'977

AN-777
APPLICATION NOTE

A DUAL PROCESSOR SYSTEM FOR USE IN THE EXORciser

Prepared by:
Raymond H. Naugle
LSI Systems Applications

This application note describes the design of a dual processor system for use in the EXORciser. In this system, the processors have control of a common bus on opposite phases of the clock.

MOTOROLA *Semiconductor Products Inc.*

A DUAL PROCESSOR SYSTEM FOR USE IN THE EXORciser

INTRODUCTION

This paper describes the design of a dual processor system with a minimum of additional circuitry over a single processor system. The system is EXORciser compatible, operating under a common program, EXbug.

DUAL PROCESSING

You can increase your system's throughput by using two processors instead of one. The added speed is accomplished by doubling the available processing power which includes increased interrupt handling capability for the same system. The system may also be designed so the cost will be only a little more than a single processor system.

Many configurations are possible when connecting two processors together. Figures 1 and 2 show two implementations each using two processor subsystems connected by some logic interface. Although these figures show two completely separate systems, they may share some common components such as power supplies, clock circuitry, etc. Also, these two implementations may be extended for use in a system with more than two processors.

In Figure 1, PIAs are used for communication between the two systems. The PIA contains two 8-bit data ports with additional internal logic to take care of the "handshaking" between the two processors when the data is transferred. The PIA can be programmed so that one is an input port and the other is an output port. In this scheme, one processor could handle the peripheral input and some data formatting, then send the data to the second processor for the data maipulation. After the task is complete, the second processor sends formatted data back to the first processor for unformatting and output to a peripheral. Thus, while the first processor is receiving/transmitting data, the second processor could be doing the "number crunching".

Figure 2 shows a FIFO (First-In First-Out Register) being used for transferring data between the processors. (Shown in the diagram is data flow in one direction, but the same logic could be repeated for data transfer in the other direction.) In this configuration, the first processor may stack data for use by the second processor and the second processor may read data as needed, thus saving processing time from answering interrupts for data being transferred in. Here, the processors are allowed to operate more efficiently.

Another approach to the dual-processor implementation is shown in Figure 3. Here both processors utilize the same bus, operating on opposite phases of the clock. This system eliminates the need for a second bus structure and most of the hardware or software for communication between processors. Since the processor transfers data only during the $\phi 2$ portion of its clock cycle, it needs to have access to the bus only during this time and not during $\phi 1$. Therefore, when one processor is in the $\phi 1$ portion of its clock cycle, the other processor is in its $\phi 2$ portion and has control of the bus. This doubles the effective speed of the data transfer rate. (Since the bus cycle time has been cut in half, the response time of the bus parts should be checked to insure proper timing.)

Communication between the two processors may now be done in common RAM. This feature saves adding a couple of PIAs or FIFOs plus extra interface logic.

Some other features of this system include eliminating some ROM where common routines may be utilized. If both processors execute the same program, the cost of the total system's ROM will be cut in half.

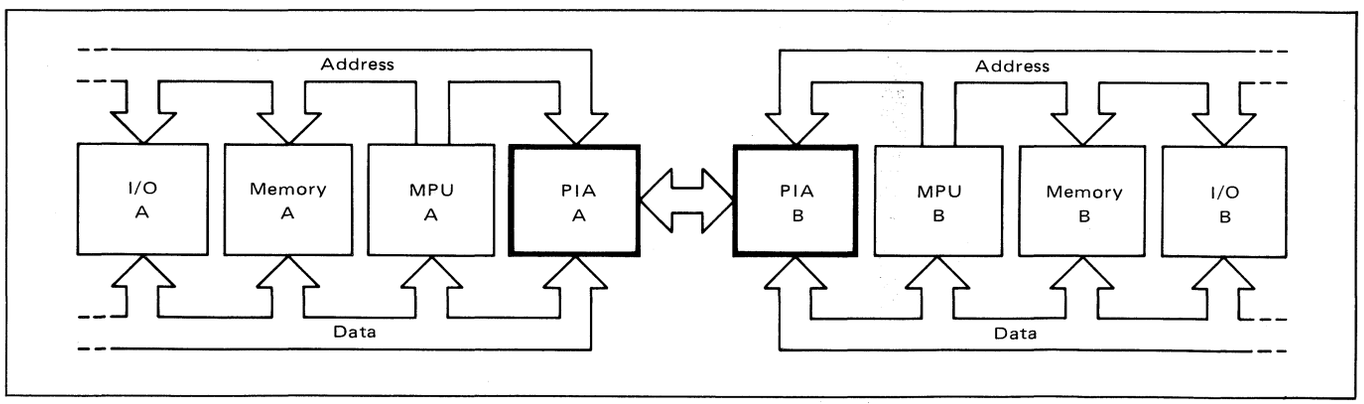

FIGURE 1 – Dual Processor System Using PIAs

EXORciser and EXbug are trademarks of Motorola Inc.

