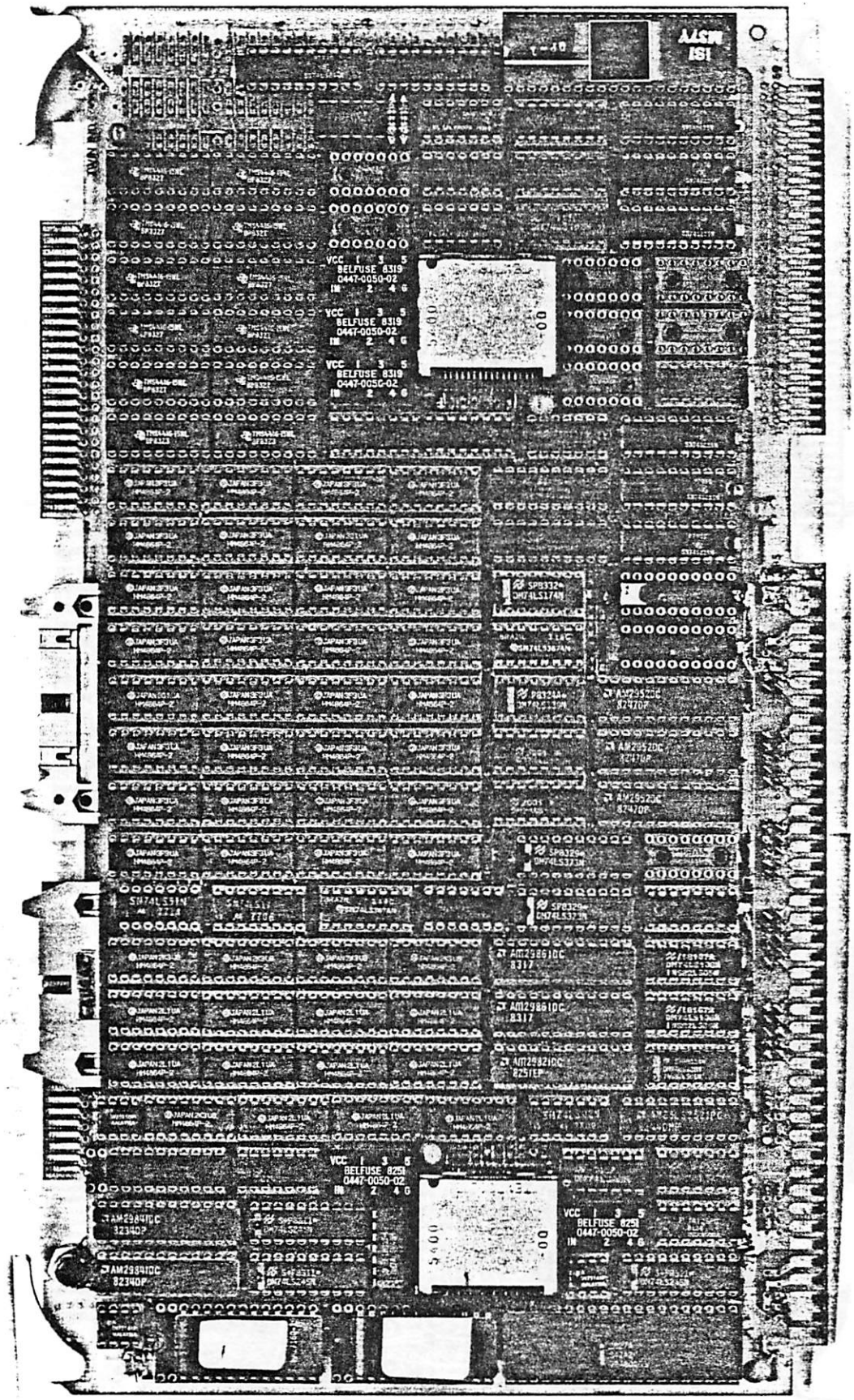
"SPUDS" BLOCK DIAGRAM

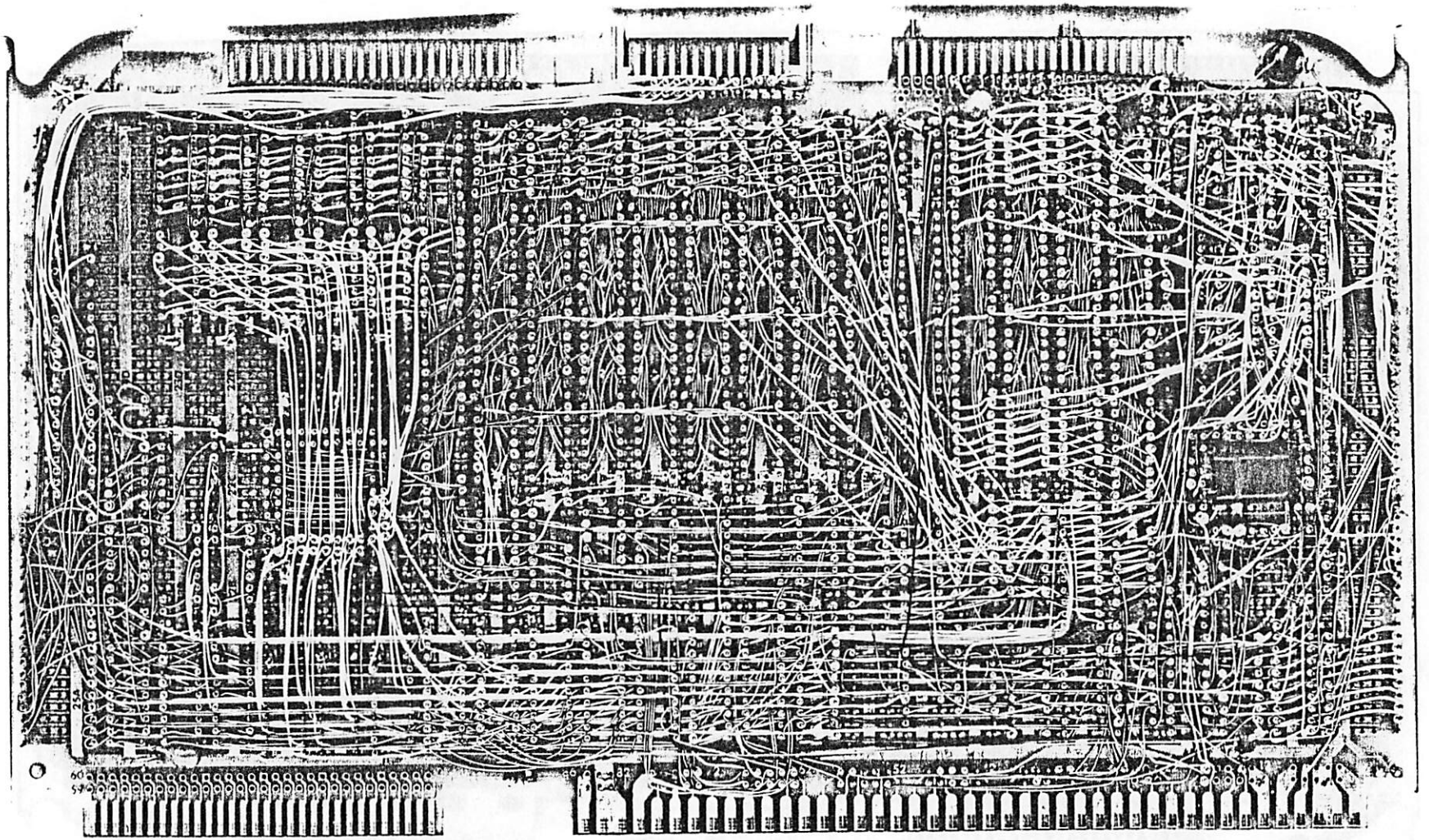


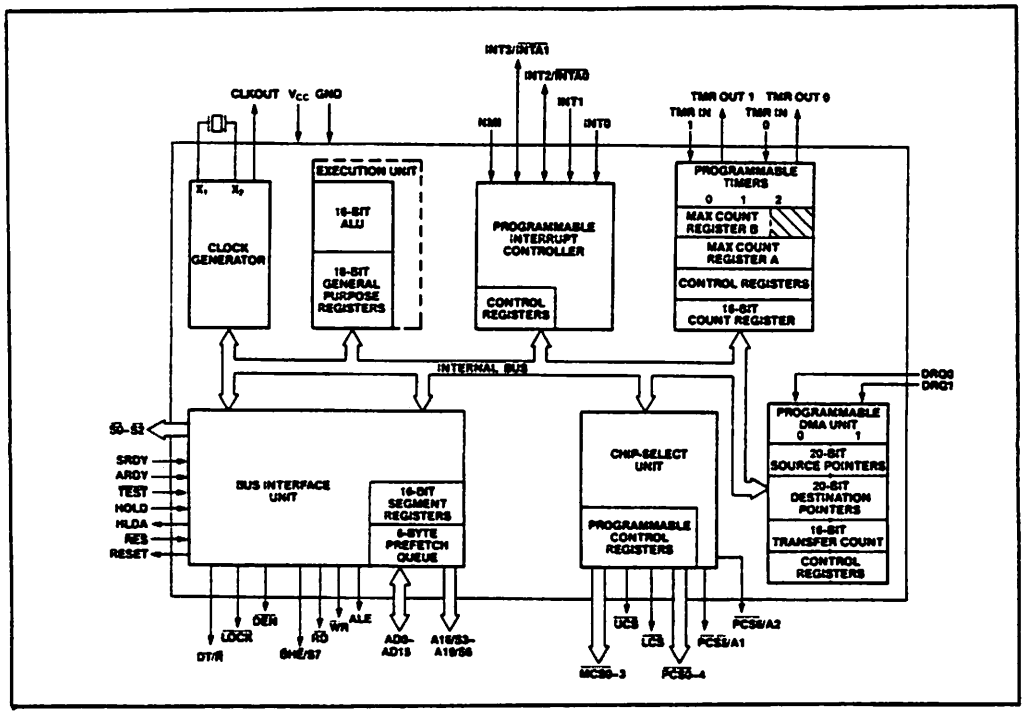
BLOCK DIAGRAM OF SPEECH RECOGNITION SYSTEM



