AN-771
Application Note

ADVANCED SEMICONDUCTOR DEVICES (PTY) LTD
P.O. Box 2944, Johannesburg 2000
3rd Floor, Vogas House
123 Pritchard Street/Corner Mool Street
Johannesburg
Tel. No. 23-2856

MEK6800D2 MICROCOMPUTER KIT SYSTEM EXPANSION TECHNIQUES

Prepared By
Wayne Harrington
Microprocessor Applications/Systems Engineering

The bus architecture of the MEK6800D2 Kit Microcomputer provides straightforward design options for memory or I/O port expansion. This note outlines techniques for interfacing an 8K or 16K memory array with the kit. A technique is also outlined whereby a data terminal-based ROM monitor such as MINIBUG may co-reside with the basic kit ROM JBUG Monitor. The resulting two-monitor system allows the user to switch between either the JBUG I/O port or the MINIBUG I/O port for moving data to and from RAM.

MOTOROLA Semiconductor Products Inc.

MEK6800D2 MICROCOMPUTER KIT SYSTEM EXPANSION TECHNIQUES

INTRODUCTION

The Motorola MEK6800D2 kit microcomputer system (hereafter referred to as MEK/D2) is a complete computer requiring only a +5 V power supply to begin microprocessor evaluation. It features a hexadecimal keyboard for data and command entry and seven-segment LED array for data display. In addition, the MEK/D2 provides an audio cassette I/O data transfer capability. Figure 1 presents a functional block diagram of the basic system. The intent of this note is to describe some useful system expansion techniques which exploit the architecture of MEK/D2 computer. This note is intended to supplement the information provided in the MEK/D2 manual and is divided into sections which discuss memory expansion, data I/O port expansion and expanded system application considerations.

Off-board memory expansion involves only minor changes in addressing and control logic plus certain elementary control-handshake logic to support both dynamic memory arrays and provide MPU control for slow memory arrays.

The inclusion of an I/O port to add data terminal communication in addition to the keyboard module function is accomplished by inserting control logic which converts MEK/D2 into a dual-monitor microcomputer system. This modification allows the basic MEK/D2 JBUG monitor ROM and its ACIA to coreside with a MINIBUG ROM/ACIA combination. The JBUG-ROM/ACIA pair support keyboard and audio cassette data I/O transfer while MINIBUG, along with its ACIA, supports RS-232 or current loop-configured data terminals. Each ROM/ACIA pair may be manually initialized or software-accessed from the user program.

The capability to select, initialize, or address locations in either monitor ROM at will provides useful system application benefits. These include moving data between various storage media, directly addressing proven subroutines in either ROM from user program and manually selecting either monitor as desired to exploit the most useful commands of each during a software or system development phase. These modifications convert the MEK/D2 into a powerful software development tool.

RANDOM ACCESS MEMORY EXPANSION

Functional Design

The basic MEK/D2 Microcomputer Module provides for a maximum of 512 bytes of On-Board static RAM. Expansion for additional memory is accomplished by providing address and data bus buffers as well as some Off-Board control logic.

Figure 2 presents a functional block diagram summary of the supplemental logic necessary to support Off-Board memory expansion. Shaded blocks represent logic available with the basic MEK/D2 system. This convention holds for all schematics and diagrams in this note.

Certain static RAMs require up to 100 ns of data hold following chip deselect. The 10 ns data hold specified for the MC6800 MPU is insufficient to meet this requirement. The data bus enable (DBE) stretch network shown must be added if this type of RAM is utilized in the Off-Board expansion array. The Memory Control Handshake Logic provides control and timing signals between logic resident on Off-Board memory systems and the MPU clock module. Data transceivers, with a control logic block, are required to buffer bidirectional data to the Off-Board memory array as shown. The block labled "Array Select Decoder" represents logic for converting high-order address decode signals to Memory-Block enabling signals. These activitate either the On-Board or Off-Board array within the appropriate addressing range of a memory reference instruction.

Logic Design

Figure 3 shows a network which exploits the propagation delay of non-inverting CMOS buffers to generate a "stretched" $\phi 2$ for processor and peripheral data bus enable. This network delays the falling edge of DBES approximately 125 ns with respect to DBE. This meets the data hold time requirement of most static RAMs. Trim capacitor C_t may be added for fine adjustments to account for device variations in accordance with the equation shown.

Memory Control Handshake Logic is shown in Figure 4. Clocked latches E17A and E17B provide signals to control either dynamic memory refresh or slow-memory access on a synchronous basis with respect to MPU timing.