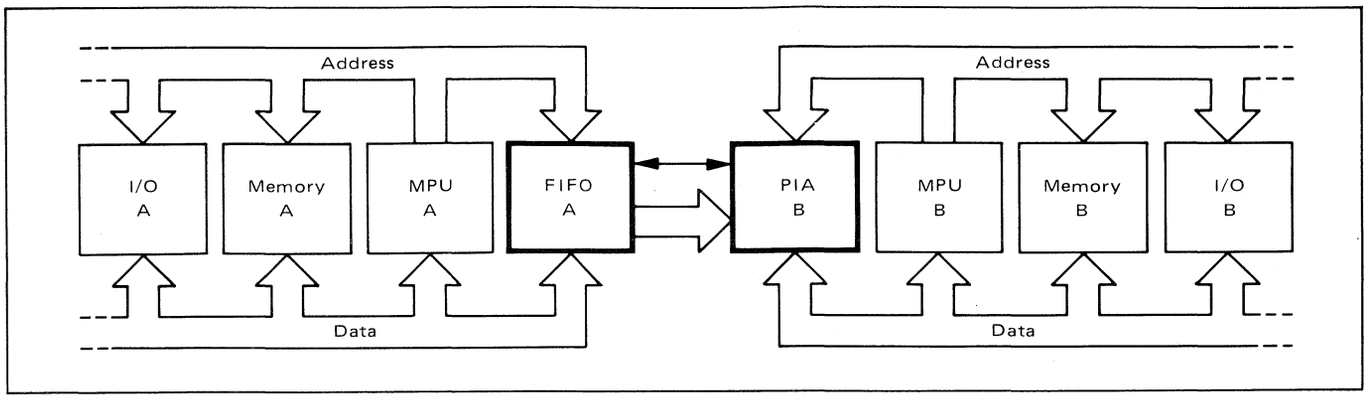

FIGURE 2 – Dual Processor System Using a FIFO

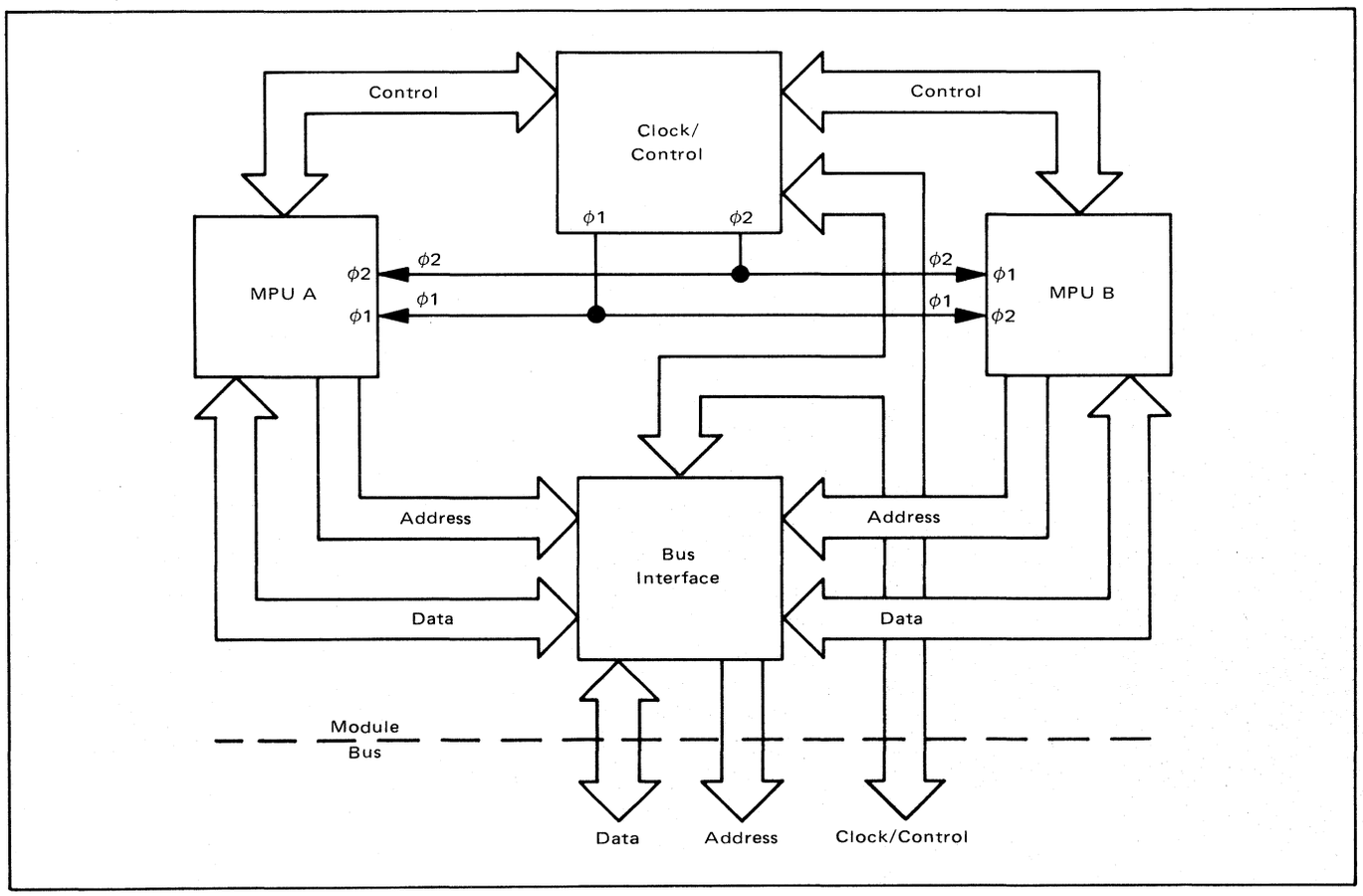

FIGURE 3 - Dual Processor System Operating on Opposite Phases

If the same program is used for both processors, each processor will probably need some dedicated "scratchpad" RAM and I/O. Dedicating certain blocks of RAM and I/O to one processor can be done by including a clock signal as part of the address decoding. If processor A needs a certain block of memory dedicated to it, the clock $\phi 2$ signal should be included in the address decoding for that block. Then when processor B tries to access that block during its $\phi 2$ portion of the clock, the clock's $\phi 2$ is low, thus, that memory block isn't fully addressed and won't respond, while the memory enabled with the clock's $\phi 1$ signal will respond only to processor B.

SYSTEM DESIGN

Of the three dual processor systems described, the latter approach was used because of its simplicity, minimum package count, and common utilization of memory and software. The dual processor system was designed in accordance with the following set of requirements.

First the system will use one EXORciser with little modification, since the EXORciser has the needed bus structure and power supplies.

Second, both processors will execute the software resident to the EXORciser system, namely EXbug. Using this program will eliminate extensive software develop-

ment for the system, since EXbug has routines to load and punch program tapes, change memory, and start execution of the programs entered, along with some program debugging capability.

Third, each MPU must have dedicated memory and I/O to accommodate the EXbug program. The I/O (serial communication utilizing an ACIA) must interface to both a TTY 20 mA loop and an RS-232C terminal interface. In addition, each MPU must have a ROM for vectoring to a system reset address.

Fourth, each processor will operate at a 1 MHz clock cycle time. With both processors operating at 1 MHz, the bus will operate at 2 MHz or 500 ns cycle time. To be included in the clock design is the ability to refresh memory on a "cycle-stealing" basis, thus transparent to the MPU. Also to be included should be the ability to slow the clock to allow data transfer for the slower responding bus parts.

Fifth, when one or both of the MPUs are halted (either by a WAI instruction or pulling the $\overline{\text{Halt}}$ line low) the address and data bus should go into the high-impedance state, with the exception of VMA. VMA should go to a "0" level so erroneous reading and writing does not occur. An option should be included so control of the bus may be taken over during the time when an MPU is in a halt condition.

CIRCUIT DESCRIPTION

The EXbug program requires a minimum of an ACIA, RAM, and ROM for its hardware support. These items are needed for program operation and are duplicated for each MPU. The addition of a PIA and its supporting hardware will enable the following EXbug commands: (;P), (n;P), (N), (;N), (n;N), ($T), and ($S). (See the EXORciser User's Guide for command descriptions.) Although not included in this system, the address decoding must allow for the PIA so that no other component on the bus will respond to that address.

Figure 4 shows the memory map for this system. Common user memory is allocated addresses $0000 to $EFFF (Hex). The EXbug program occupies addresses $F000 to $FBFF. The address block $FC00 to $FFFF is a block of memory dedicated to each MPU. In this block is the mutually exclusive hardware support for the EXbug program.

The block diagram for this system implemented as one module for the EXORciser is shown in Figure 5. This system is divided into three blocks plus the MPUs. The three blocks consist of Clock/Control, Bus Interface, and Dedicated Memory and I/O.

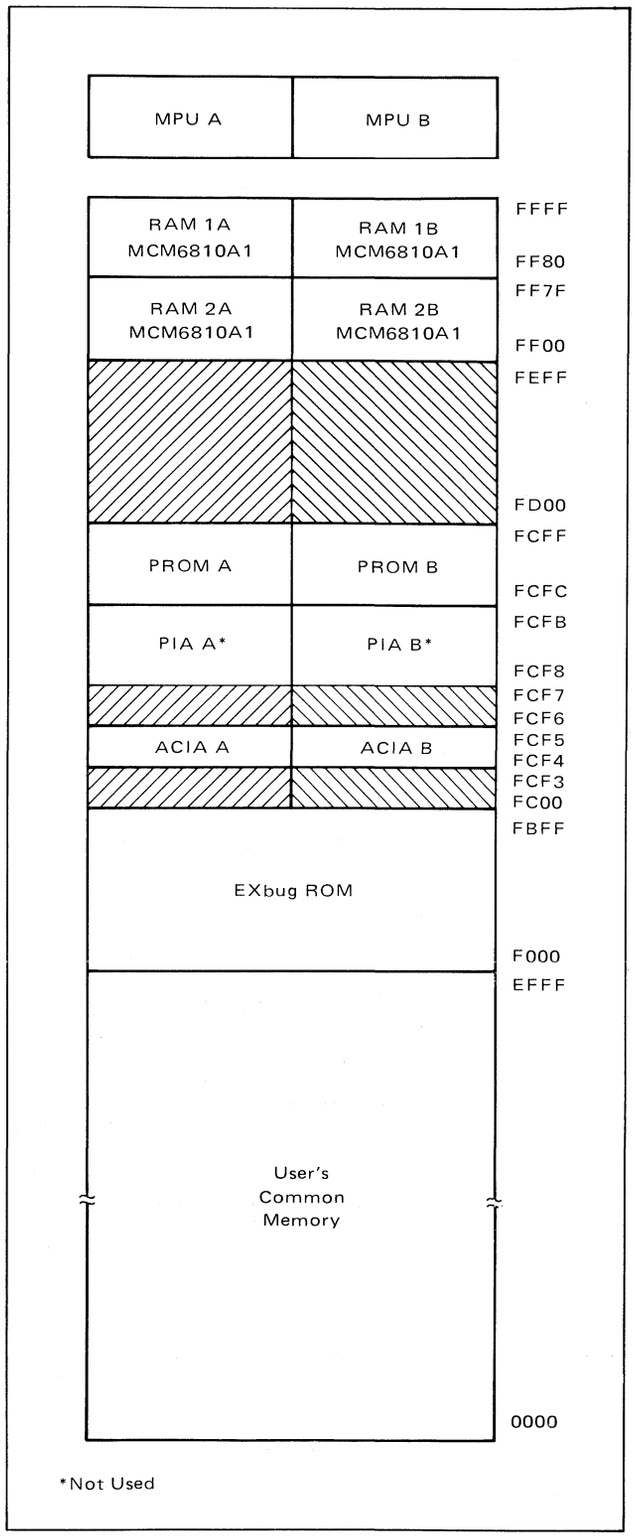

FIGURE 4 – Dual Processor System Memory Map

Clock and Control

The clock and control section is described in two sections: Reset and Clock Design.

Reset. The reset of either the A or the B MPU can be actuated by a number of signals (Figure 6). Both the A and B systems are reset by a power-on reset. When the +5 V power is turned on, the timer (MC1455) is automatically triggered and pulls the $\overline{\text{Master Reset}}$ line low, resetting the whole system. When the timer has timed out (about 400 ms) the reset line is brought high allowing both the A and B MPU to come out of the reset mode and start execution. The $\overline{\text{Master Reset}}$ line may also be pulled low by a signal from the bus. The bus master reset and the power-on reset are wire-ORed so either may cause the whole system to be reset.