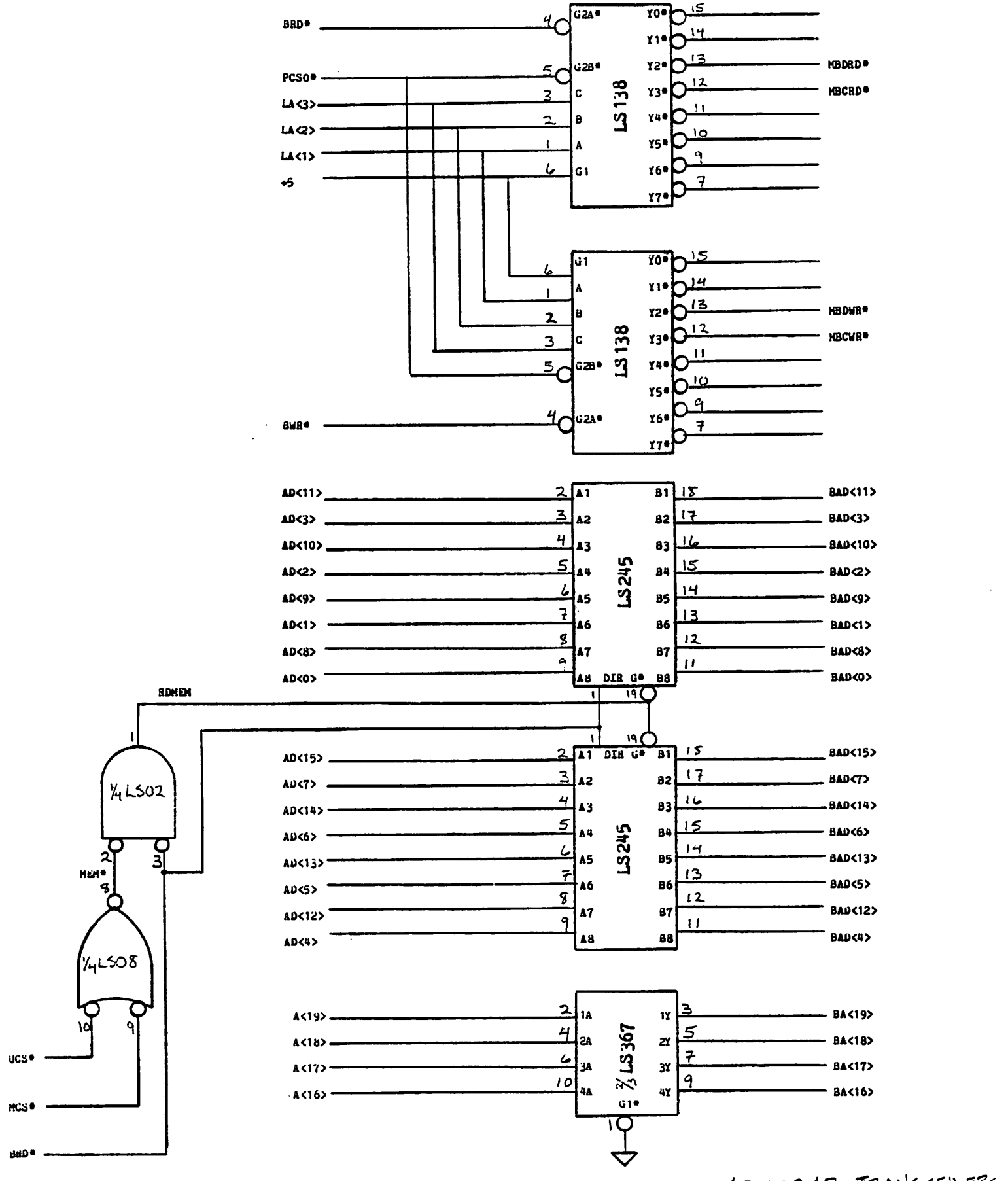
LAYOUT OF COMPLETE SPEECH RECOGNITION SYSTEM

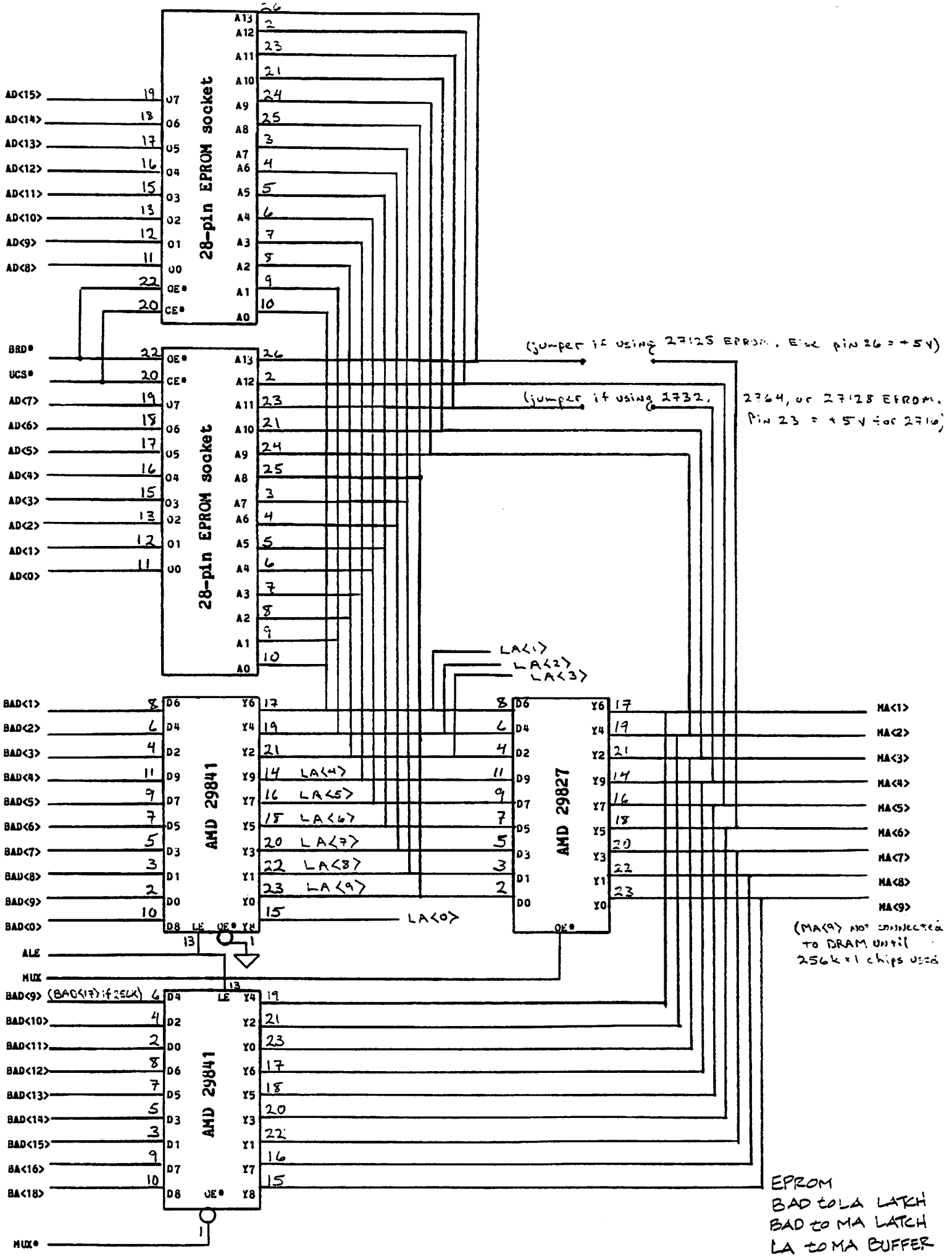

VCC 1 3 5
BELFUSE 8319
0447-0050-02
IN 2 4 6
VCC 1 3 5
BELFUSE 8319
0447-0050-02
IN 2 4 6
VCC 1 3 5
BELFUSE 8319
0447-0050-02
IN 2 4 6
VCC 1 3 5
BELFUSE 8259
0447-0050-02
IN 2 4 6
VCC 1 3 5
BELFUSE 8259
0447-0050-02
IN 2 4 6

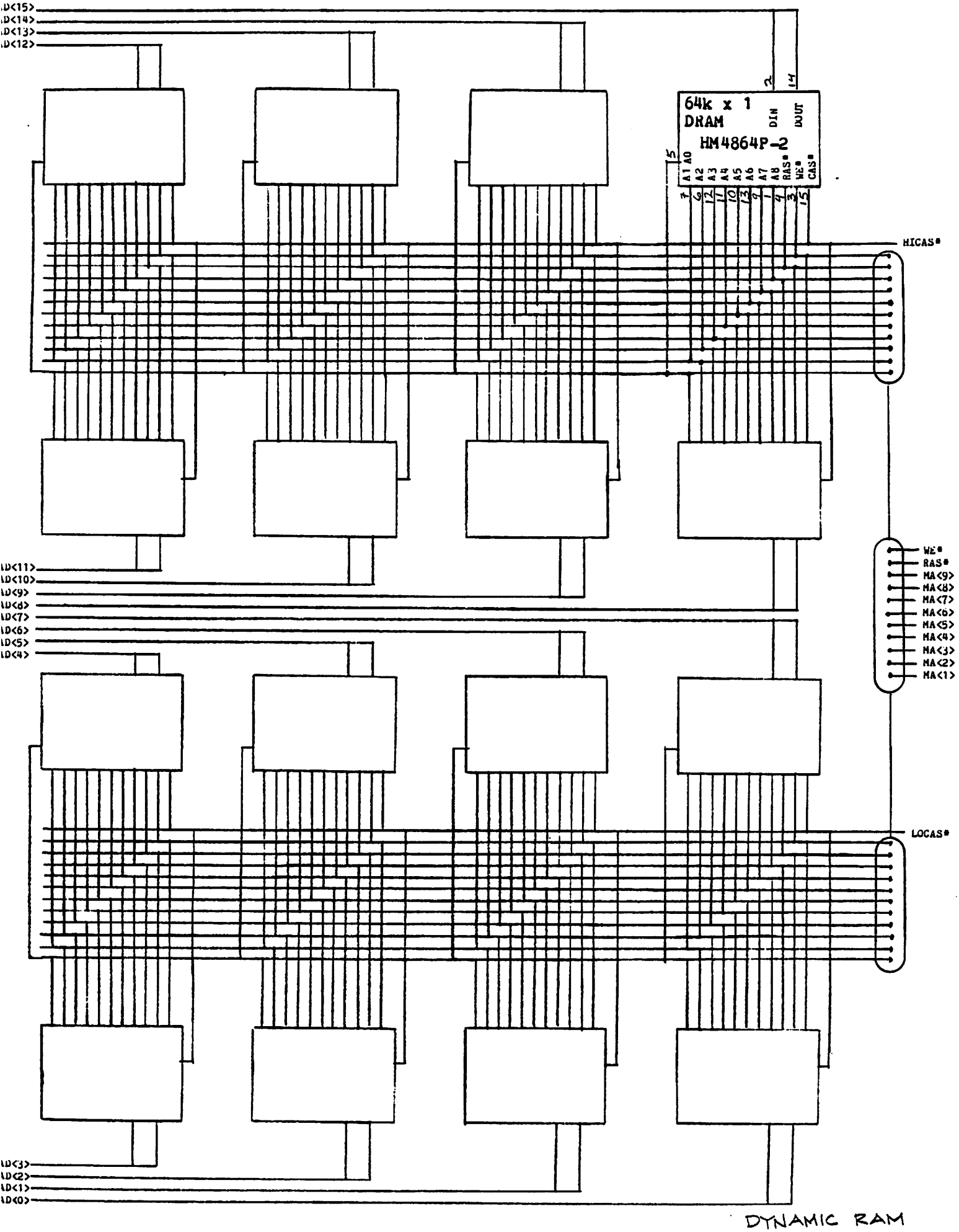



INT3/INTA1
INT2/INTA0
INT1
INT0
NMI
CLKOUT
VCC
GND
TMR OUT 1
TMR OUT 0
TMR IN 1
TMR IN 0
EXECUTION UNIT
16-BIT ALU
16-BIT GENERAL PURPOSE REGISTERS
CLOCK GENERATOR
PROGRAMMABLE INTERRUPT CONTROLLER
CONTROL REGISTERS
PROGRAMMABLE TIMERS
0 1 2
MAX COUNT REGISTER B
MAX COUNT REGISTER A
CONTROL REGISTERS
16-BIT COUNT REGISTER
INTERNAL BUS
DRQ0
DRQ1
PROGRAMMABLE DMA UNIT
0 1
20-BIT SOURCE POINTERS
20-BIT DESTINATION POINTERS
16-BIT TRANSFER COUNT
CONTROL REGISTERS
S0-S2
SRDY
ARDY
TEST
HOLD
HLDA
RES
RESET
BUS INTERFACE UNIT
16-BIT SEGMENT REGISTERS
6-BYTE PREFETCH QUEUE
CHIP-SELECT UNIT
PROGRAMMABLE CONTROL REGISTERS
DT/R
DEN
LOCK
RD
WR
ALE
BHE/S7
AD0-AD15
A16/S3-A19/S6
UCS
LCS
MCS0-3
PCS0-4
PCS5/A2
PCS6/A1