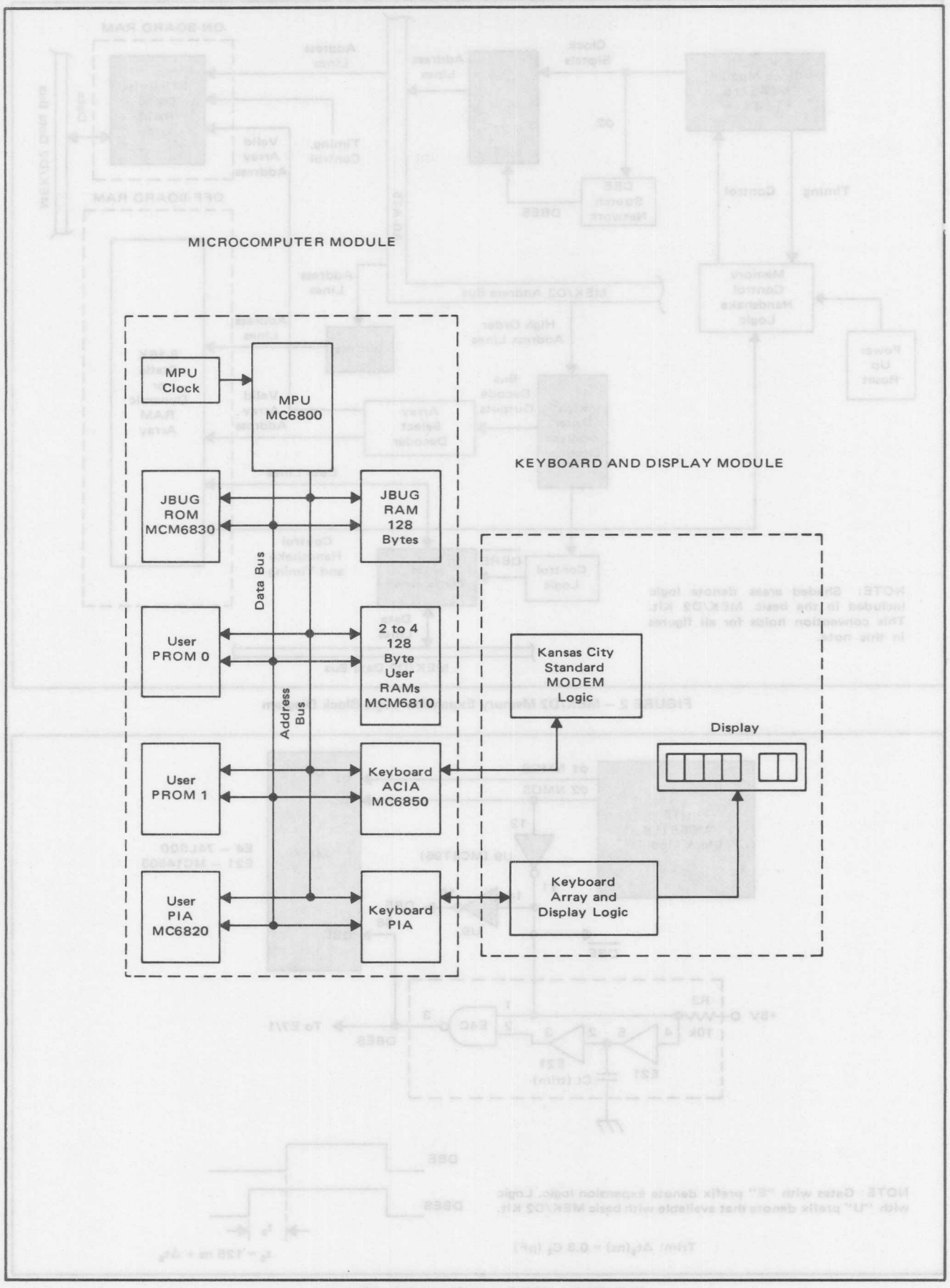


FIGURE 1 – MEK6800D2 Block Diagram

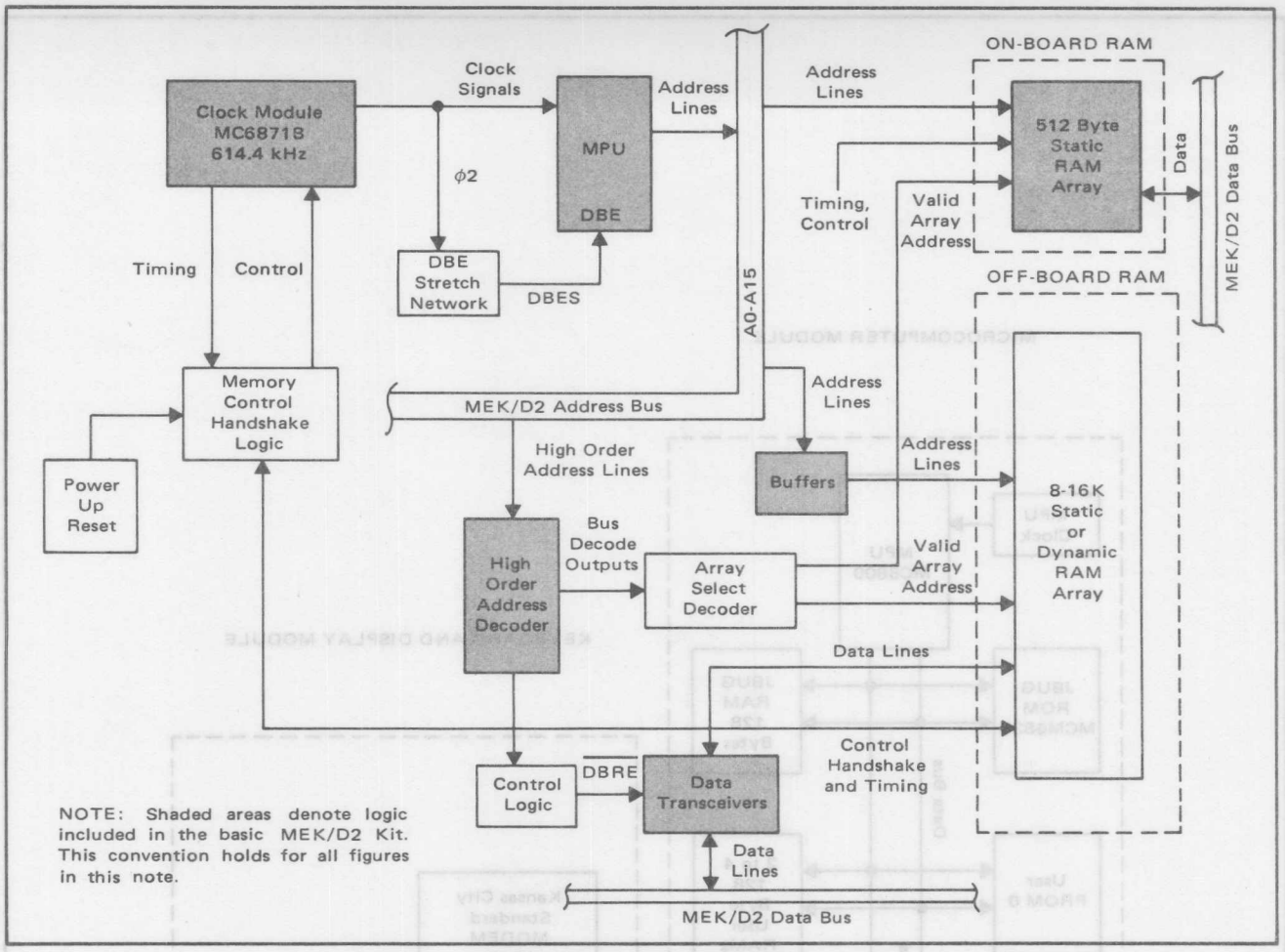


FIGURE 2 – MEK/D2 Memory Expansion Logic Block Diagram

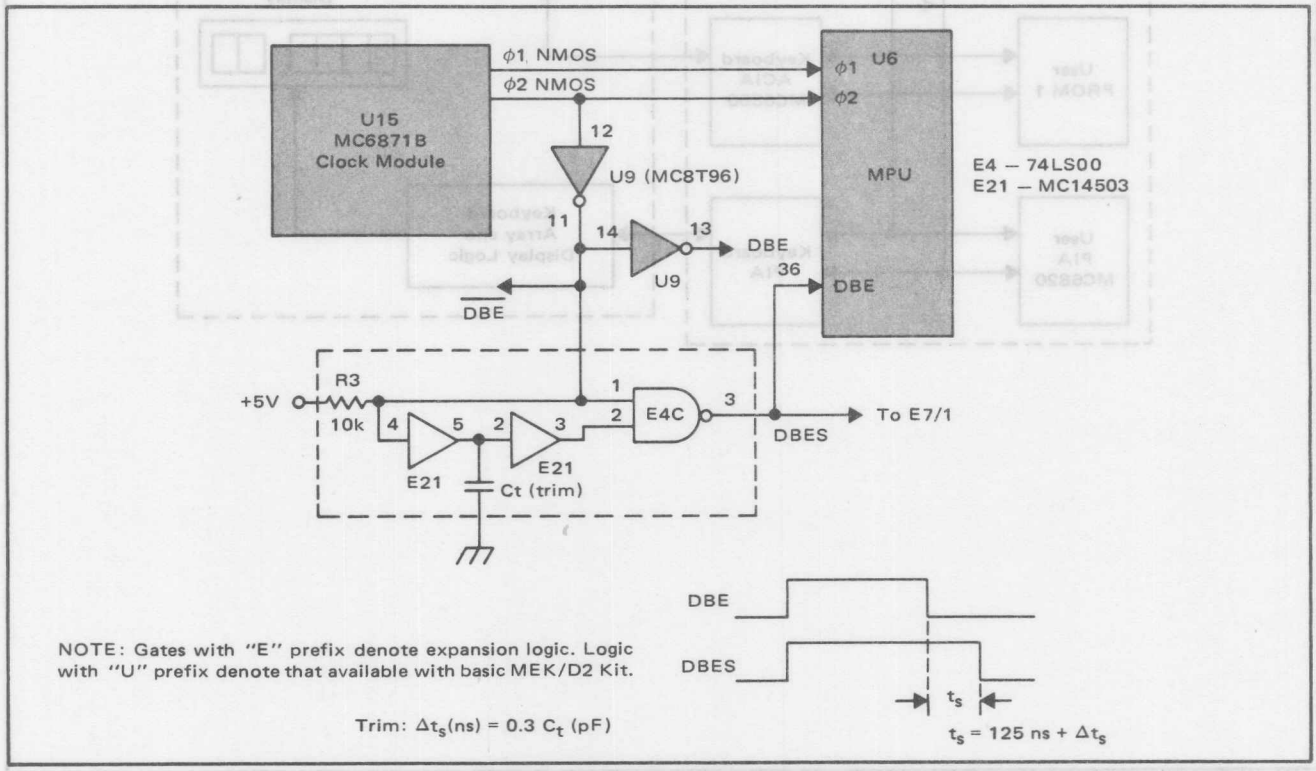


FIGURE 3 – DBE Stretch Network for Memories with Non-Zero Data Hold Time Requirement

Dynamic memory cells store data in the form of electronic charge on the capacitance inherent in MOS transistor junctions. This charge must be periodically "refreshed." This is accomplished in most dynamic memories by performing a "dummy" read or write operation on each cell. In the case of the 8K Dynamic RAM Module (MEX6815-3), complete memory refresh is accomplished by a modified internal read operation on each of 32 columns once every 64 μs. (memory system organization is 128 rows X 32 columns). The columns are accessed by an address multiplexer which is pulsed by the Refresh Grant (RG) handshake signal once every 64 μs.

The power-up reset network, composed of E9 and E4D, sets latch E17A on power-up to insure a proper initialization of the refresh-handshake logic. E9 also automatically initializes the MPU system on power-up by pulsing E6/12.