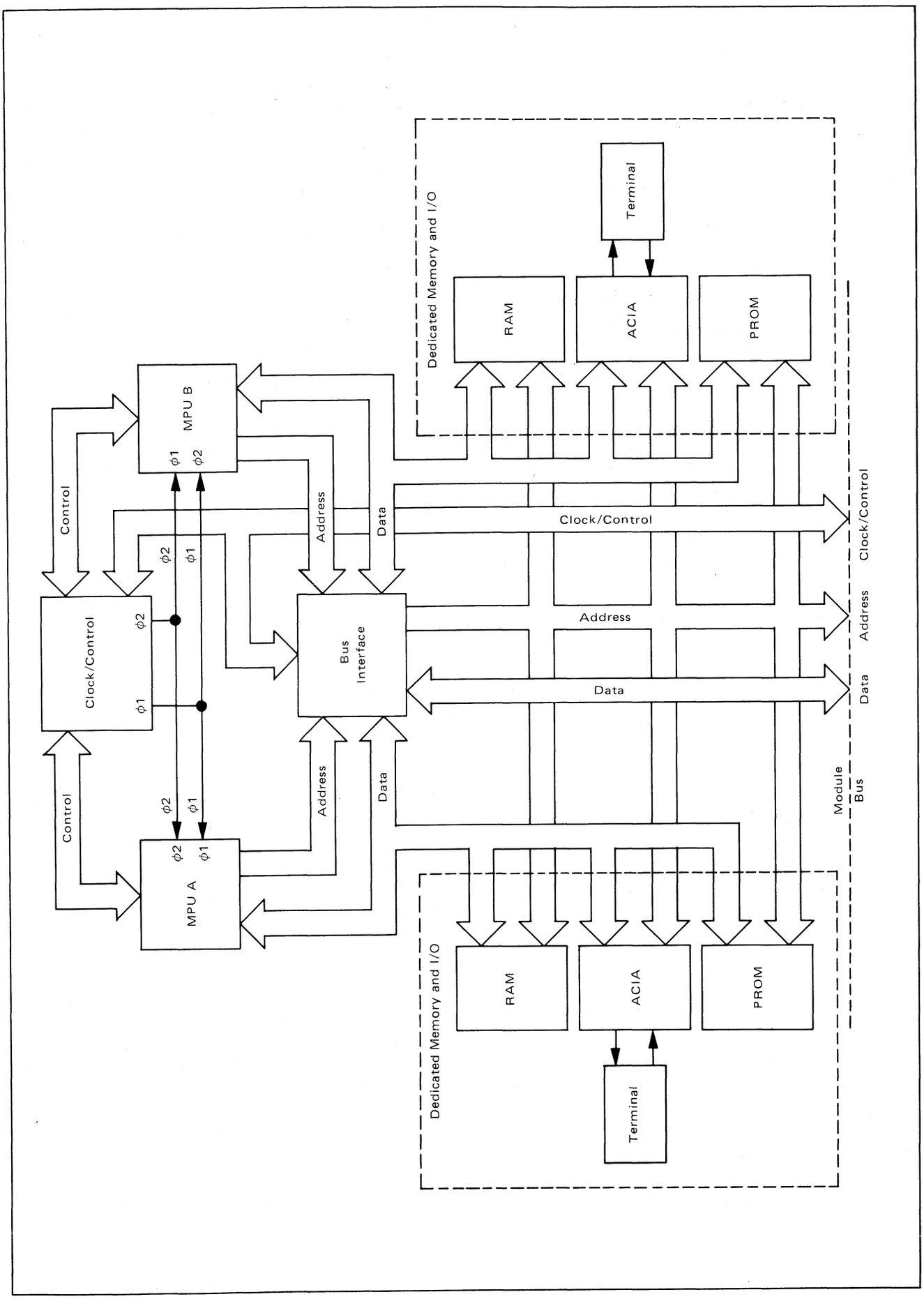

FIGURE 5 – Dual Processor System for Use with EXORciser

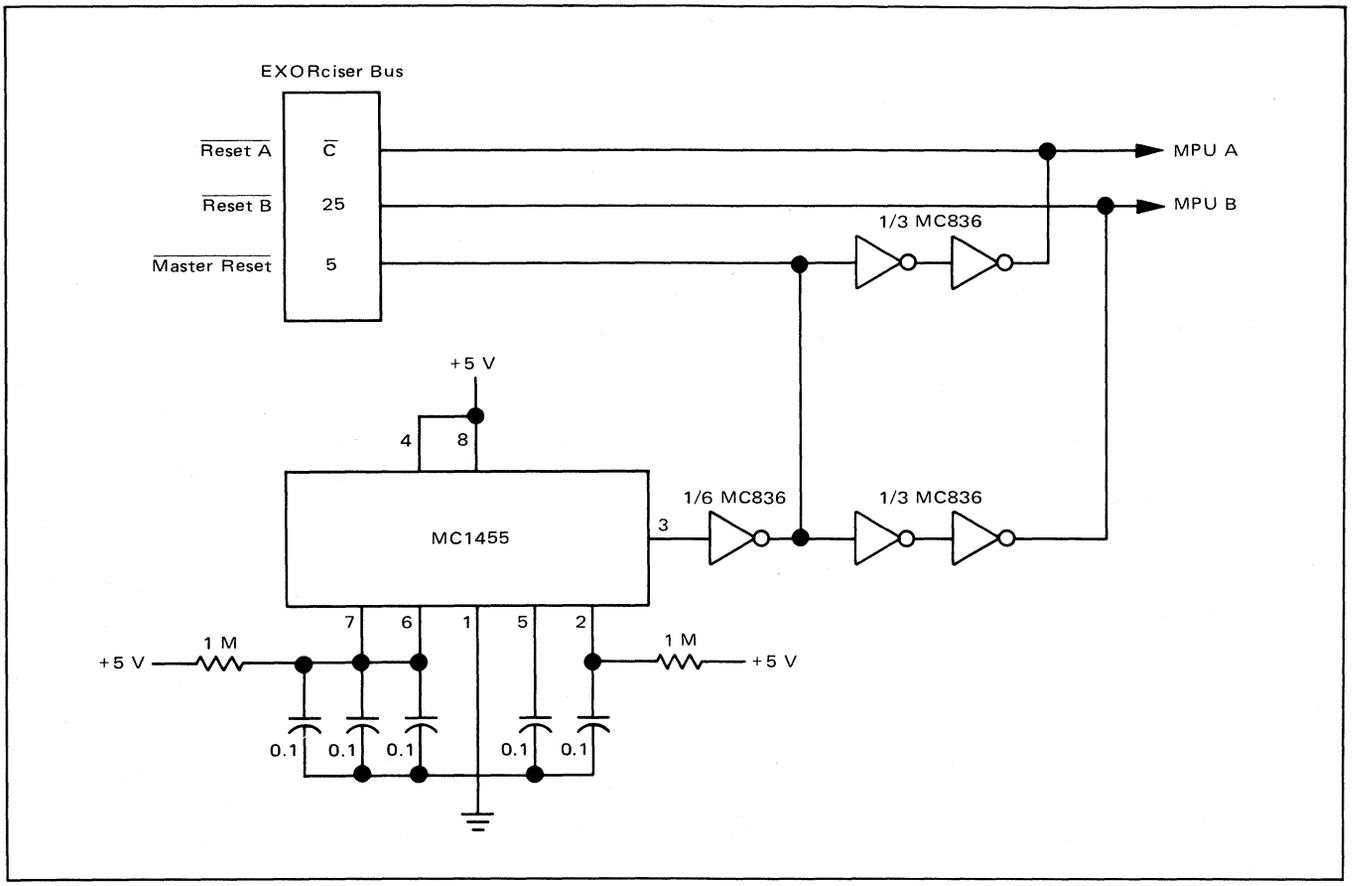

FIGURE 6 – System Reset

Each MPU also has its own reset line ($\overline{\text{Reset A}}$ and $\overline{\text{Reset B}}$) which will reset the respective MPU and the bus parts connected to it. These two reset lines are pulled low each time the entire system is powered up or when the $\overline{\text{Master Reset}}$ line on the bus is pulled low. Also, an individual MPU may be reset by pulling the reset line associated with it ($\overline{\text{Reset A}}$ or $\overline{\text{Reset B}}$) to ground. The reset from the bus is wire-ORed with the reset from power-on/$\overline{\text{Master Reset}}$ so either signal may reset the MPU.

Clock Design. The system clock can be analyzed in two parts: The bus clock and the MPU clock (Figures 7 and 8). The bus clock runs at 2 MHz and provides the 2ϕ2 signal for the bus. Additional circuitry is included in the bus clock to allow for the slow response time of some bus parts and for the refreshing of dynamic memory.