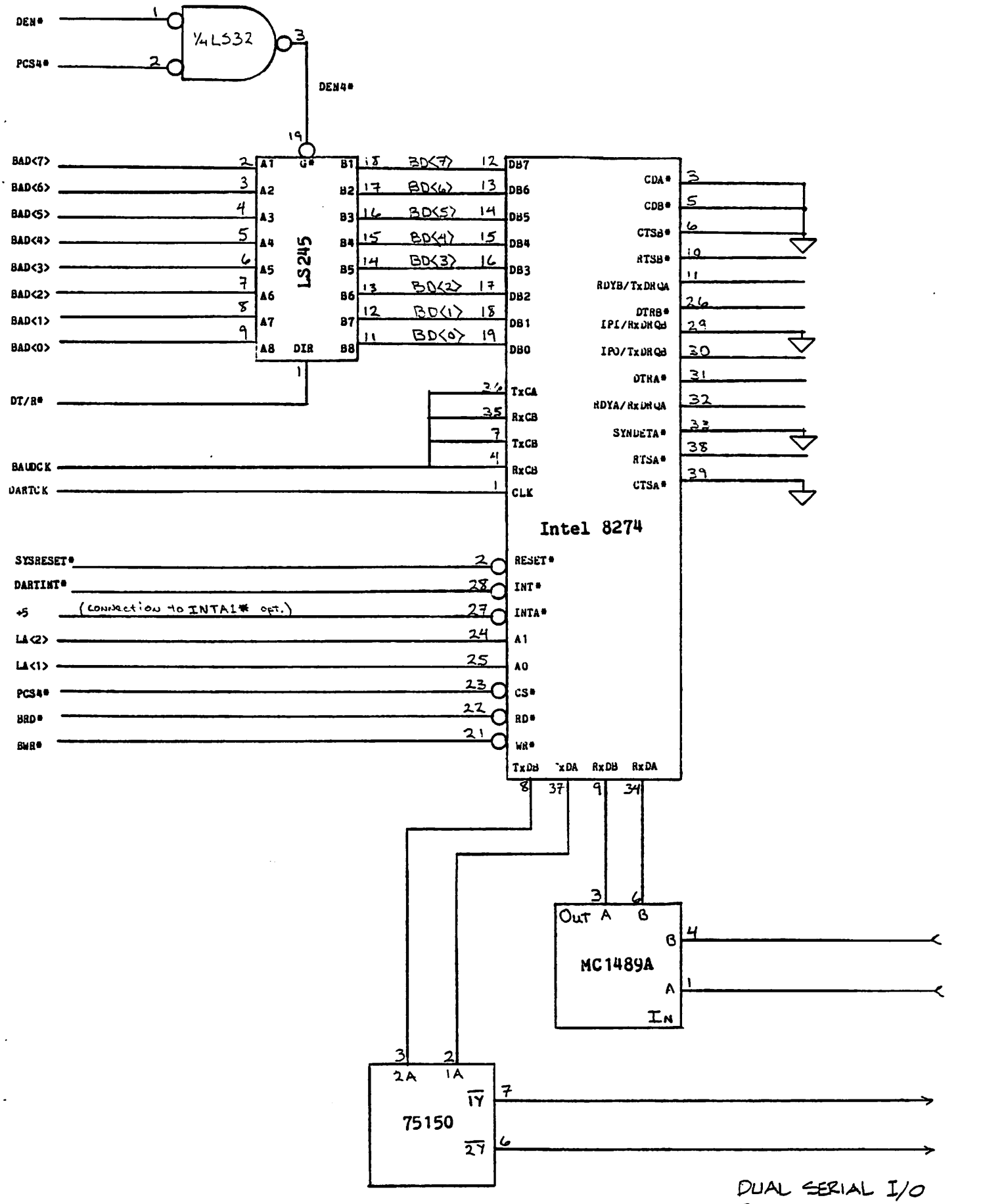
INTEL 80186


BRD*
PCS0*
LA<3>
LA<2>
LA<1>
+5
G2A*
G2B*
C
B
A
G1
LS138
Y0*
Y1*
Y2*
Y3*
Y4*
Y5*
Y6*
Y7*
MBDRD*
MBCRD*
BWR*
MBDWR*
MBCWR*
AD<11>
AD<3>
AD<10>
AD<2>
AD<9>
AD<1>
AD<8>
AD<0>
A1
A2
A3
A4
A5
A6
A7
A8
DIR
G*
B1
B2
B3
B4
B5
B6
B7
B8
LS245
BAD<11>
BAD<3>
BAD<10>
BAD<2>
BAD<9>
BAD<1>
BAD<8>
BAD<0>
RDMEM
AD<15>
AD<7>
AD<14>
AD<6>
AD<13>
AD<5>
AD<12>
AD<4>
BAD<15>
BAD<7>
BAD<14>
BAD<6>
BAD<13>
BAD<5>
BAD<12>
BAD<4>
1/4 LS02
MEM*
1/4 LS08
UCS*
MCS*
BRD*
A<19>
A<18>
A<17>
A<16>
1A
2A
3A
4A
1Y
2Y
3Y
4Y
G1*
2/3 LS367
BA<19>
BA<18>
BA<17>
BA<16>
AD TO BAD TRANSCEIVERS
DECODED STROBES

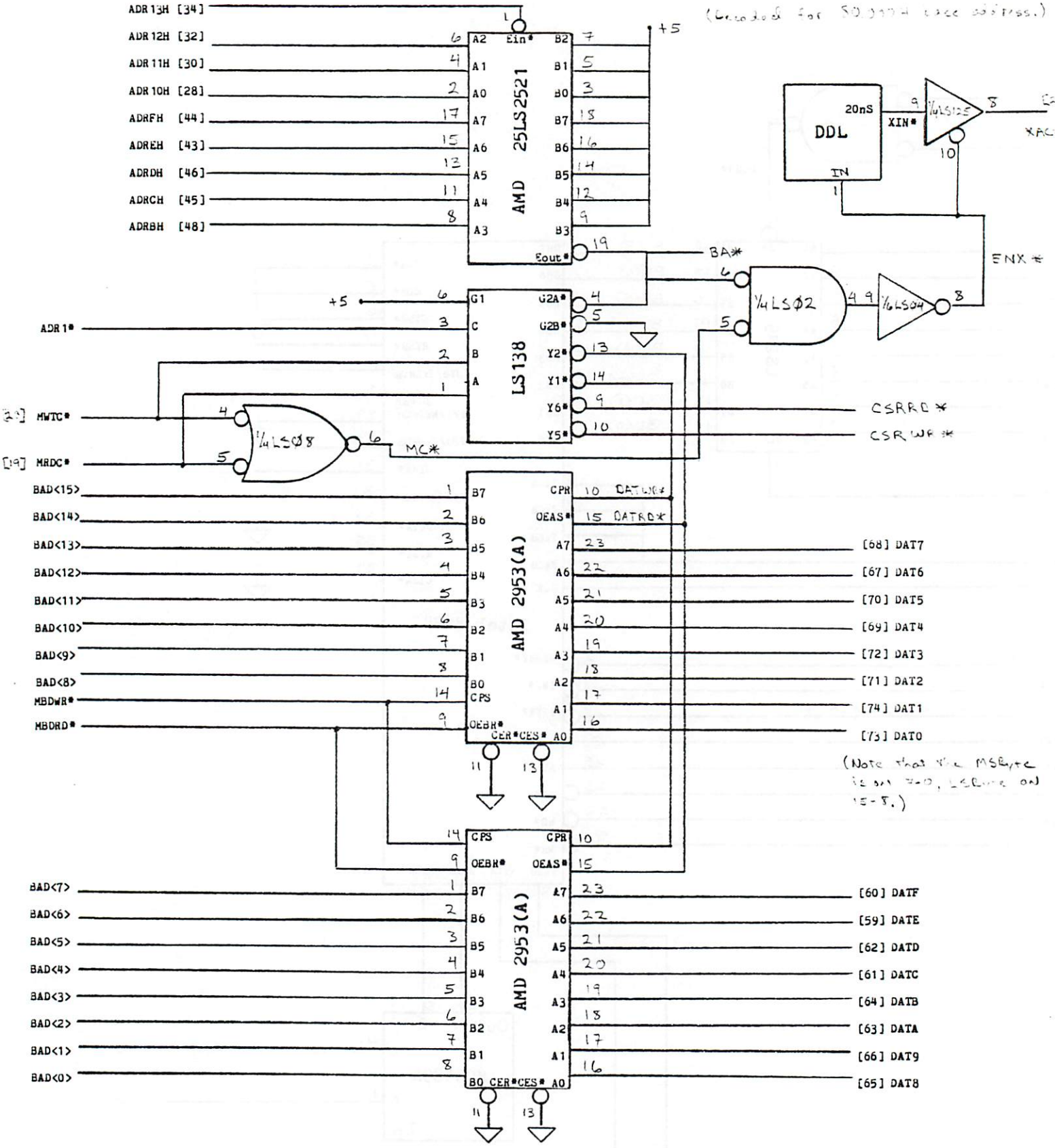

28-pin EPROM socket
28-pin EPROM socket
AMD 29841
AMD 29827
AMD 29841
AD<15>
AD<14>
AD<13>
AD<12>
AD<11>
AD<10>
AD<9>
AD<8>
AD<7>
AD<6>
AD<5>
AD<4>
AD<3>
AD<2>
AD<1>
AD<0>
BRD*
UCS*
(jumper if using 27125 EPROM. Else pin 26 = +5V)
(jumper if using 2732, 2764, or 27128 EPROM. Pin 23 = +5V for 2716)
LA<1>
LA<2>
LA<3>
LA<4>
LA<5>
LA<6>
LA<7>
LA<8>
LA<9>
LA<0>
BAD<1>
BAD<2>
BAD<3>
BAD<4>
BAD<5>
BAD<6>
BAD<7>
BAD<8>
BAD<9>
BAD<0>
MA<1>
MA<2>
MA<3>
MA<4>
MA<5>
MA<6>
MA<7>
MA<8>
MA<9>
(MA<9> not connected to DRAM until 256k x 1 chips used)
ALE
MUX
BAD<9> (BAD<17> if 256K)
BAD<10>
BAD<11>
BAD<12>
BAD<13>
BAD<14>
BAD<15>
BA<16>
BA<18>
MUX*
EPROM
BAD to LA LATCH
BAD to MA LATCH
LA to MA BUFFER


64k x 1
DRAM
HM4864P-2
DIN
DOUT
A1 A0
A2
A3
A4
A5
A6
A7
A8
RAS*
WE*
CAS*
HICAS*
LOCAS*
WE*
RAS*
MA<9>
MA<8>
MA<7>
MA<6>
MA<5>
MA<4>
MA<3>
MA<2>
MA<1>
MD<15>
MD<14>
MD<13>
MD<12>
MD<11>
MD<10>
MD<9>
MD<8>
MD<7>
MD<6>
MD<5>
MD<4>
MD<3>
MD<2>
MD<1>
MD<0>