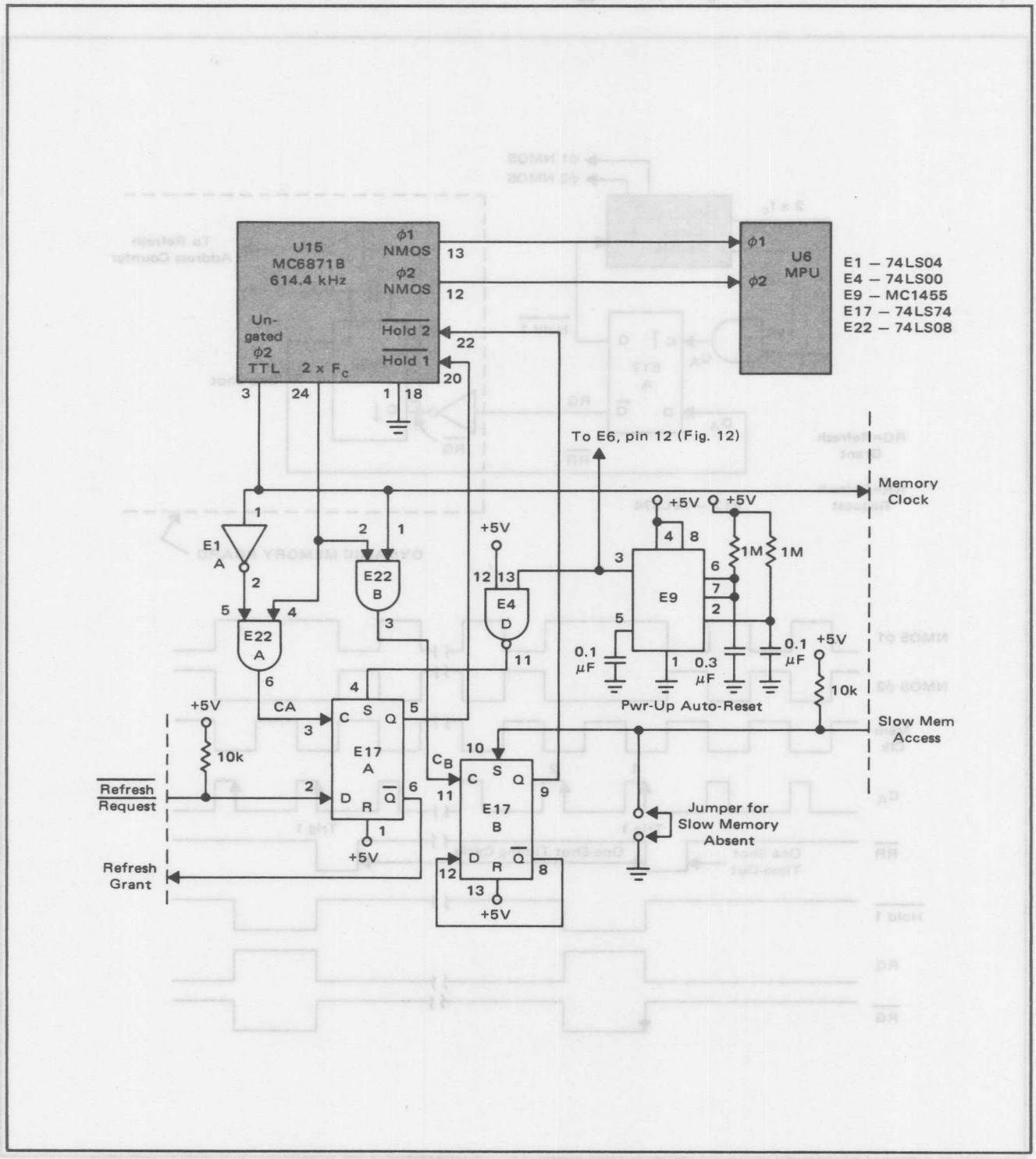


FIGURE 4 – Memory Control Handshake Logic

Figure 5 presents an example of refresh-handshake timing between latch E17A and logic on a dynamic memory system. The latch is clocked by AND-gate output C_A. The first low-to-high transistion of C_A (pulse 1) following time-out of the refresh-period one-shot (8602) samples the logical zero state appearing at the D input of E17A. This state and its complement are transferred by the rising edge of C_A to the Q and Q outputs of E17A as the signals Hold 1 and Refresh Grant (RG), respectively. The resultant falling edge of RG retriggers the 8602 to start a new timing cycle as shown in the diagram. This action returns the Request Refresh (RR) signal to logical one. This is sampled by the low-to-high transition of C_A, which returns Hold 1 high. The resulting Hold 1 signal applied to the H1 input of the MPU clock module is correctly phased to meet H1 set-up and release time requirements and "freezes" the MPU clock in the phase relation shown. The resulting RG pulse automatically increments the refresh address counter for the next refresh cycle.