When using dynamic memory, a refresh cycle must be provided to recharge the memory cells. To execute a refresh cycle, the memory module generates a request for refresh ($\overline{\text{REF REQ}}$) signal. This signal asks the clock for a refresh cycle. On the next 2ϕ1 portion of the clock cycle (see Figure 9A), the clock generates a refresh grant (REF GNT) signal telling the bus that a refresh cycle has been granted and it is now taking place. During the refresh cycle, the 2ϕ2 clock remains low for that cycle of the clock. Since some memory modules need a clock for timing during a refresh cycle, another clock signal is also generated called Memory Clock (MEM CLK). This clock signal is the same as 2ϕ2 except during a refresh cycle it continues to cycle while 2ϕ2 remains low for that clock period.

Also included in the bus clock design is the ability to stretch the 2ϕ2 or data transfer portion of the cycle. Since the clock period is 500 ns (2 MHz) some of the logic interfaced to the bus may not have enough time to respond to the fast rate and requires more time for the data transfer. When an MPU addresses such logic, the logic should respond by pulling the Memory Ready

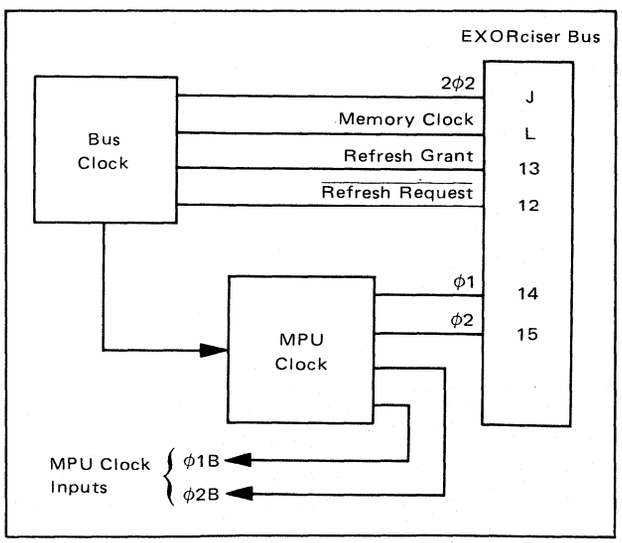

FIGURE 7 – System Clock Block Diagram

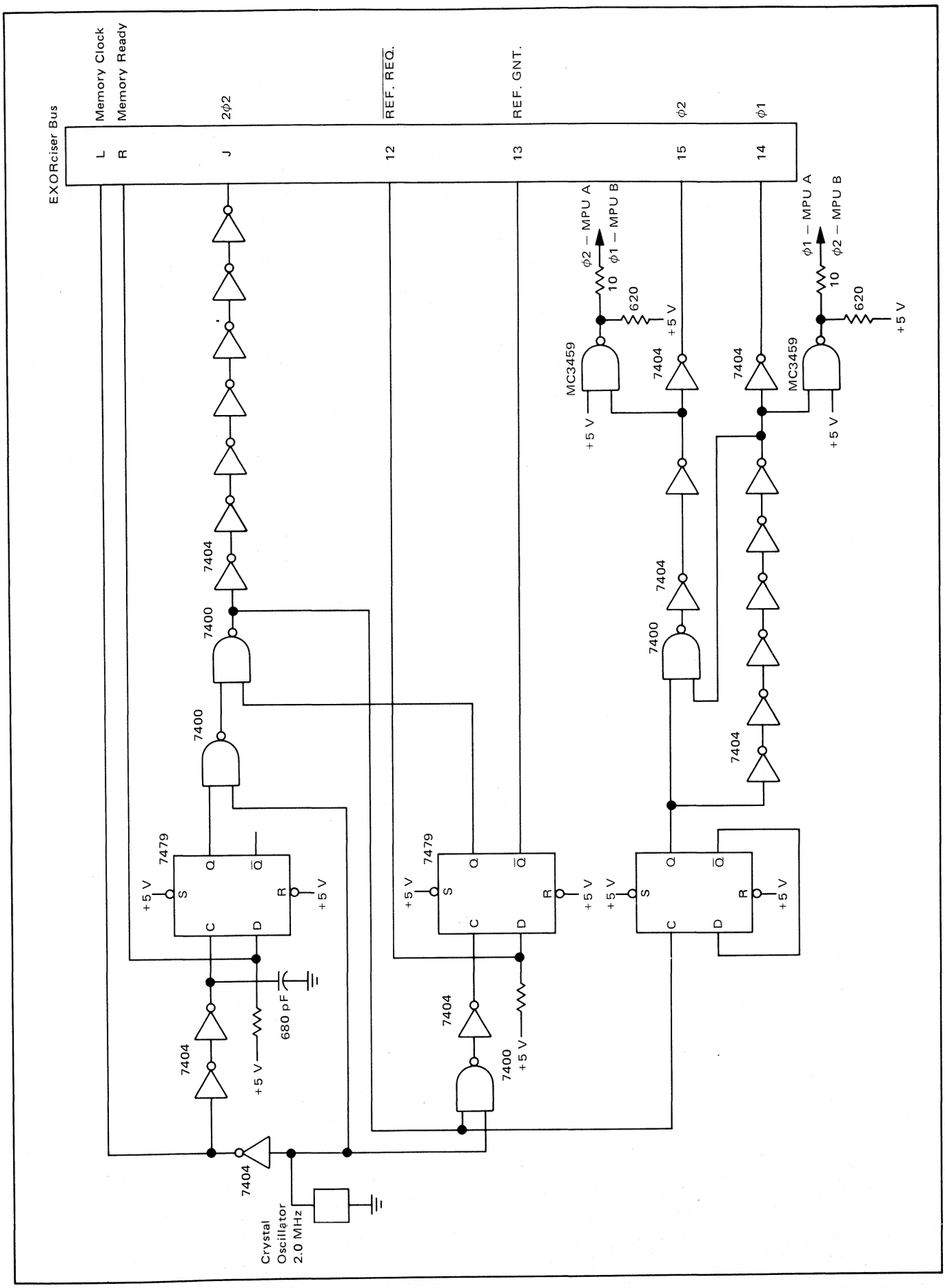

FIGURE 8 – System Clock Schematic

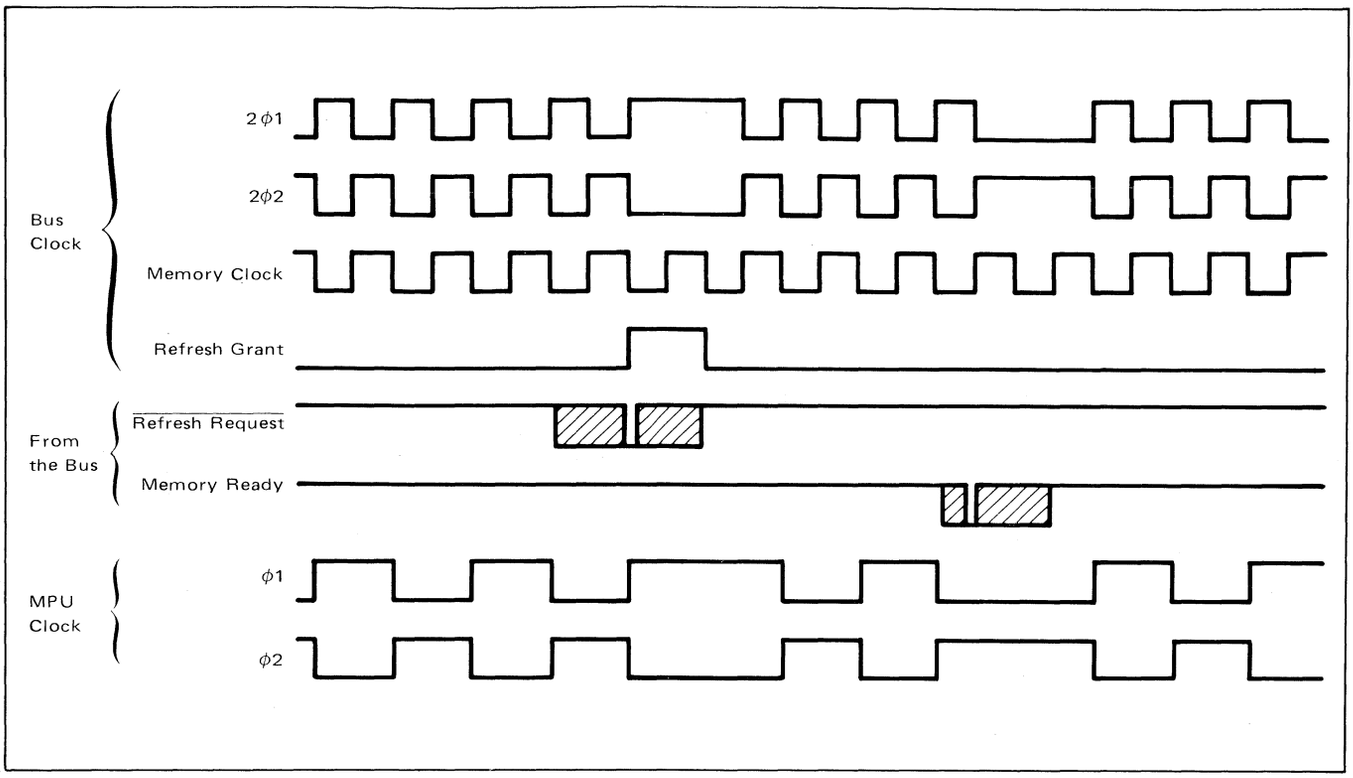

FIGURE 9A – System Clock Timing Diagram – Refresh Cycle, Slow Memory Cycle

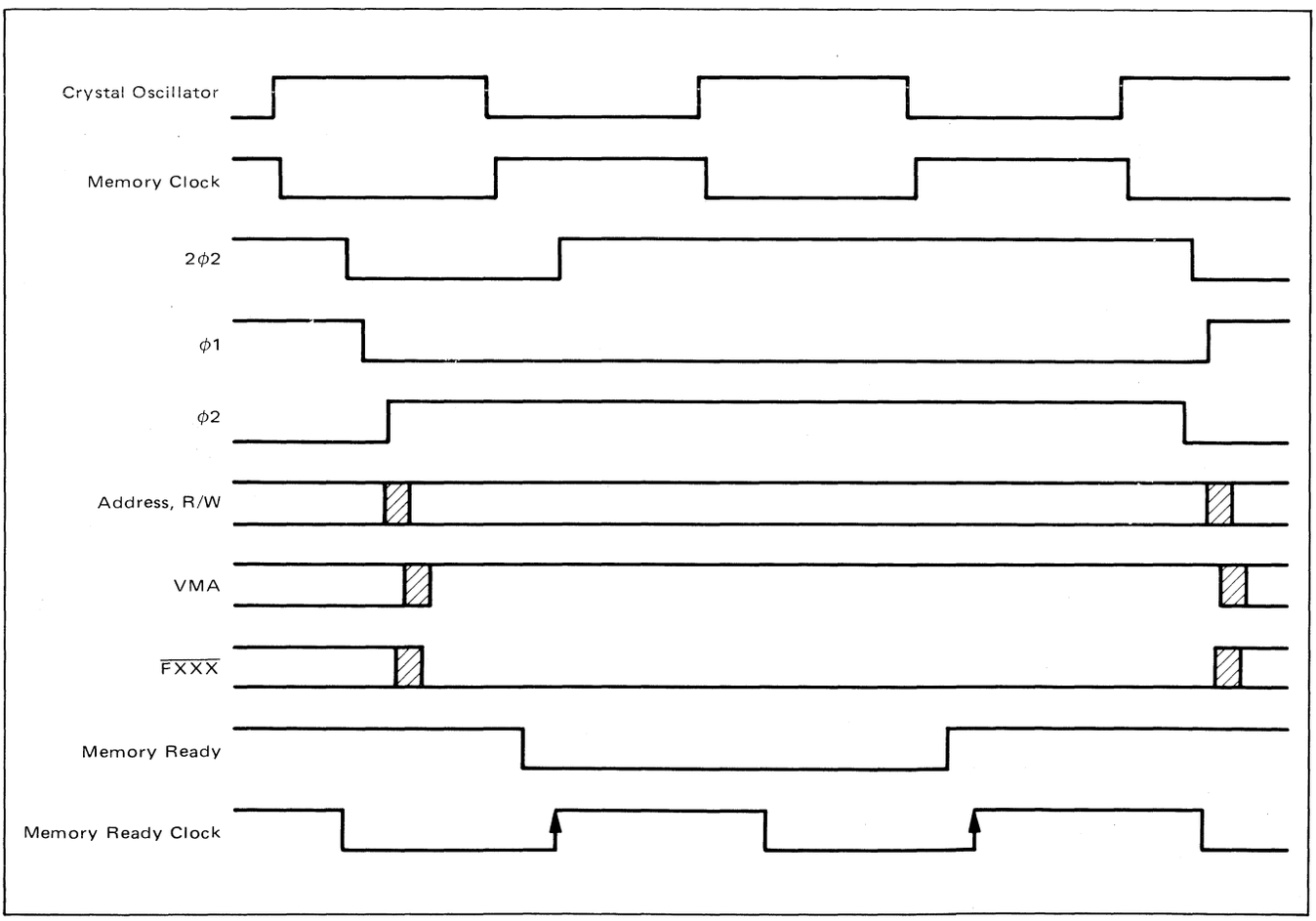

FIGURE 9B – System Clock Timing Diagram – Addresses $F000 to $FFFF on Bus and Valid

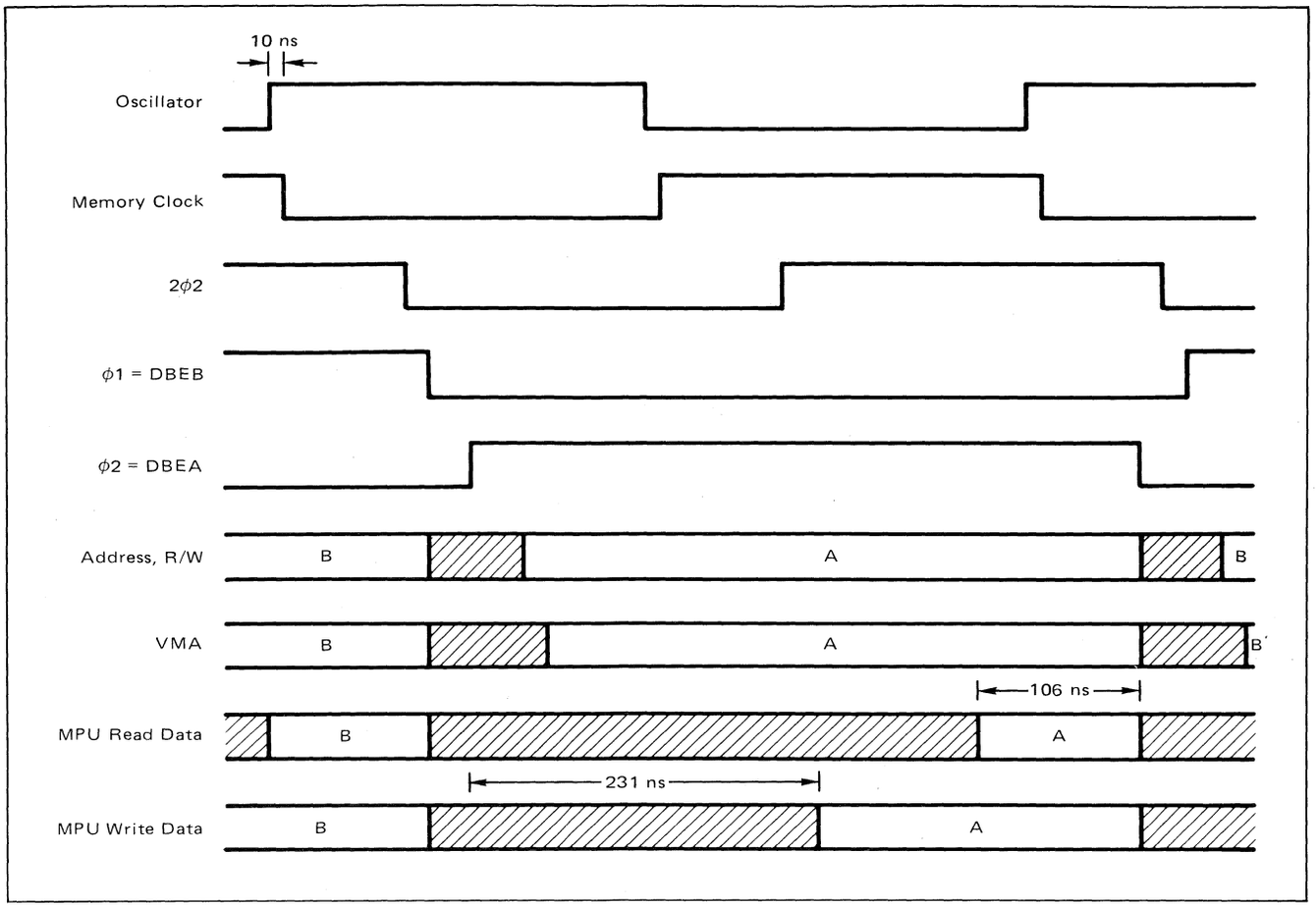

FIGURE 9C – System Clock Timing Diagram – Data Transfer To and From MPU

line to ground, asking the clock to stop in the 2ϕ2 portion of the cycle in progress. When the data has transferred, the Memory Ready line is brought high again. (Memory Clock is unaffected by this operation and keeps on cycling.) Therefore, when the Memory Ready signal is brought high, the bus clock output 2ϕ2 waits in the 2ϕ2, or high, portion of the cycle until the Memory Clock signal has completed its 2ϕ2 portion of the cycle. Figure 9B shows the timing when addresses $F000 to $FFFF (EXbug portion of memory) are on the bus and valid. This cycle stretching is done to allow for response time of the ROM, RAM, and ACIA.