DYNAMIC RAM



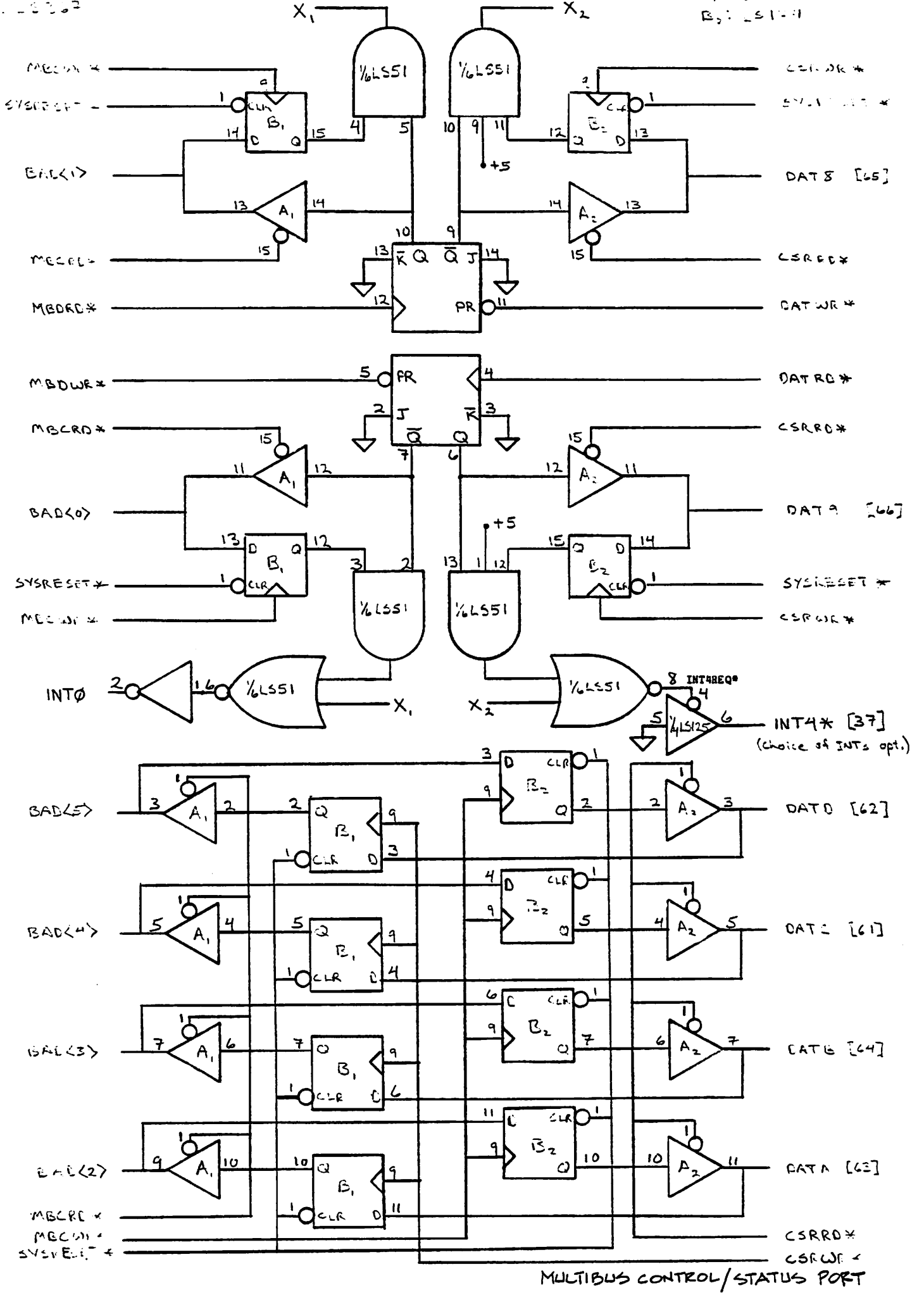
DUAL SERIAL I/O
RS-232 BUFFERS
BAD to BD TRANSCEIVER

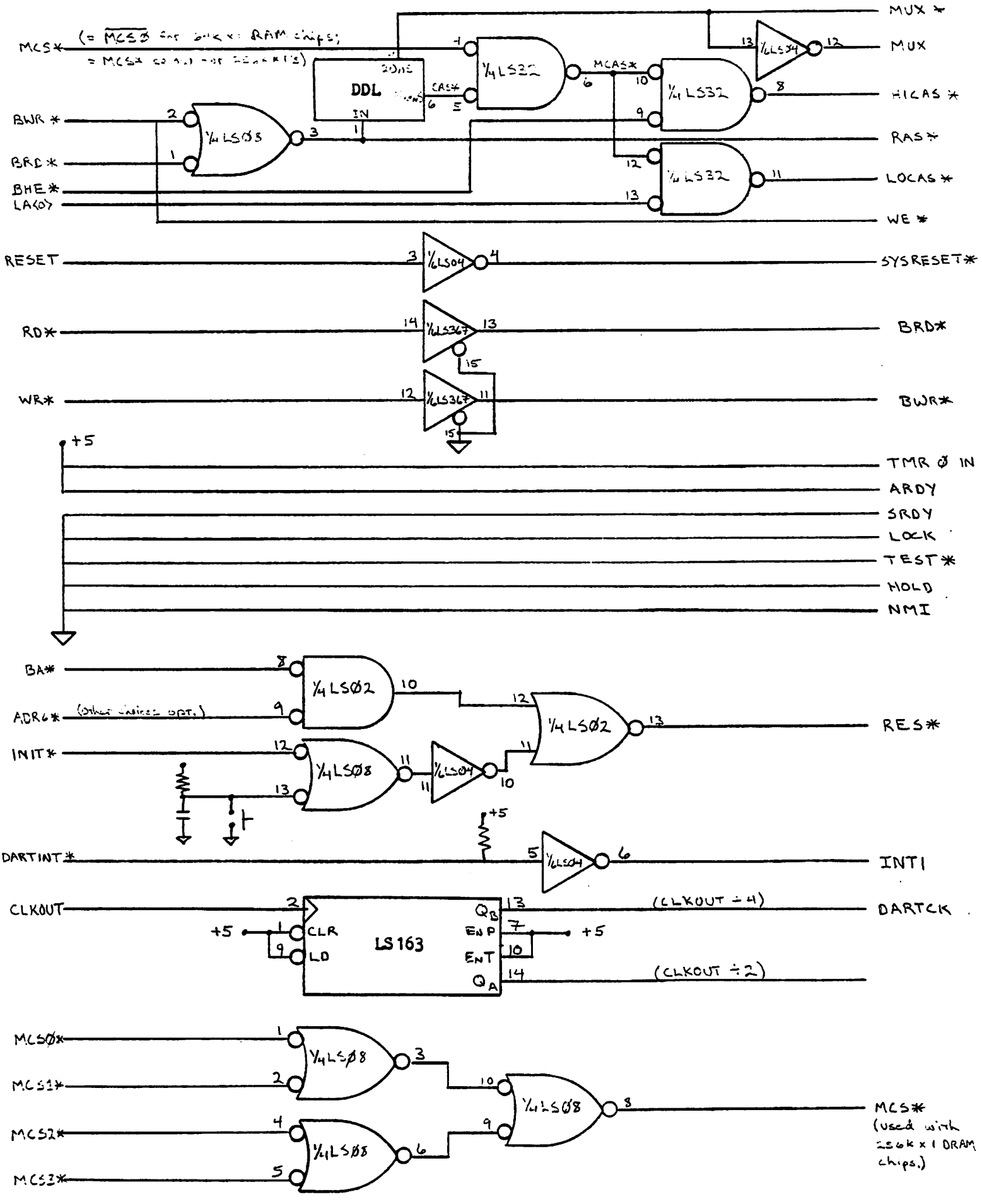


(Decoded for 50 word I/O space address.)

(Note that the MSByte is on D7-D0, LSByte on 15-8.)

MULTIBUS ADDRESS DECODE
DATA PORT
XACK* GENERATION

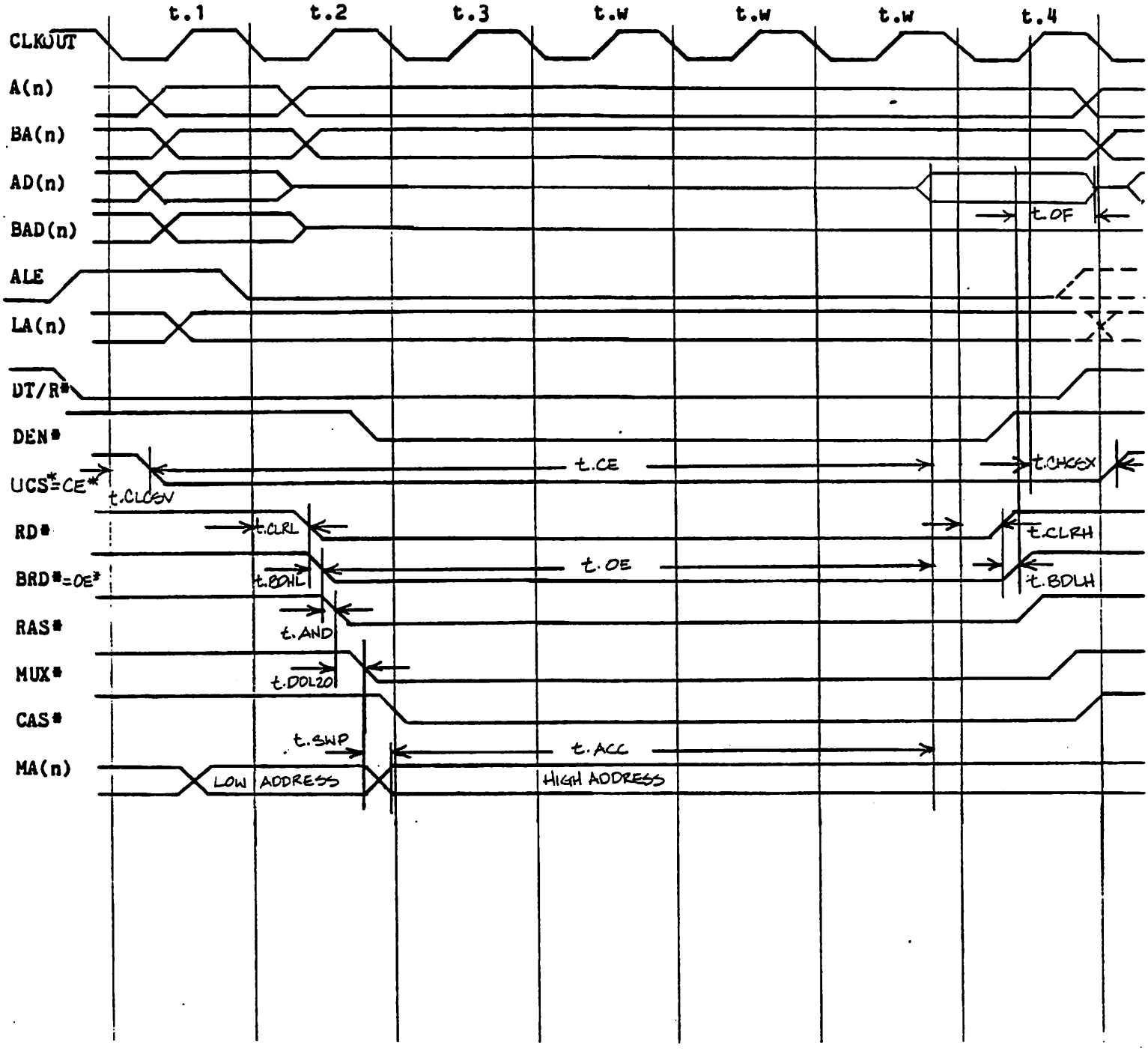

X1
X2
1/6LS51
+5
MBCLR*
SYSRESET*
BAD<1>
MBCRD*
MBDWR*
DAT8 [65]
CSRRD*
DAT WR*
MBDRD*
DAT RD*
BAD<0>
DAT9 [66]
SYSRESET*
INT0
INT4*
INT4REQ*
(choice of INTs opt.)
BAD<5>
DAT0 [62]
BAD<4>
DAT1 [61]
BAD<3>
DAT6 [64]
BAD<2>
DATA [63]
CSRWR*
MULTIBUS CONTROL/STATUS PORT