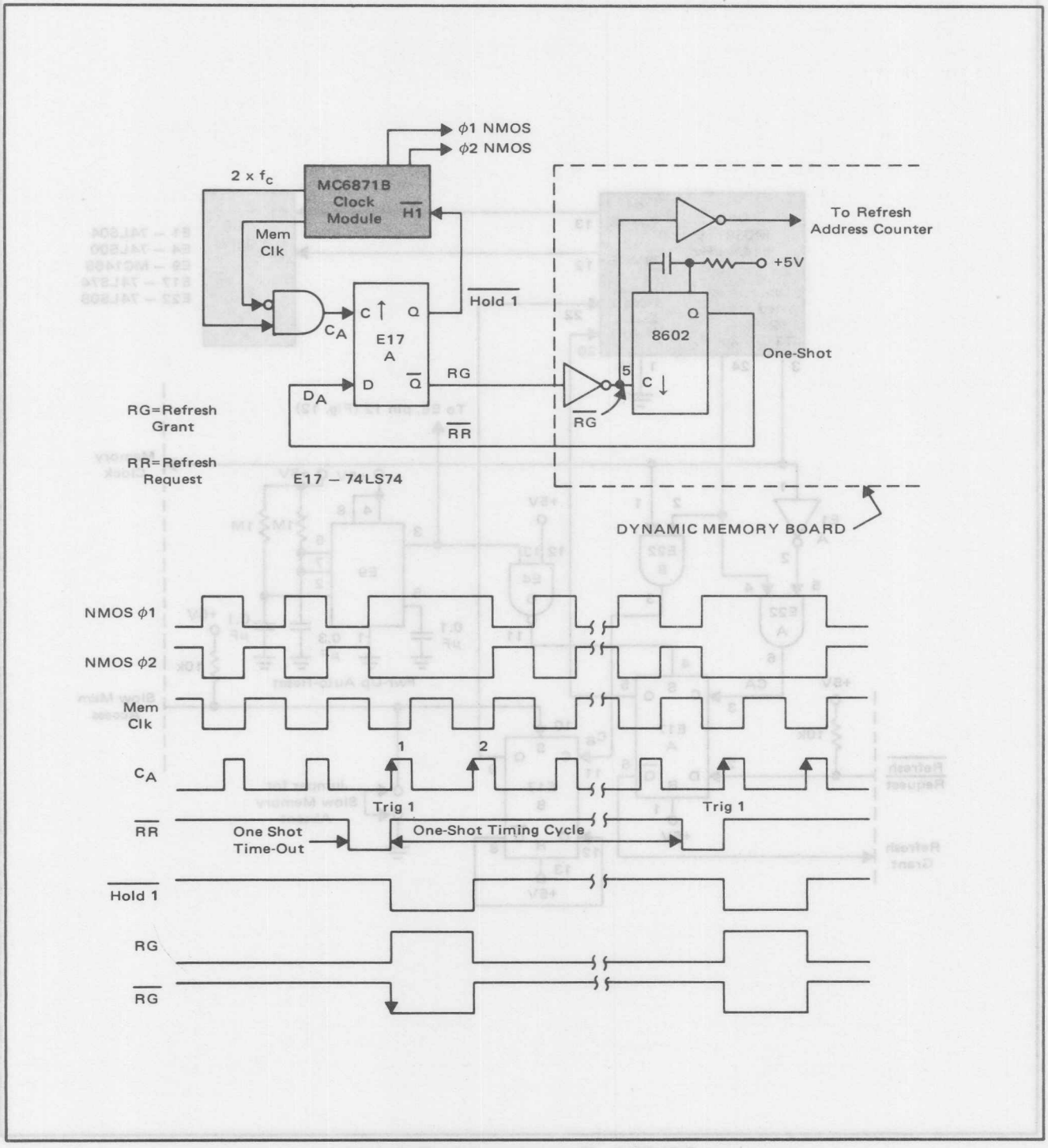


FIGURE 5 – Memory Refresh Handshake Logic and Waveforms

Figure 6 presents a typical example of slow memory control with handshake-timing between latch E17B and memory control logic on a slow memory board. Slow memory control signals are required to account for memories (or peripherals) whose access times are in the range of 540 to 4500 ns. The control signals provide proper slow memory data acquisition by freezing the MPU clock. This effectively allows the MPU to "wait" for memory data to return and still meet the maximum MPU bus memory access time specification of 540 ns. The access time upper limit of 4500 ns is determined by the maximum allowable clock phase 2 high time of 4500 ns. High times in excess of this value will introduce data loss within the MPU dynamic registers. These registers use the MPU clock for refresh, just as with memory cells in dynamic RAM. The sequence of events for a slow memory access are described in the waveform timing diagram. The array decoder output, AS, goes high following the low-to-high transition of $\phi 2$ for a memory reference within the addressing range of the array. The high-state of AS (or Slo Mem Acc) applied to the asynchronous-set input of latch E17B releases the hold-set condition on the latch and allows it to be clocked by the first C_B pulse. This forces Hold 2 (Q) low, which freezes the MPU clock in the phase relation shown. Hold 2 is returned high with the low-to-high transition of the next C_B pulse, since latch E17B is connected as a toggle flip-flop. Since Hold 2 is returned to logic 1, the clock is allowed to resume as shown, and the cycle is complete. The resulting freeze of the clock cycle with $\phi 2$ high and $\phi 1$ low adds a 1-clock-cycle delay to the normal access time available. This scheme may be extended with additional counters and logic in place of the toggle flip-flop to hold the clock a multiple-number of MEM Clk cycles for very slow memories. The total hold time must not exceed the 4500 ns maximum limit.

A key integrated circuit for generating system bus chip-select or enabling signals in the MEK/D2 is the high-order address decoder U11 – a 2-line to 4-line decoder/demultiplexer. This logic element decodes the three-most-significant bits, A15-A13, of the address bus in accordance with the following truth table.

A15	A14	A13	Bus Enable Term	Comments
0	0	0	$\overline{RAM}$ = 0	Enables 512 byte array On-Board RAM
0	0	1	$\overline{2/3}$ = 0	Enables 8K range of user stack
0	1	0	$\overline{4/5}$ = 0	Enables 8K range of user stack
0	1	1	$\overline{PROM\ 1}$ = 0	Enables user PROM located at 6000^*_{16}
1	0	0	$\overline{I/O}$ = 0	Enables ACIA located at 8008_{16}
1	0	1	$\overline{STACK}$ = 0	Enables 128 byte RAM used by JBUG monitor
1	1	0	$\overline{PROM\ \emptyset}$ = 0	Enables user PROM located at $C000_{16}$
1	1	1	$\overline{ROM}$ = 0	Enables JBUG ROM located at $E000_{16}$