The MPU clock is a derivative of the bus clock. This clock takes the 2ϕ2 signal from the bus clock section (Figure 8) and uses the negative going edge of 2ϕ2 to toggle a flip-flop, dividing the bus clock by two. The clock outputs of this circuit produce non-overlapping ϕ1 and ϕ2 signals. These signals correspond to the clock inputs to MPU A (MPU B uses ϕ2 as its ϕ1 input and ϕ1 as its ϕ2 input). The ϕ1 and ϕ2 signals used to drive the MPUs are from the outputs of an NMOS address line driver (MC3459). Although these clock signals don't meet the worst case specification for the MPU's clock inputs, they have been found to work satisfactorily.

Since the system operates at 2 MHz, the timing relationships between the MPU and bus must be carefully analyzed. The MPU clock generates non-overlapping ϕ1 and ϕ2 signals. In this generation, the ϕ2 clock is held low six gate delays longer than the ϕ1 clock signal so the ϕ2 clock is high for a shorter time than ϕ1. Figure 9C shows the timing necessary for data transfer to and from the MPU. (The timing diagram is looking at the bus after the typical delay times on the dual processor module have been included.)

BUS INTERFACE

The bus interface associated with each MPU will be enabled only during the ϕ2 portion of that MPU's clock cycle. Since the MPUs operate on opposite phases, each set of drivers/receivers will be enabled only on half of a clock cycle in a non-overlapping fashion. An MPU control signal, Bus Available (BA), is also used to enable the bus interface. The BA signal, when high, indicates the MPU has stopped (either by a Wait instruction or the $\overline{\text{Halt}}$ line going low) and the bus is available. This signal disables all the bus drivers of that MPU during its normal ϕ2 cycle.

The bus interface (Figure 10) is divided into two sections: The address drivers and the data drivers/receivers. The address drivers buffer the 16 address bits, R/W and VMA. These drivers are enabled during each ϕ2 clock of that MPU unless the BA signal is high, then the drivers are left in the high-impedance state. When the bus is available, the VMA' (see Figure 11) and VMA are normally

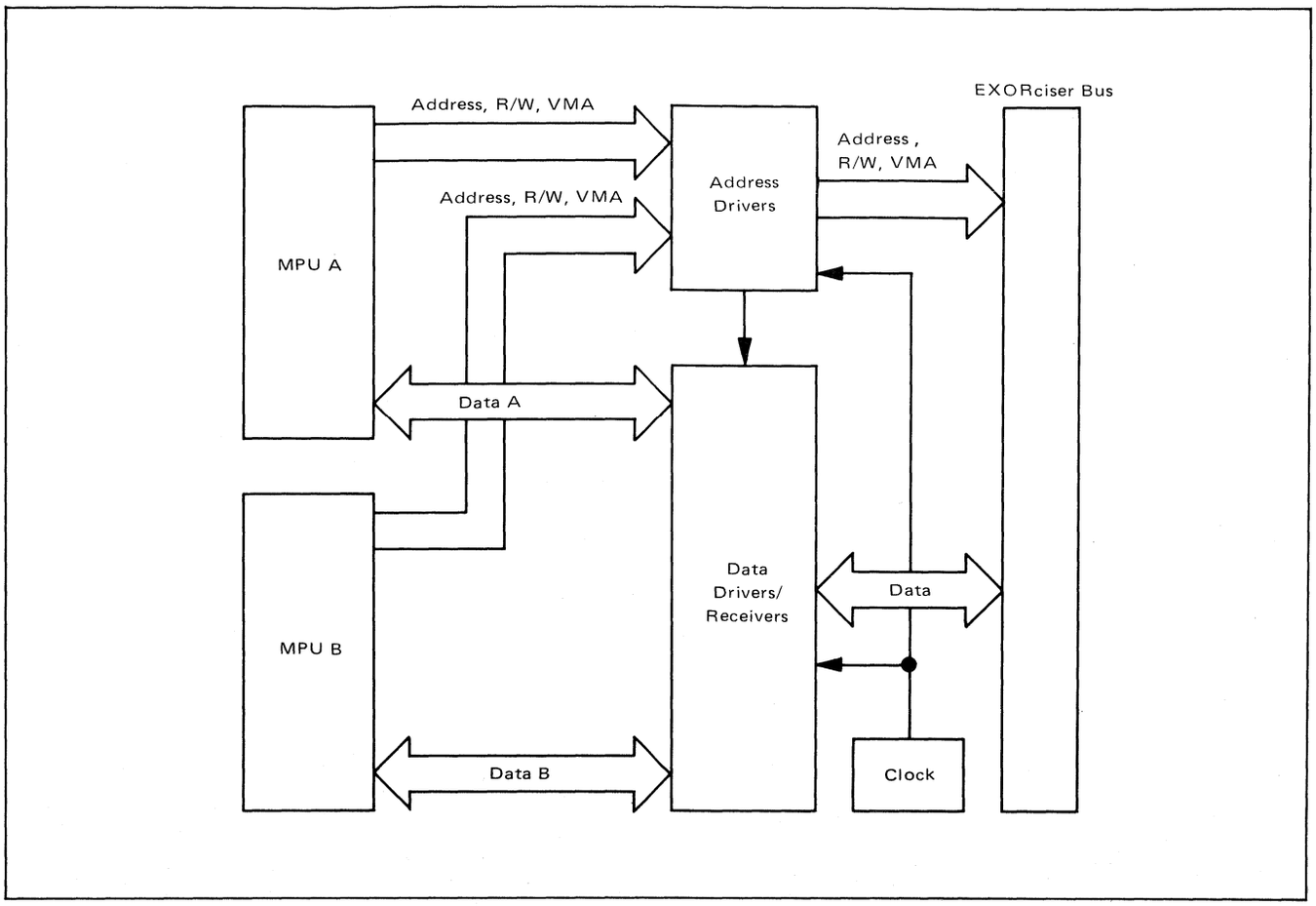

FIGURE 10 – Bus Interface Block Diagram

held at a "0" level. If some peripheral (i.e., Direct Memory Access) needs the bus, the VMAEXT signal must be used to get VMA on the bus to go to a "1" level. Here the peripheral will raise to a "1" level the VMAEXT line when the valid address is on the bus for data transfer.

The data drivers/receivers are bidirectional three-state devices. To enable either the drivers or receivers, it requires a combination of the R/W, VMA, BA and $\phi 2$ signal for the MPU being buffered (Figure 12). The bus drivers are enabled when the MPU's clock is in its $\phi 2$ portion, R/W is low indicating a write function and BA is low. The bus receivers are enabled when the MPU's clock is the the $\phi 2$ portion, R/W is high indicating a read function, VMA is high indicating the address is valid and BA is low.

DEDICATED MEMORY AND I/O

To minimize the logic needed, the two MPUs and the dedicated memory and I/O were incorporated in one module for the EXORciser system. For each MPU, this will eliminate some of the data bus interface, since the ACIA, RAM, and ROM data lines may be wired directly to the respective MPU. In addition, by placing both A and B sides of the memory and I/O on one module, the address decoding redundancy may be eliminated.

Table 1 shows the address decoding necessary to uniquely decode each component (ACIA, PIA, RAM and ROM) as to its bus address. This addressing is enabled with a signal called $\overline{FCXX}$, where address bits A10 to A15 are at a logic 1 level. Also, in the enabling of each component is the clock's $\phi 1$ or $\phi 2$ signal, defining which MPU is talking to the bus.

The first two machine cycles after an MPU has been reset, the ROM is enabled and the RAM at the restart vector address is disabled. This offset is added to the ROM so the restart may vector to the proper address. After the first two cycles, the ROM is then addressed in its normal memory location (Figure 13).

The ROM also contains the control character necessary to program the ACIA. EXbug programs the ACIA according to the speed of the terminal, so when a teletypewriter is connected (110 baud), the ACIA is programmed for 1 start bit, 8 data bits, and 2 stop bits. Otherwise, the ACIA is programmed for 1 start bit, 8 data bits, and 1 stop bit. To indicate when a teletypewriter is connected, the TTY input line is grounded (Figure 14).

The TTY and terminal interface is shown in Figure 14. This circuit supports a 20 mA loop for TTY interface and a standard RS-232C terminal interface. Included is a bit rate generator to supply both ACIAs with the proper baud rate for the teletype or terminal connected.