RANDOM LOGIC

5.6. Timing Diagrams

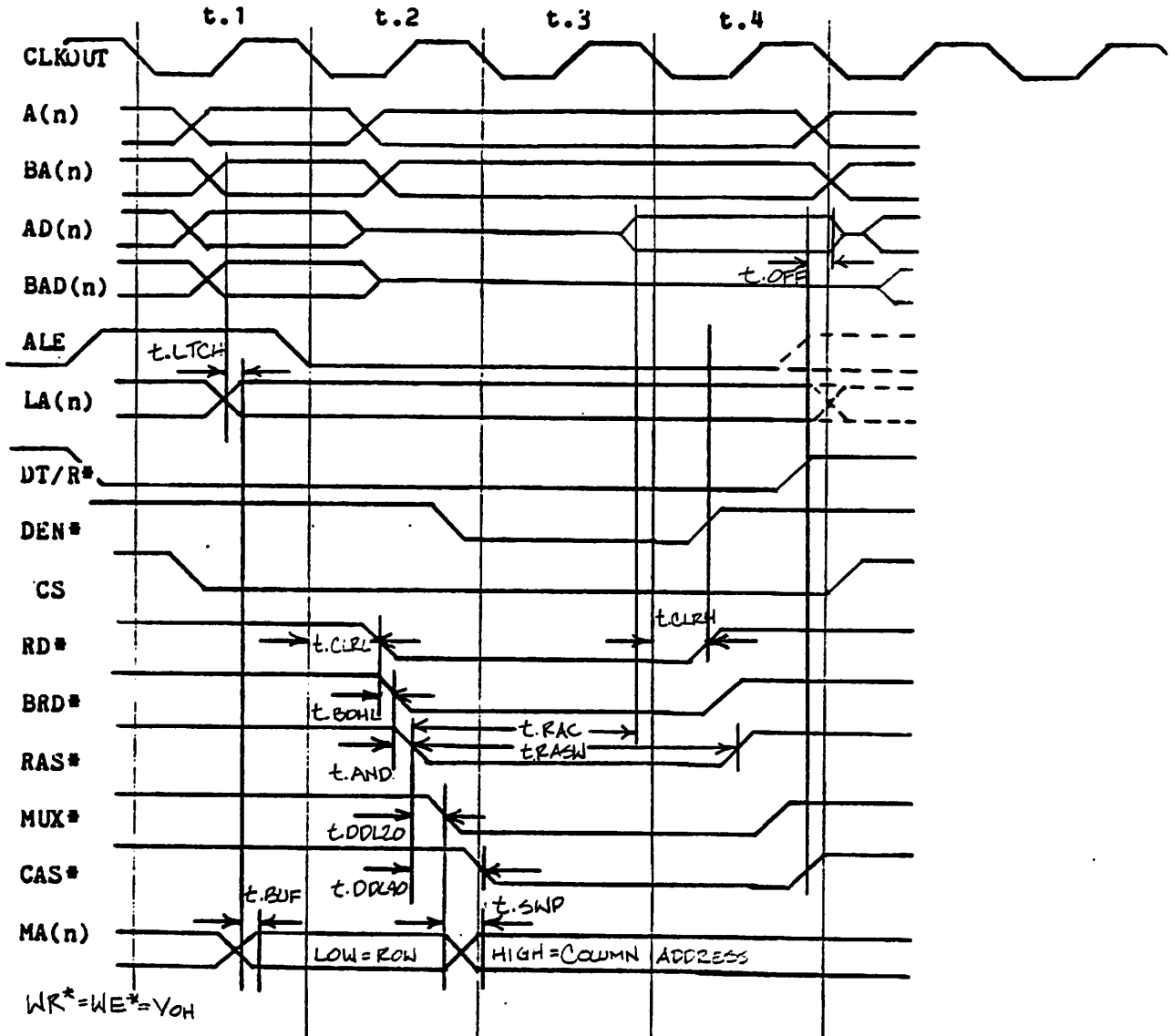
EPROM READ CYCLE



t.CLCSV:	Chip select active delay	66 max.
t.CLRL:	RD* active delay	10-70
t.BDHL:	Tri-state, hi-lo	7-18
t.AND:	AND gate	10-20
t.DDL20:	DDL 20 nS tap	20-24
t.SWP:	10-bit latch tri-state en.	10-17
t.CE	Access time from CE*	350 MAX
t.OE	Access time from OE*	120 MAX
t.ACC.	Access time from address	350 MAX
t.CHCSX	Chip select inactive delay	10-35
t.CLRH	RD* inactive delay	10-55
t.BDLH	Tri-state, lo-hi	5-15
t.OF	Data off delay	0-100

Meets '186 data set up and hold times.

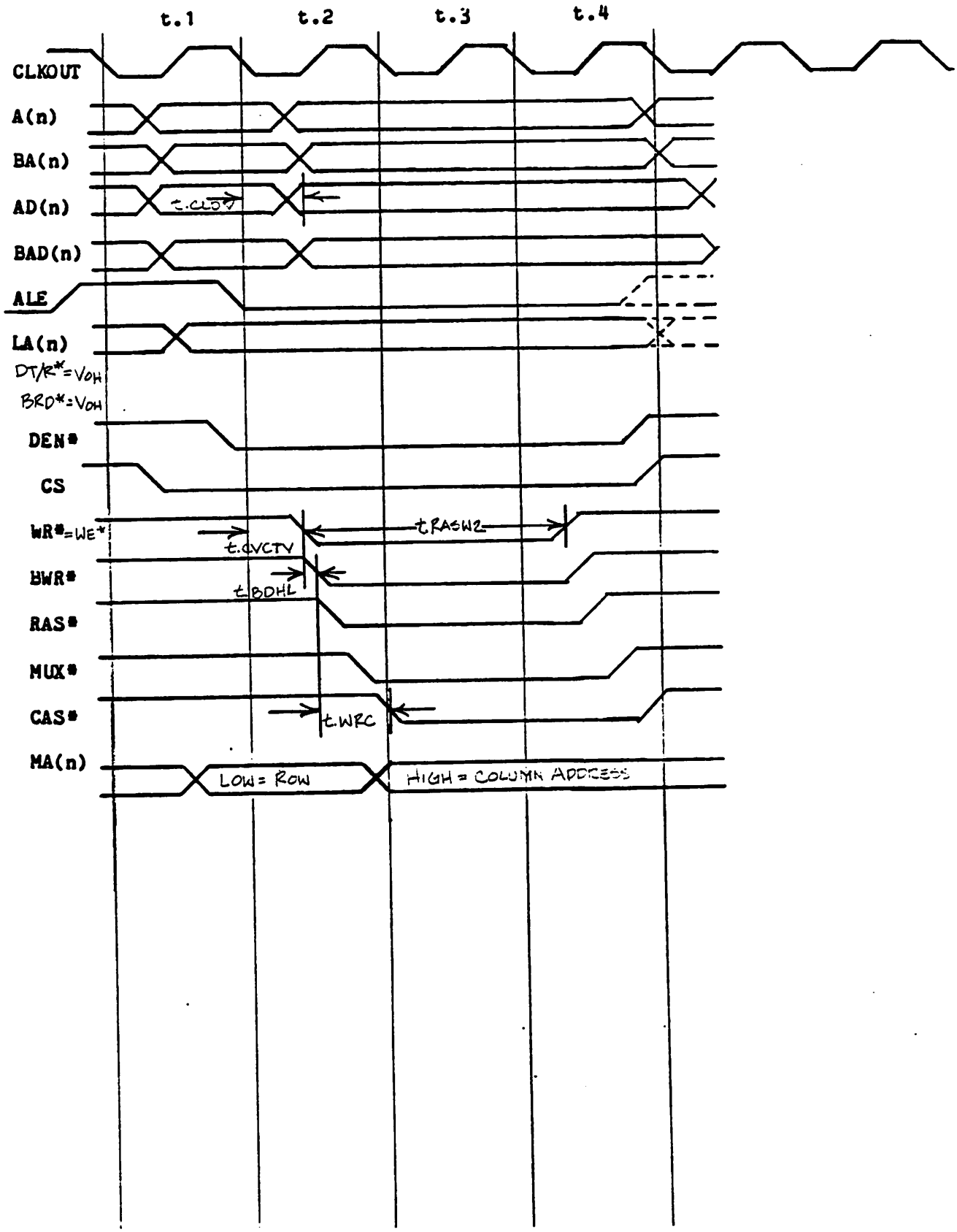
DRAM READ CYCLE



t.LTCH:	Data out delay, LE=hi	6-13
t.BUF:	Data out delay, OE*=lo	5-11
t.CLRL:	RD* active delay	10-70
t.BDHL:	Tri-state, hi-lo	7-18
t.AND:	AND gate	10-20
t.DDL20:	DDL 20 nS tap	20-24
t.DDL40:	DDL 40 nS tap	40-44
t.SWP:	10-bit latch tri-state en.	
t.RASW:	RAS* pulse width =	
	CAS* pulse width = t.RLRH =	150 min.
t.RAC:	Access time from RAS*	150 max.
t.CLRH:	RD* inactive delay	10-55
t.OFF:	Output buffer turn off	0-40

Meets all DRAM specs.

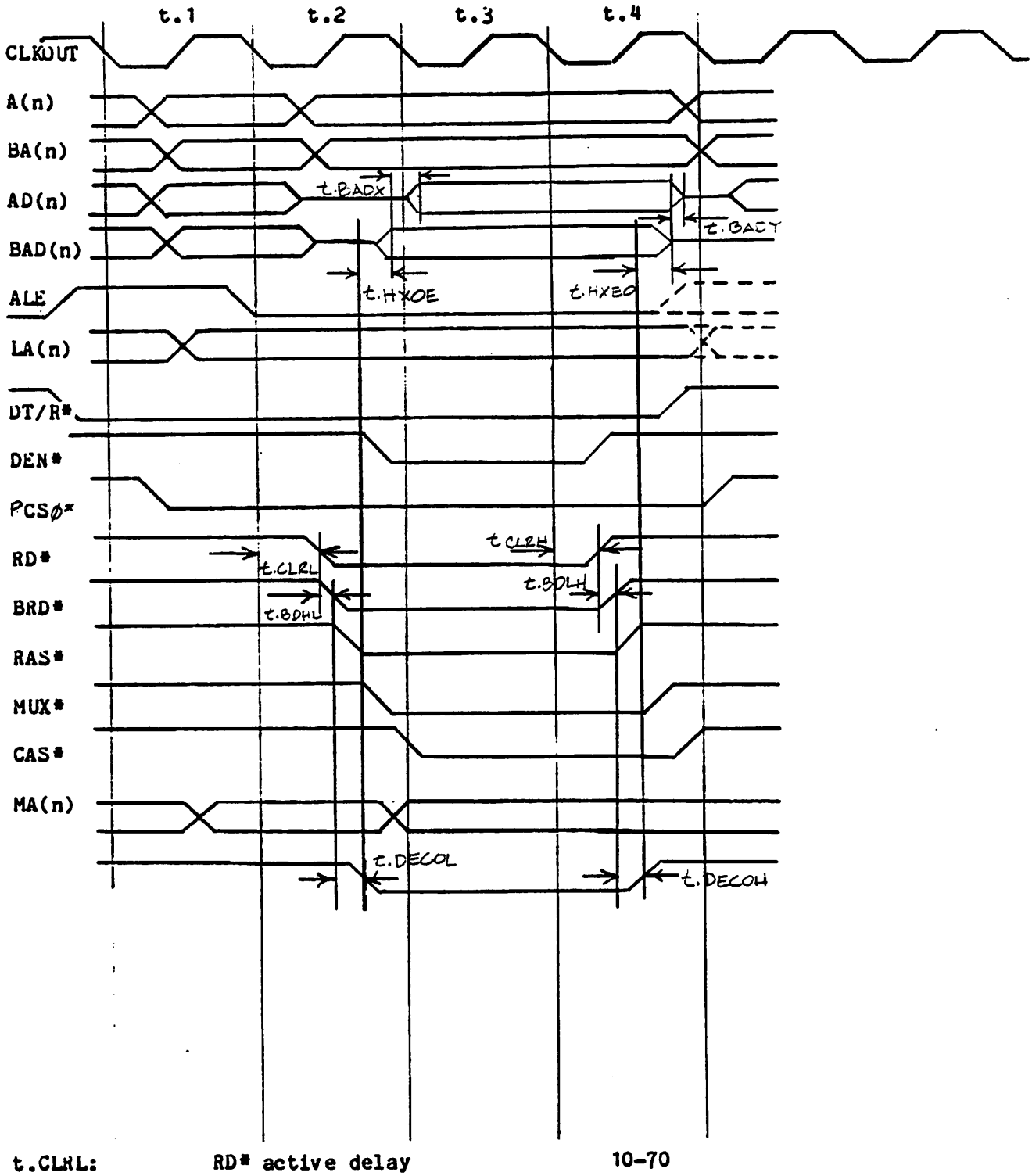
DRAM WRITE CYCLE. See DRAM READ CYCLE TIMING also.