*Denotes Base 16 (hexadecimal) number

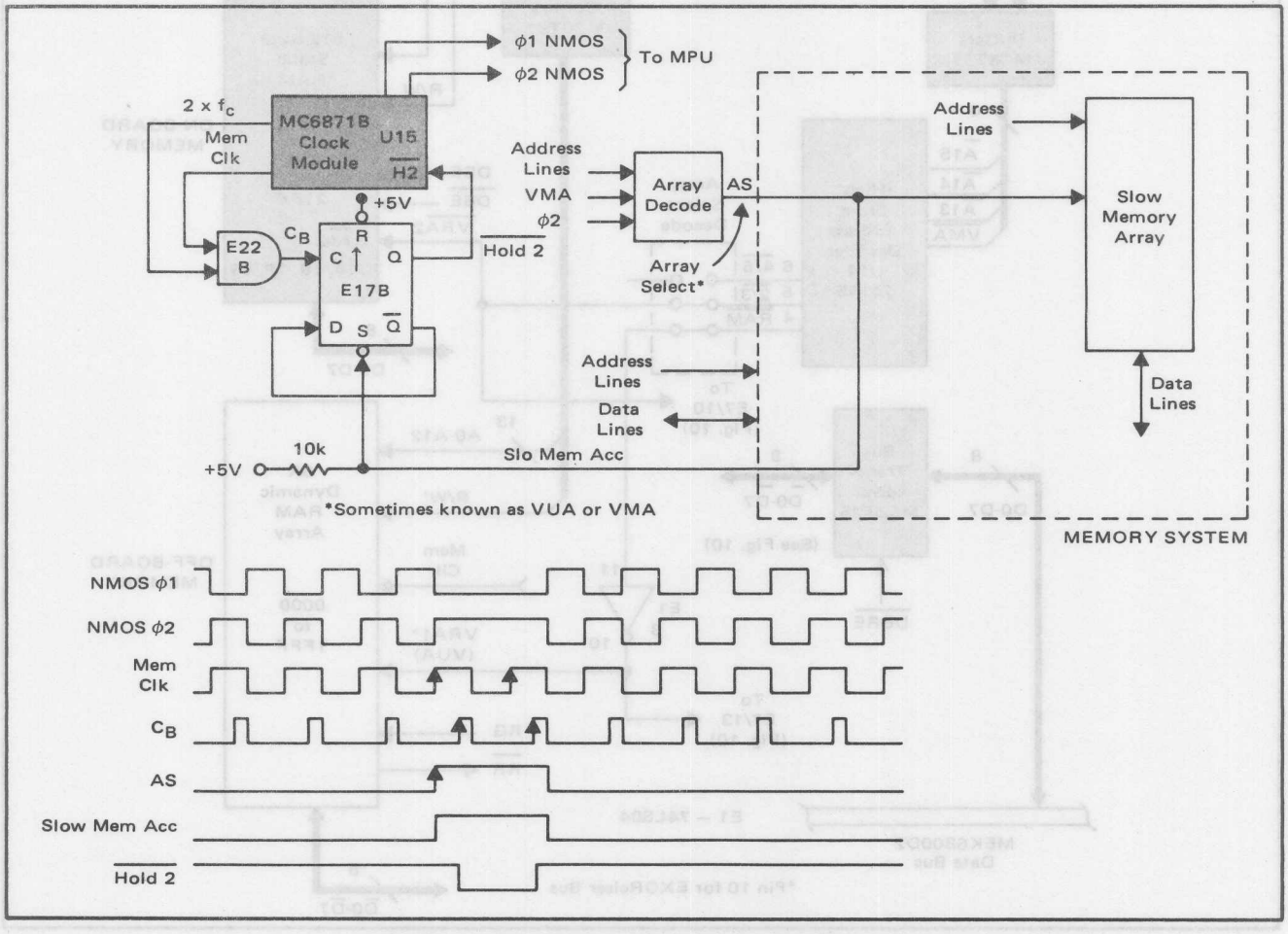


FIGURE 6 – Slow Memory Handshake Logic and Waveforms

This scheme divides the 64K addressing range of the MPU into eight 8K blocks. The 512 byte static RAM array is placed in the bottom 8K range, the next two 8K blocks are reserved for expansion RAM, the fourth contains a user PROM, etc.

Figure 7 presents the Bus Peripheral Allocation Map for the basic MEK/D2 system. Exact address boundaries of the bus peripherals described in the decoder truth table are defined in this map. The decoder output terms which enable the first three 8K blocks of memory, beginning with address zero, are RAM, 2/3 and 4/5. Inspection of the map shows that within the first addressable 8K block, only 512 bytes are dedicated to static RAM. This produces a memory addressing "gap" in the range 0200 to 1FFF as far as continuous addressing within the first 8K block is concerned. This problem may be solved by additional decoding of the three RAM select signals above so as to place an 8K expansion RAM in the first 8K addressing block, or a 16K expansion RAM within the first two 8K addressing blocks. The 512

	Address Ranges
JBUG ROM	E000-E3FF
User PROM	C000-C3FF
JBUG Stack/RAM	A000-A07F
Keyboard Module PIA	8020-8023
Keyboard Module ACIA	8008-8009
User PIA	8004-8007
User PROM	6000-7FFF
User RAM	4000-5FFF
User RAM	2000-3FFF
User Static RAM	0000-01FF