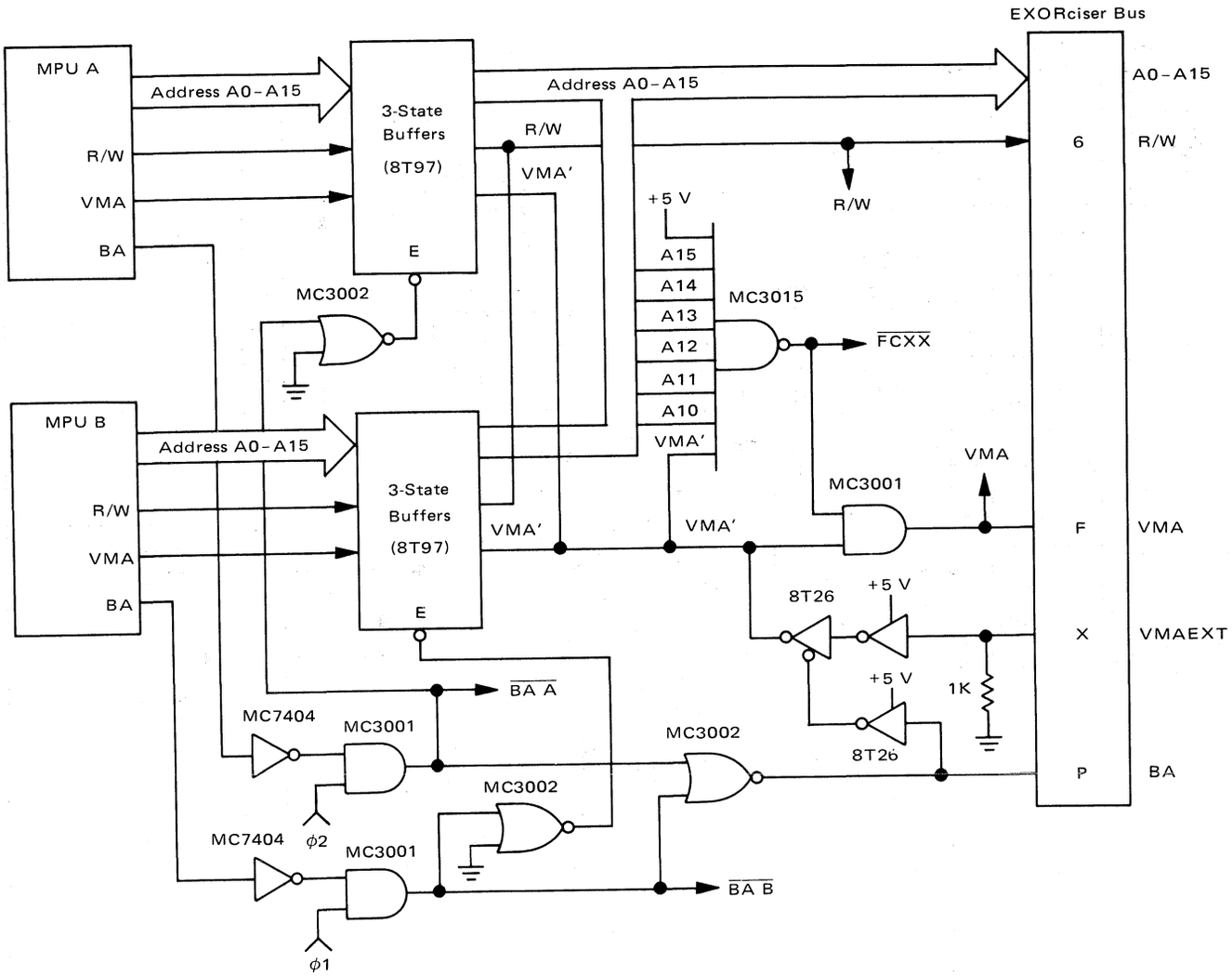

FIGURE 11 — Address Driver Logic

TABLE 1
Address Decoding
(Address Bits A10 to A15 = 1)

Device	Addresses	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
RAM 1	$FF80 — $FFFF	1	—	1	X	X	X	X	X	X	X
RAM 2	$FF00 — $FF7F	1	—	0	X	X	X	X	X	X	X
ROM	$FCFC — $FCFF	0	—	—	—	—	—	1	1	X	X
PIA	$FCF8 — $FCFB	0	—	—	—	—	—	1	0	X	X
ACIA	$FCF4 — $FCF5	0	—	—	—	—	—	0	—	—	X

Where: 1 = High Enable
0 = Low Enable
X = Address/Register Select
— = Don't Care

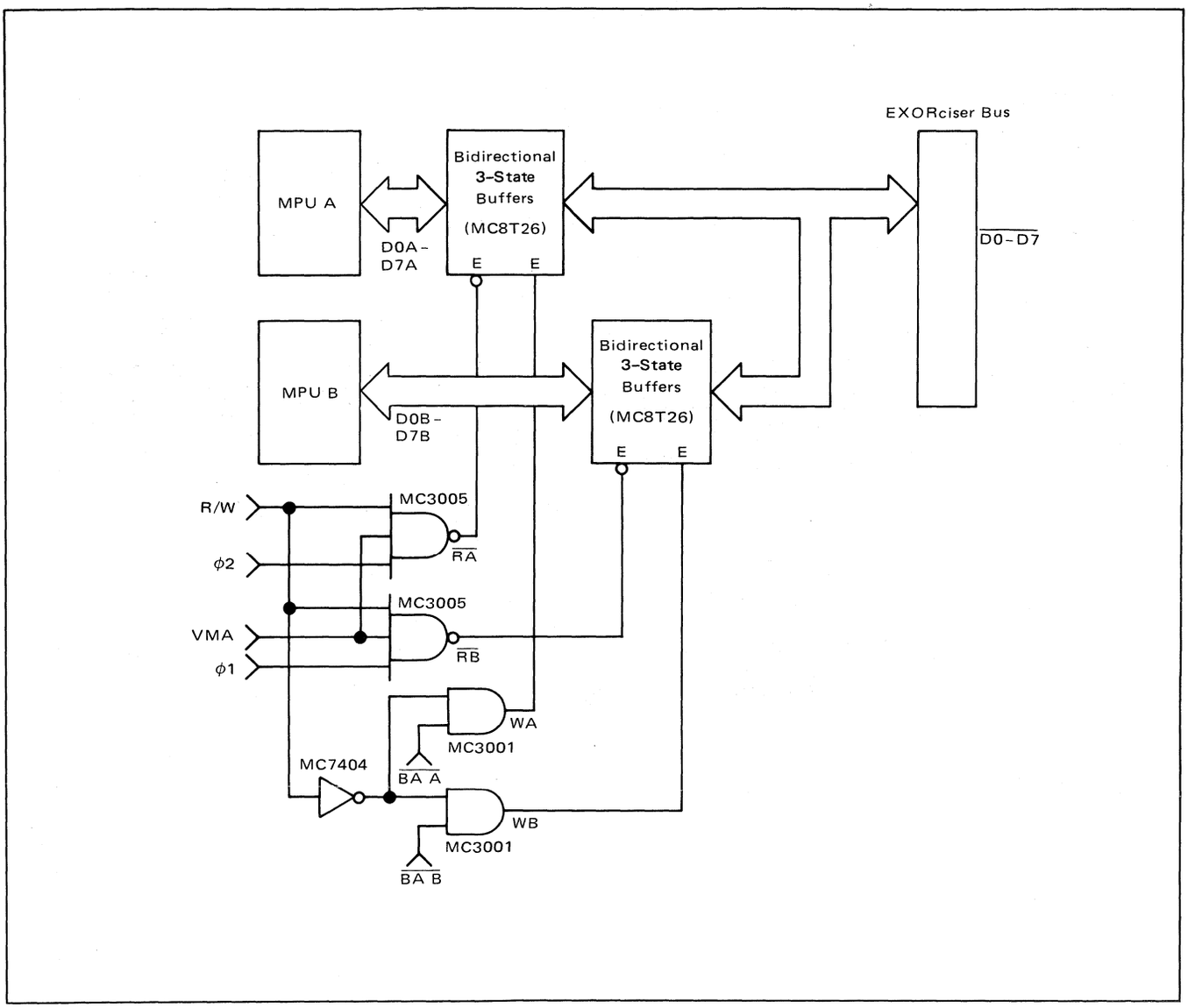

FIGURE 12 — Data Drivers/Receivers

MODIFICATIONS TO THE EXORciser

A few modifications to the basic EXORciser system were needed for the system to operate properly.

First, the MPU module supplied in the EXORciser must be removed. This is obvious, since the dual-processor system contains both of the MPUs for the system.

Second, U20 (MC7473) on the Debug Module must be removed and a short inserted between pins 13 and 11. This will disable the power-up/restart sequence of the module.