t.RASW2:	RAS* width = CAS* width = WE* width = t.WLWH =	160 min.
t.CVCTV:	Control active delay 1	10-70
t.BDHL:	Tri-state, hi-lo	7-18
t.WRC:	t.AND + t.DDL40	50-64
t.CLDV	Data valid delay	10-44

Meets DRAM data set up time. Meets write command set up time.

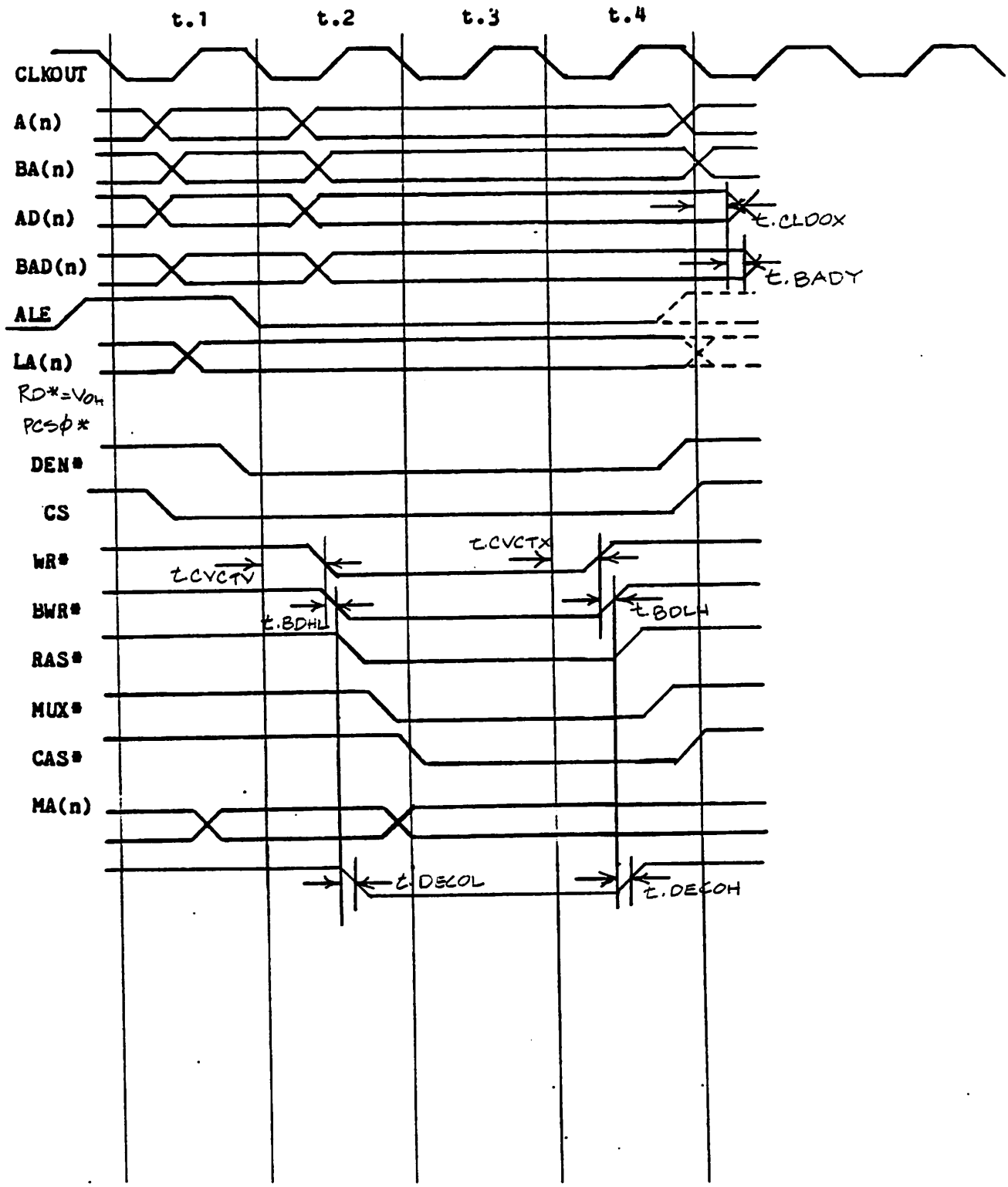
MULTIBUS DATA PORT READ CYCLE.



t.CLRL:	RD* active delay	10-70
t.BDHL:	Tri-state, hi-lo	7-18
t.DECOL:	3-8 decoder, hi-lo	21-32
t.HXOE:	OE* to data, AMD2953	27 max
t.BADX:	BAD bus xcvr delay	12-18
t.CLRH:	RD* inactive delay	10-55
t.BDLH:	Tri-state, lo-hi	9-15
t.DECOH:	3-8 decoder, lo-hi	12-18
t.HXEO:	Data float, AMD2953	22 min
t.BADY:	BAD bus xcvr float	10-25

Meets '186 data set up and hold times.
Meets '186 data float before next address.

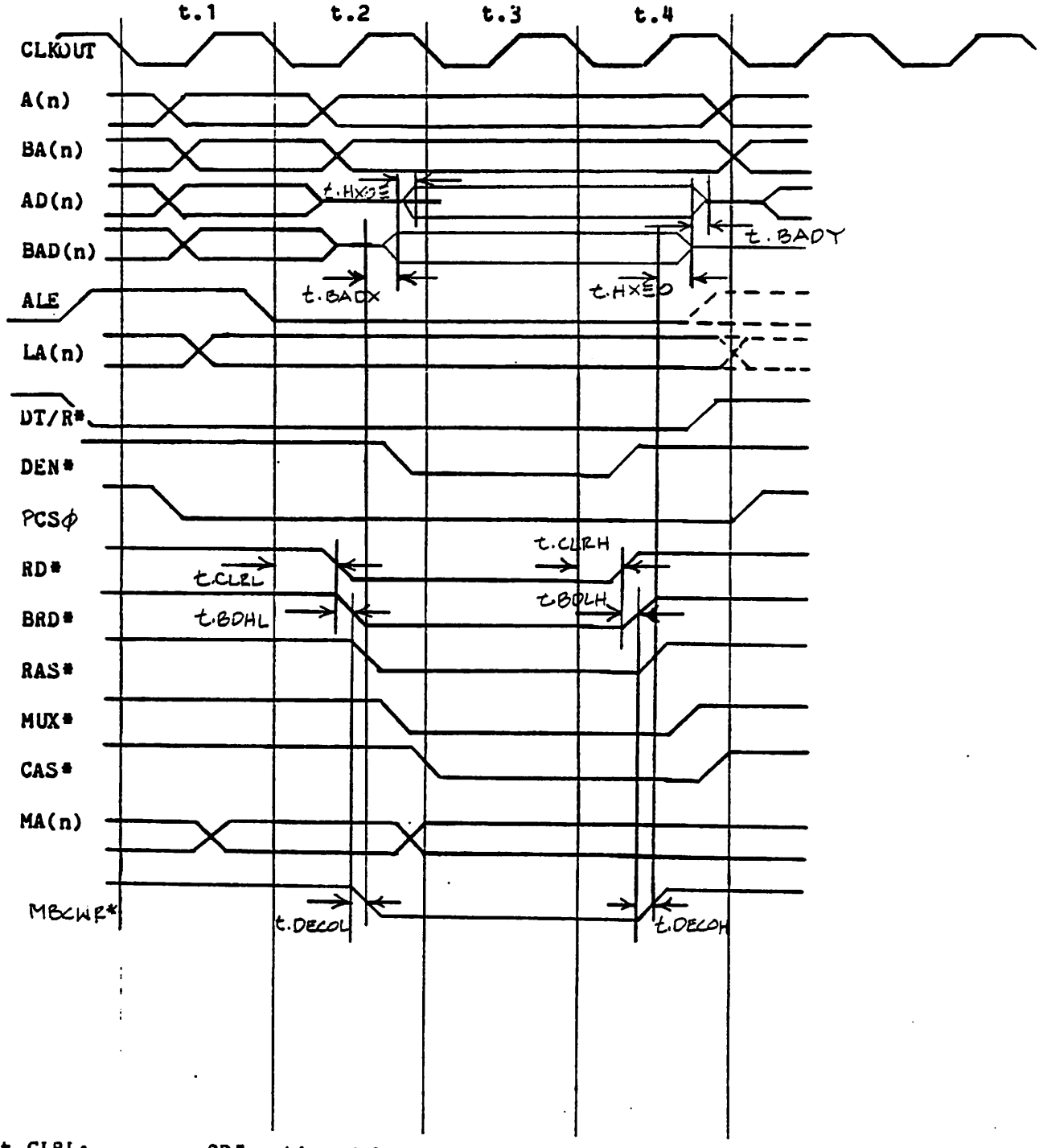
MULTIBUS DATA PORT WRITE CYCLE.



t.CVCTV:	Control active delay 1	10-70
t.BDHL:	Tri-state, hi-lo	7-18
t.DECOL:	3-8 decoder, hi-lo	21-32
t.CVCTX:	Control inactive delay	10-55
t.BDLH:	Tri-state, lo-hi	9-15
t.DECOH:	3-8 decoder, lo-hi	12-18
t.CLDOX:	Data hold time	10 min.
t.BADY:	BAD bus buffer float	10-25

Meets data set up and hold times for AMD 2953.