SUMMARY

The system as shown has utilized the following features of dual processing.

1. Share common ROM. Both processors share the common EXbug program ROM.

2. Dedicated block of memory. Both processors have exclusively at the same addresses an ACIA, RAM, and ROM.

3. One bus structure. This system shows how a one bus dual-processor system may be implemented. This also includes only one set of power supplies.

4. One clock circuit. This system uses one clock which eliminates duplication of hardware.

5. Both processors operate at 1 MHz. Except when executing the EXbug control program, both processors operate at 1 MHz doubling the overall system throughput. In this system, the EXbug program is used only for loading programs and program debugging. Once the dual-processor system's software has been debugged, the EXbug program is needed only to load the programs and initialize program execution.

After the modifications to the basic EXORciser were completed, the dual processor module was inserted into the EXORciser, terminals connected, and powered up. Both MPUs, automatically reset, started execution of the EXbug progam. A short program was executed in the common user memory to insure both processors operate at the 1 MHz speed.

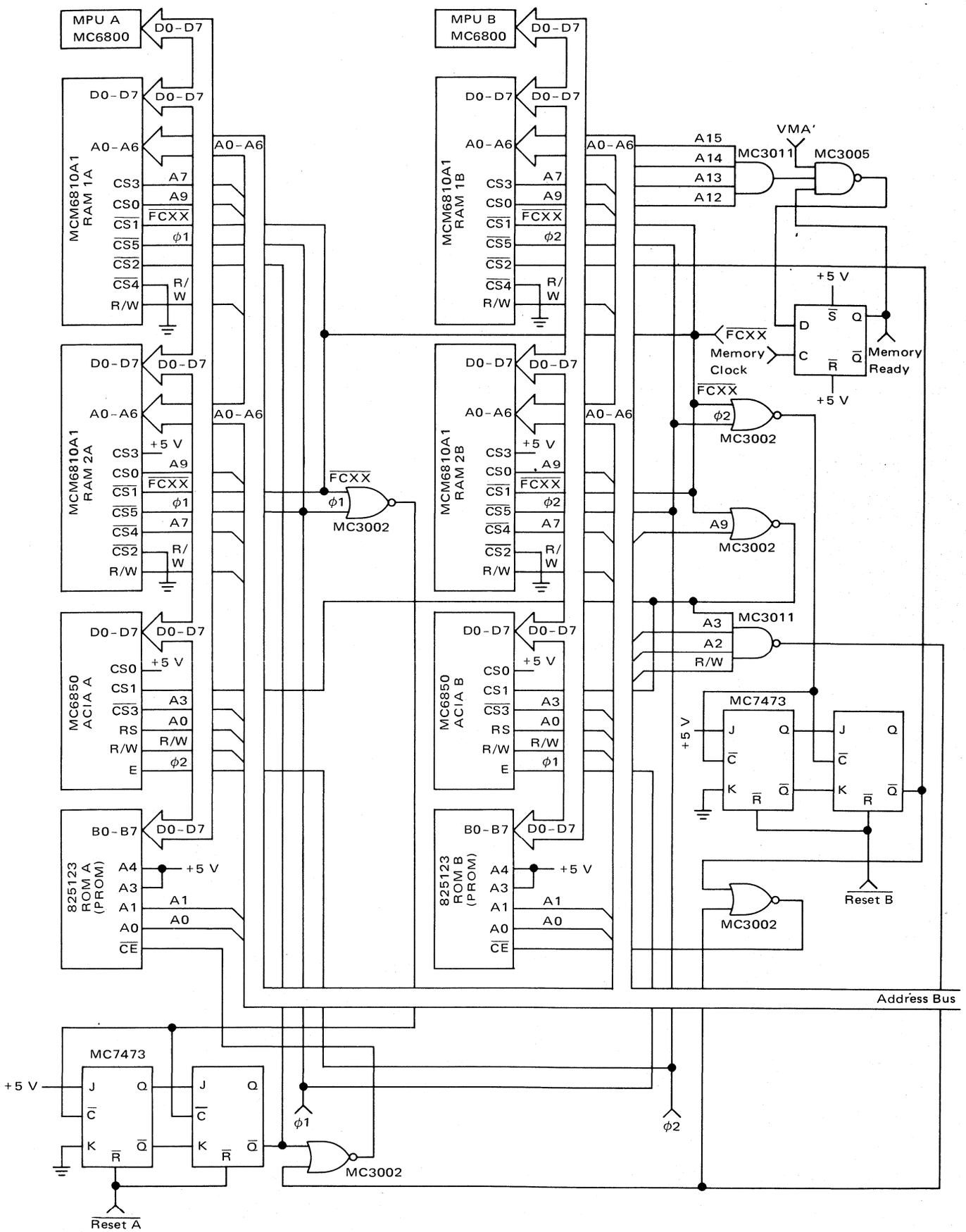

FIGURE 13 – Dedicated Memory and I/O

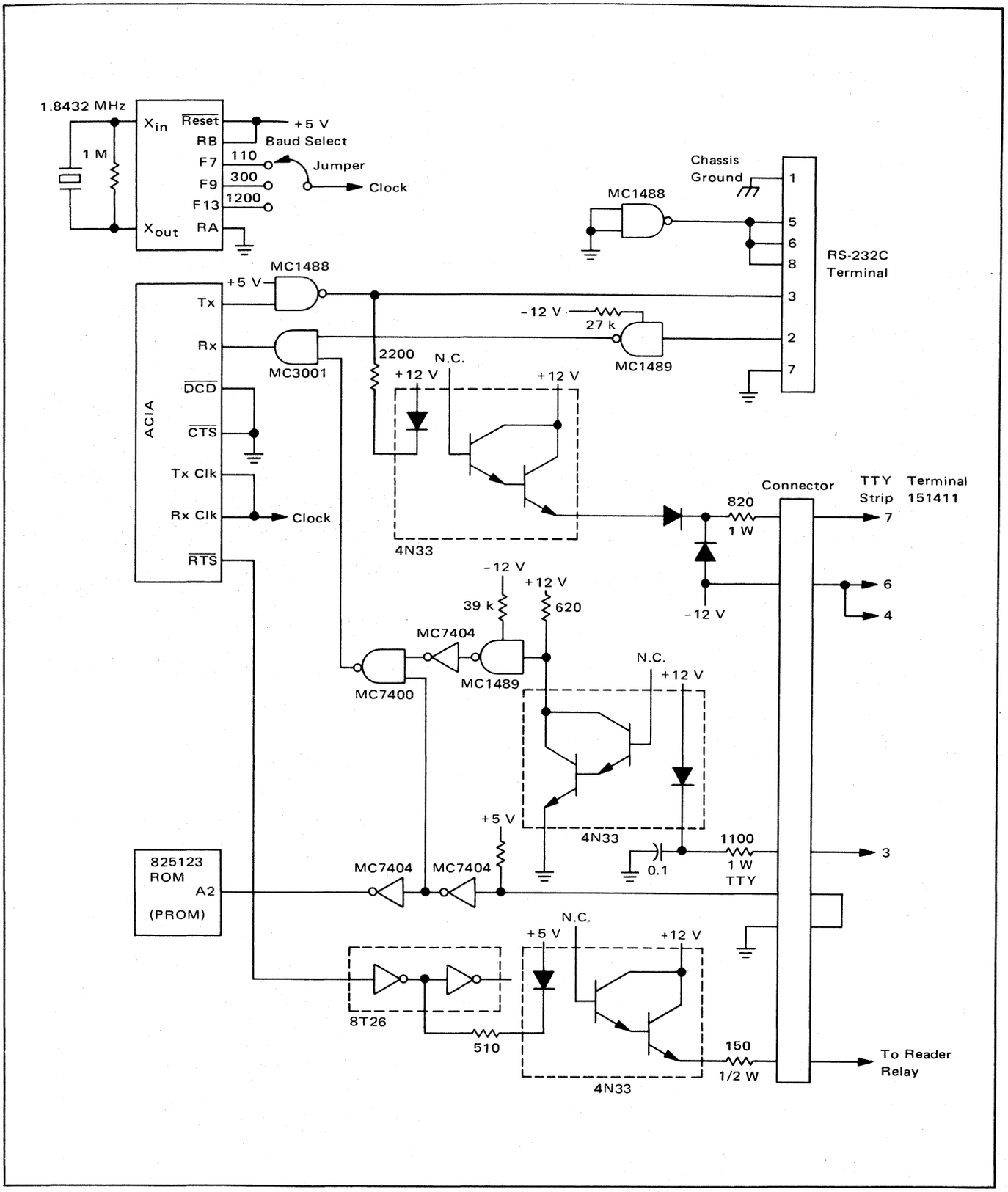

FIGURE 14 – Teletypewriter and Terminal Interface

MOTOROLA Semiconductor Products Inc.

Printed in Switzerland