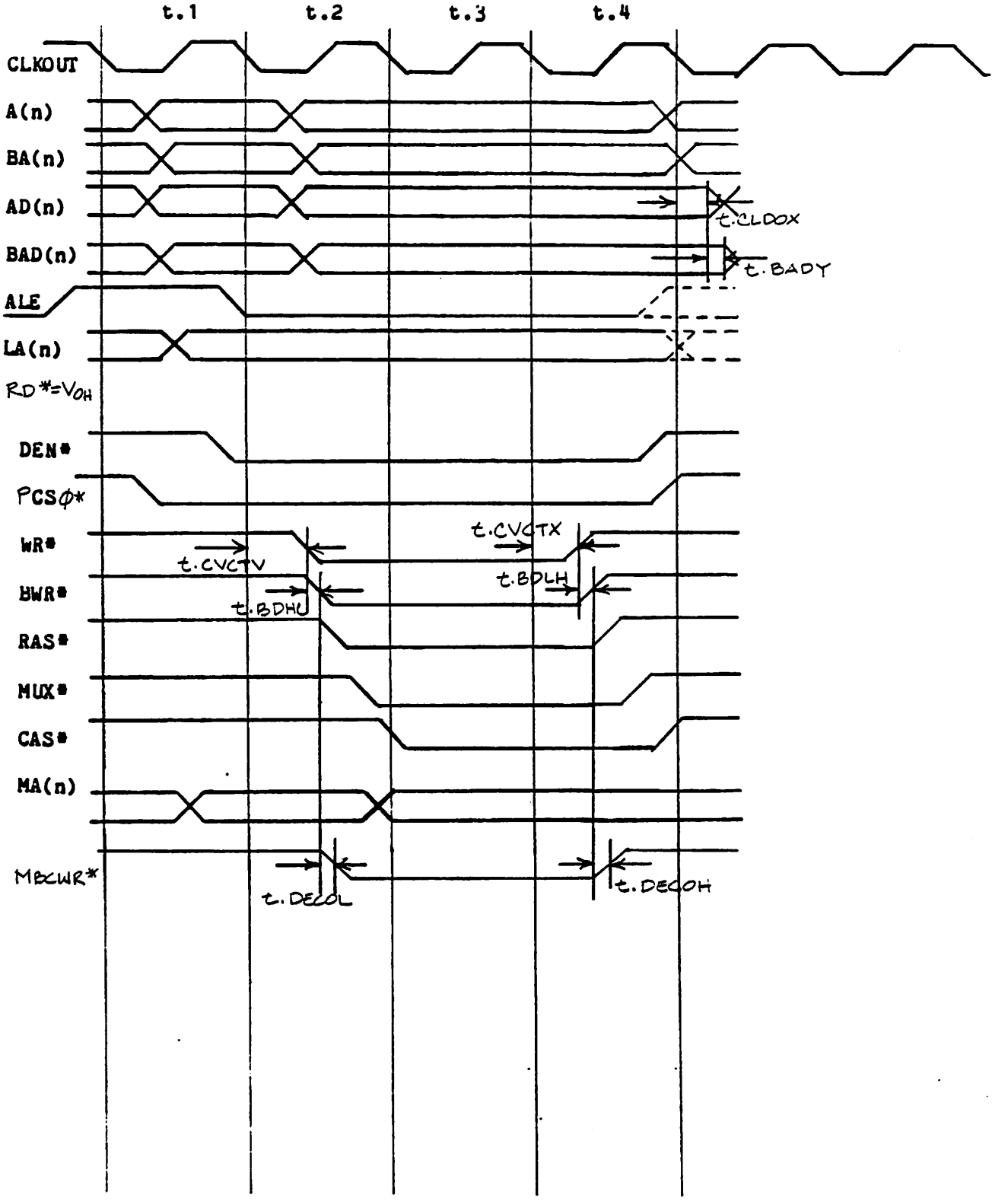
MULTIBUS CONTROL / STATUS REGISTER READ CYCLE.



t.CLRL:	RD* active delay	10-70
t.BDHL:	Tri-state, hi-lo	7-18
t.DECOL:	3-8 decoder, hi-lo	21-32
t.HXOE:	OE* to data, LS367	35-40
t.BADX:	BAD bus xcvr delay	12-18
t.CLRH:	RD* inactive delay	10-55
t.BDLH:	Tri-state, lo-hi	9-15
t.DECOH:	3-8 decoder, lo-hi	12-18
t.HXEO:	Data float, LS367	30-35
t.BADY:	BAD bus xcvr float	10-25

Meets '186 data set up and hold times.
Meets '186 data float before next address.

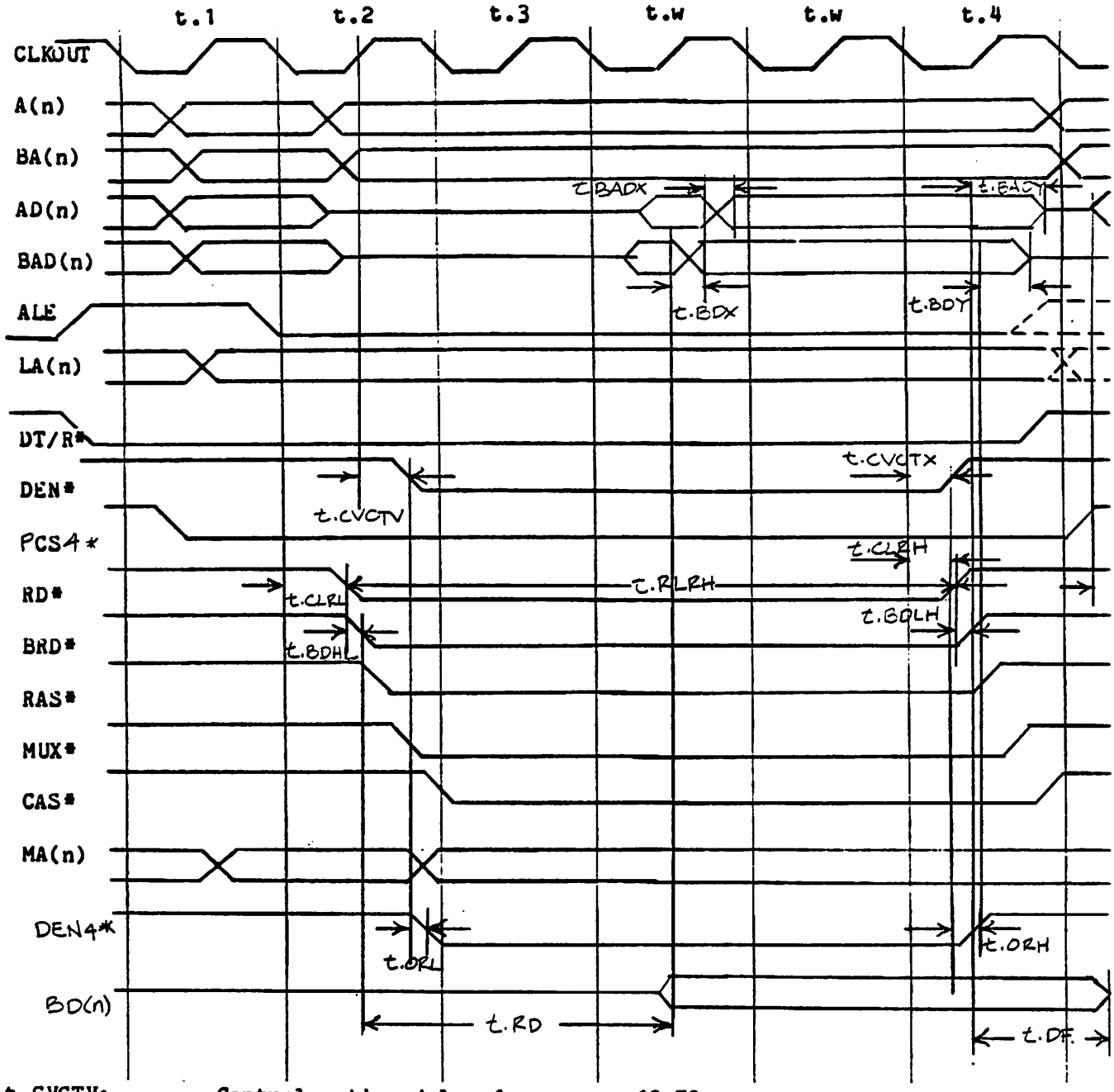
MULTIBUS CONTROL / STATUS REGISTER WRITE CYCLE.



t.CVCTV:	Control active delay 1	10-70
t.BDHL:	Tri-state, hi-lo	7-18
t.DECOL:	3-8 decoder, hi-lo	21-32
t.CVCTX:	Control inactive delay	10-55
t.BDLH:	Tri-state, lo-hi	9-15
t.DECOH:	3-8 decoder, lo-hi	12-18
t.CLDOX:	Data hold time	10 min.
t.BADY:	BAD bus buffer float	10-25

Meets data set-up and hold times for LS174.

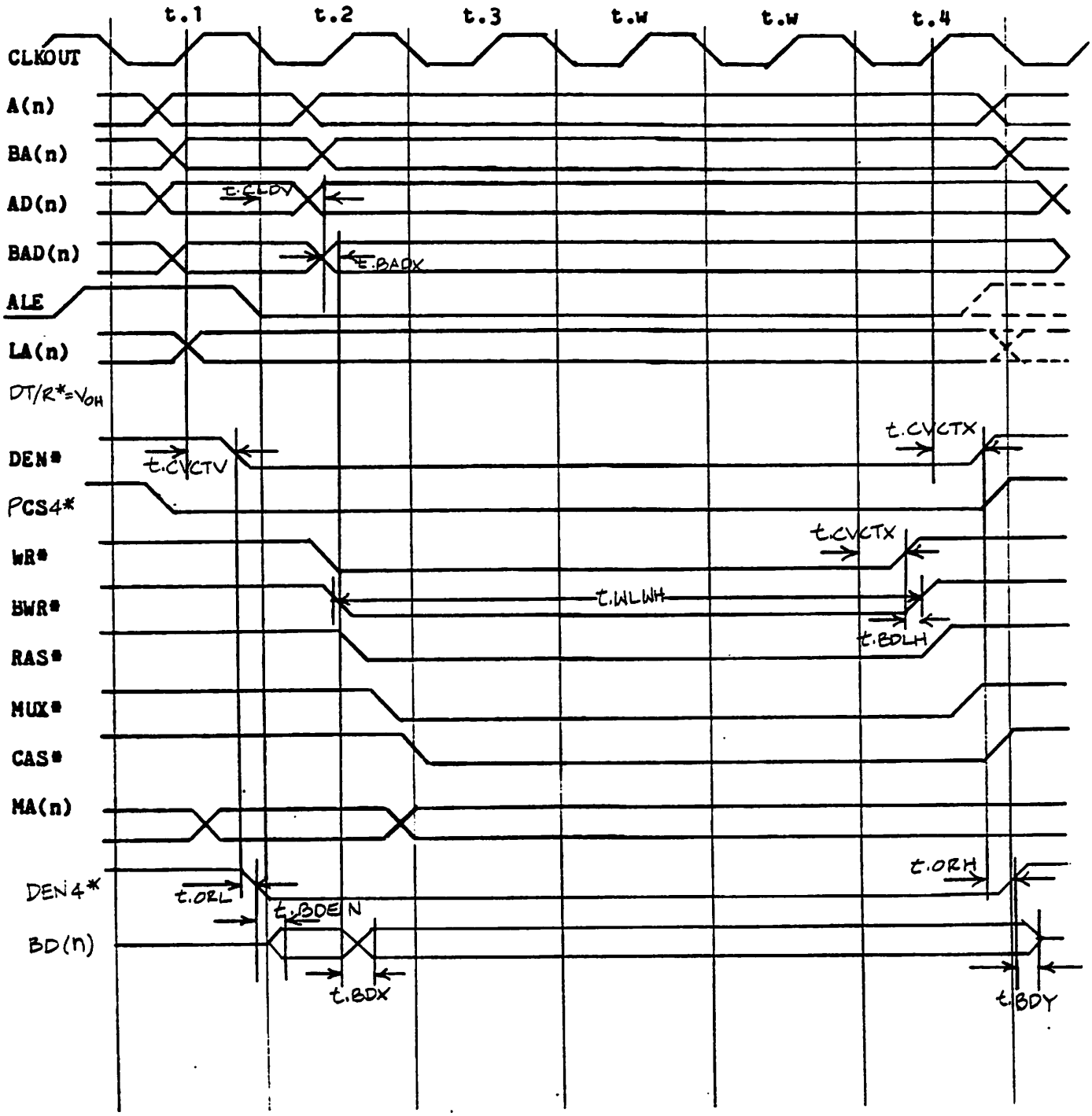
DART READ CYCLE.



t.CVCTV:	Control active delay 1	10-70
t.ORL:	OR gate, hi-lo	14-22
t.RLRH:	BRD* width	250 min.
t.CLRL:	RD* active delay	10-70
t.BDHL:	Tri-state, hi-lo	7-18
t.RD:	RD* lo to data out delay	200 max.
t.BDX:	BD bus xcvr data delay	12-18
t.BADX:	BAD bus xcvr data delay	12-18
t.CLRH:	RD* inactive delay	10-55
t.BDLH:	Tri-state, lo-hi	9-15
t.DF:	Output float delay	120 max.
t.CVCTX:	Control inactive delay	10-55
t.ORH:	OR gate, lo-hi	14-22
t.BDY:	BD bus xcvr data float	10-25
t.BADY:	BAD bus xcvr float	10-25

Meets data set up and hold times for '186.
Meets '186 data float before next address.

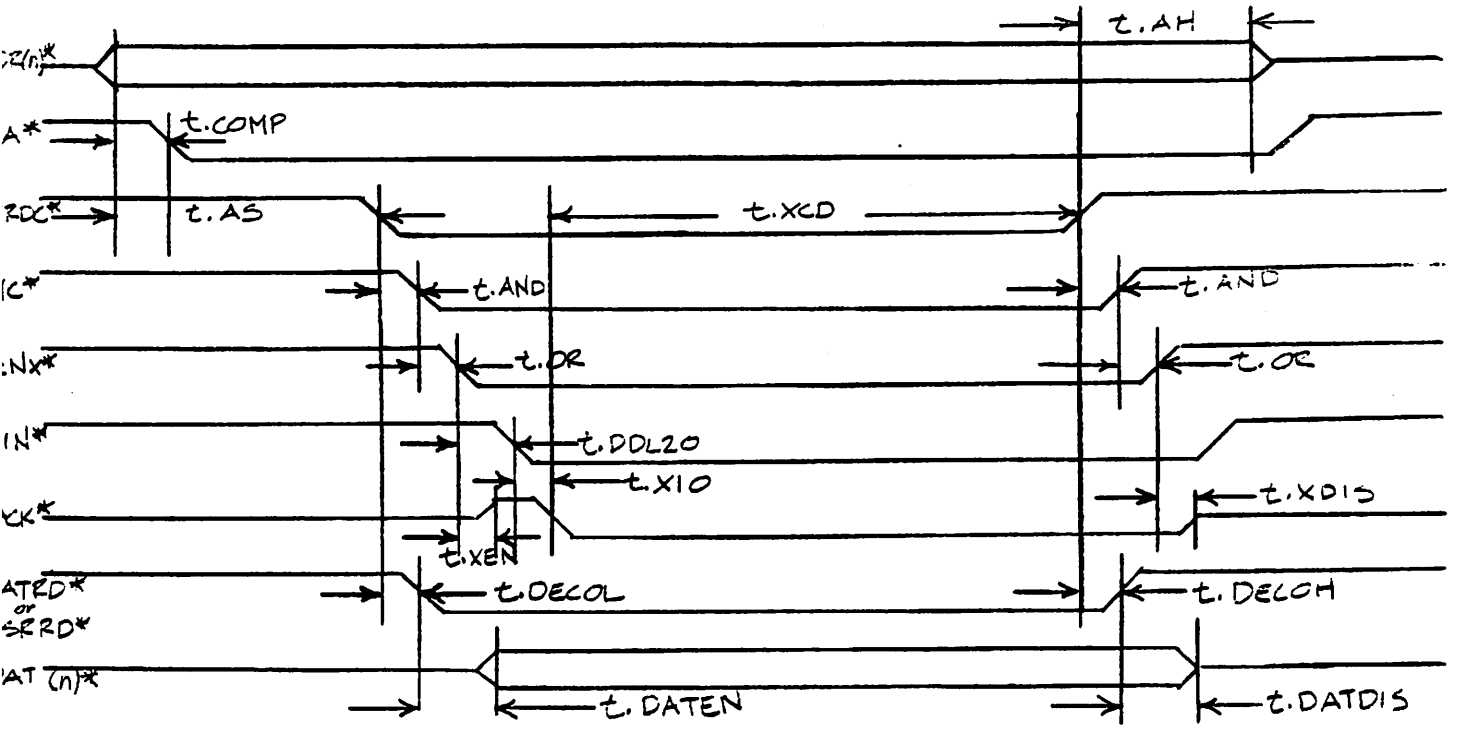
DART WRITE CYCLE.



t.CVCTV:	Control active delay	10-70
t.ORL:	OR gate, hi-lo	14-22
t.BDEN:	BD bus xcvr enable	15-30
t.CLDV:	Data valid delay	10-44
t.BADX:	BAD bus xcvr data delay	12-18
t.BDX:	BD bus xcvr data delay	12-18
t.WLWH:	WR* pulse width	260 min.
t.CVCTX:	Control inactive	10-55
t.BDLH:	Tri-state, lo-hi	9-15
t.ORH:	OR gate, lo-hi	14-22
t.BDY:	BD bus xcvr data float	10-25

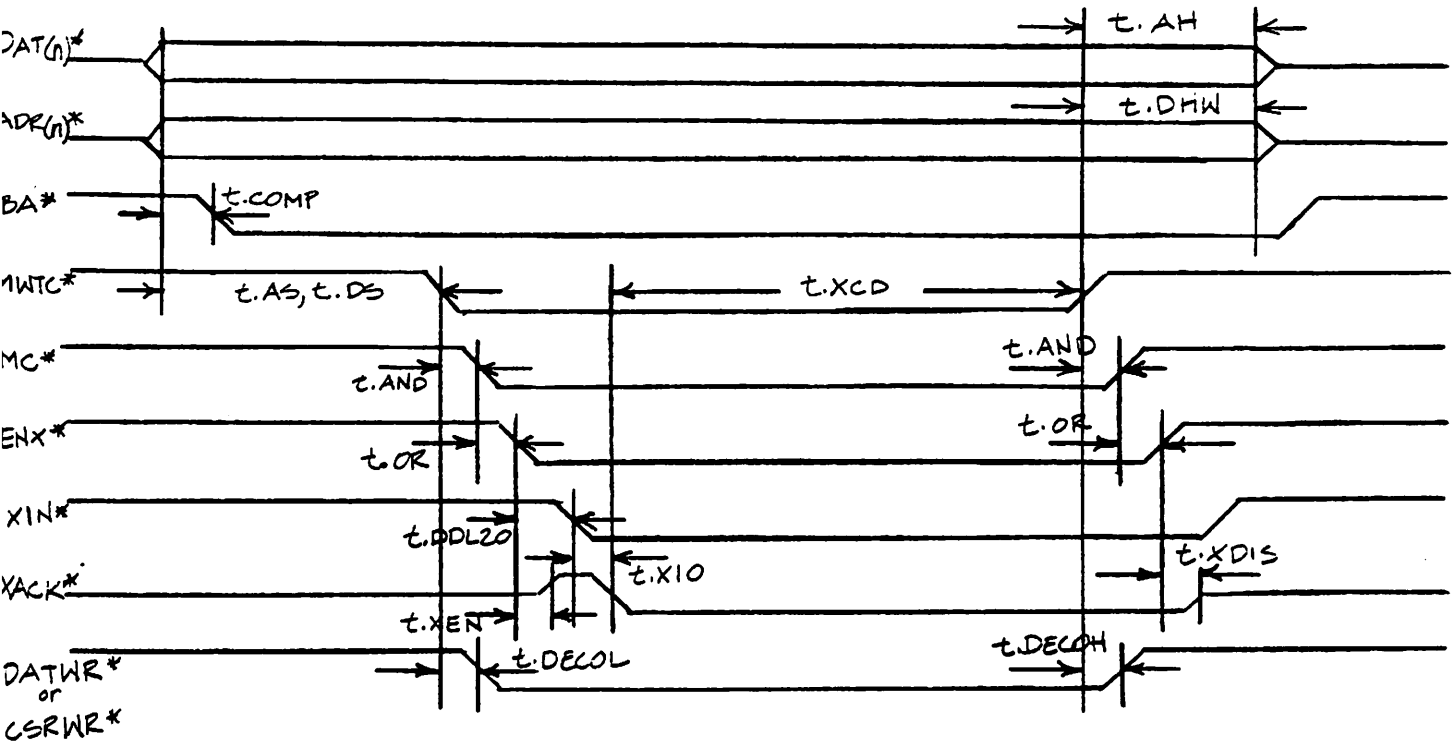
Meets data set up and hold times for 8274 DART chip.

READ OF DATA OR CSR FROM MULTIBUS



t.COMP:	8-bit address comp.	9-15
t.AS:	Address set up time	50 min.
t.AND:	AND gate	10-20
t.OR:	NOR - INV gates	20-30
t.DDL20:	DDL, 20nS tap	20-24
t.XEN:	Enable XACK tri-state	15-25
t.XIO:	XACK tri-state hi-lo	12-18
t.DECOL:	3-to-8 decoder, hi-lo	27-39
t.DATEN:	Data tri-state enable	27 MAX
t.XCD:	XACK to command delay	20 min.
t.AH:	Address hold time	50 min.
t.XDIS:	XACK tri-state disable	20 MAX.
t.DECOH:	3-to-8 decoder, lo-hi	13-27
t.DATDIS:	Data tri-state float	22 MIN.

WRITE OF DATA OR CSR FROM MULTIBUS



t.COMP:	8-bit address comp.	9-15
t.AS:	Address set up time	50 min.
t.DS:	Write data set up time	50 min.
t.AND:	AND gate	10-20
t.OR:	NOR - INV gates	20-30
t.DDL20:	DDL, 20nS tap	20-24
t.XEN:	Enable XACK tri-state	15-25
t.XIO:	XACK tri-state hi-lo	12-18
t.DECOL:	3-to-8 decoder, hi-lo	27-39
t.XCD:	XACK to command delay	20 min.
t.AH:	Address hold time	50 min.
t.DHW:	Write data hold time	50 min.
t.XDIS:	XACK tri-state disable	20 max
t.DECOH:	3-to-8 decoder, lo-hi	13-27

5.7. Parts List

1) Intel 80186

1) Intel 8274 DART

2) generic 2716 - 27128 EPROM, -200 pref.

16) Hitachi HM4864P-2 DRAM

2) AMD 2953(A) inverting bi-port

2) AMD 29841 10-bit latch

1) AMD 29827 10-bit buffer

1) AMD 25LS2521 8-bit comp.

2) Belfuse 0447-0050-02

1) TI 75150 dual line driver

1) 20 MHz crystal, HC-18U.

1) MC1489A, quad RS-232 rcvr.

3) 'LS245, 8-bit xcvr

1) 'LS125, quad tri-st.

1) 'LS02, quad nor

1) 'LS04, hex inv

2) 'LS08, quad and

1) 'LS32, quad or

1) 'LS51, and-nor

3) 'LS138, 3-8 decoder

1) 'LS109, dual J/K*

1) 'LS163, 4-bit counter

2) 'LS174, 6-bit latch

3) 'LS367, 6-bit buffer