FIGURE 7 – MEK/D2 Bus Peripheral Allocation Map

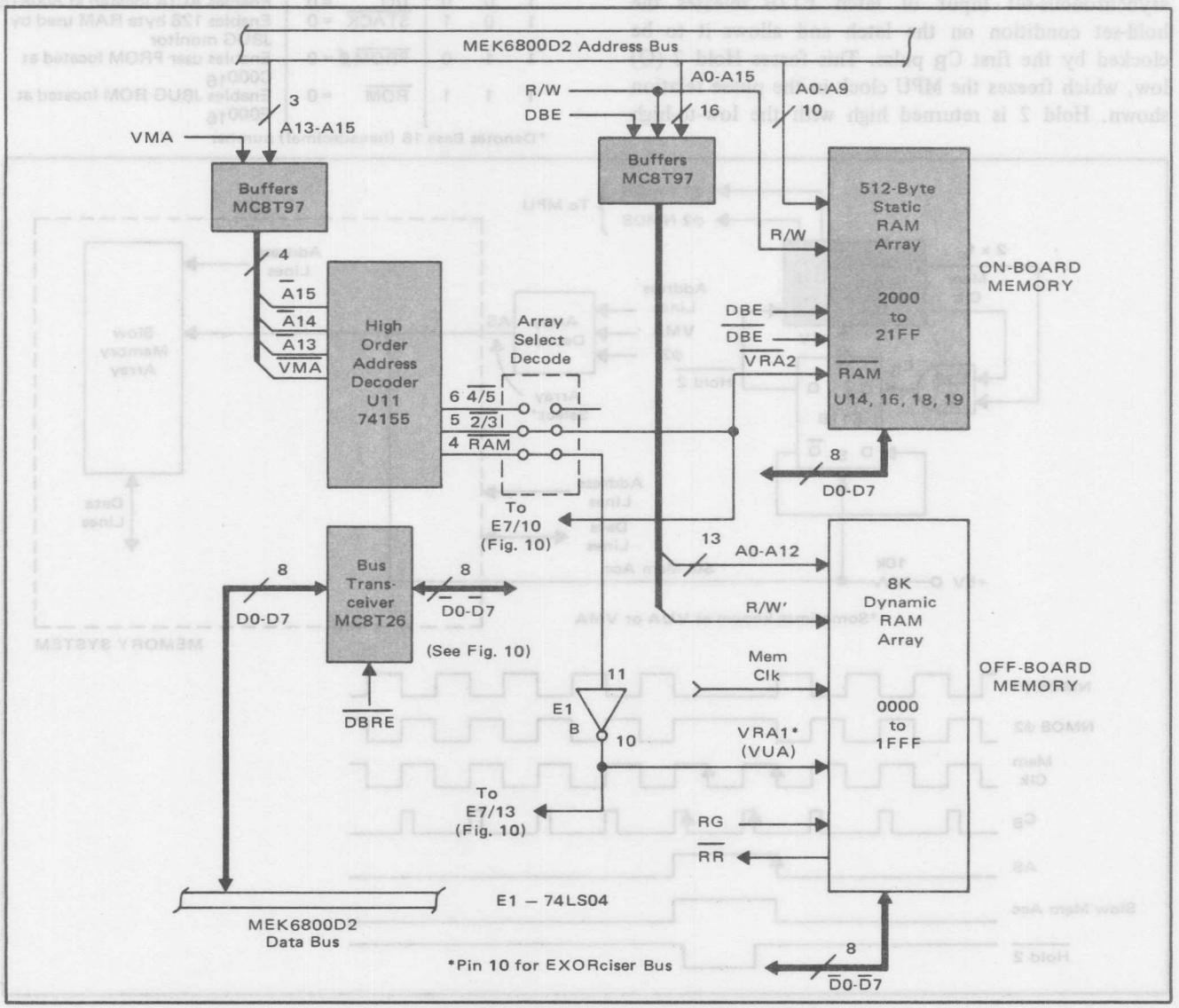


FIGURE 8 – Addressing for 8.5K Memory Configuration

byte static array is then placed in either the second or third block, respectively, "on top" of the expansion RAM. Figures 8 and 9 show the additional decode required to form either an 8.5K or 16.5K memory configuration. Control and Timing signals necessary to support these arrays are also shown.

Data flow direction to Off-Board memory is determined by the decode/control logic shown in Figure 10. This logic asserts DBRE (Data Bus Receive Enable) for any MPU read cycle involving Off-Board memory. This enabling scheme should be used with any additional Off-Board memory, whether static or dynamic.

Recent developments in semiconductor dynamic RAM system design have provided compact, cost-effective arrays such as the MMS68100 and MMS68103 produced by Motorola Memory Systems. These are available in 4K x 8, 8K x 8, or 16K x 8 size. The most notable feature of these memories is that the usual refresh-handshake logic, such as shown in Figure 4, is not required since refresh is processed by memory board logic during MPU phase 1.

I/O DATA PORT EXPANSION/MODIFICATION

Dual Monitor System – Functional Description

The basic MEK/D2 system with keyboard data entry and seven-segment light-emitting-diode display may be expanded to include a co-resident data terminal I/O capability which may be evoked manually or from user program. The software necessary to support data terminal operations is provided in firmware using a MINIBUG ROM. This ROM monitor co-resides with the JBUG monitor ROM supplied with the basic MEK/D2. ROM access and initialization is controlled by the logic shown in functional block diagram form in Figure 11. With this scheme, peripheral chip-select signals derived from the high-order address decoder (U11) are steered to the desired ROM-ACIA pair as a function of the state of the Chip Select Control signal, CSC. CSC is generated from either the manual ROM select switch (S_R) or by user-program command from the PIA. Control from user program automatically overrides the manual input but does not initiate an MPU reset cycle as does a manual

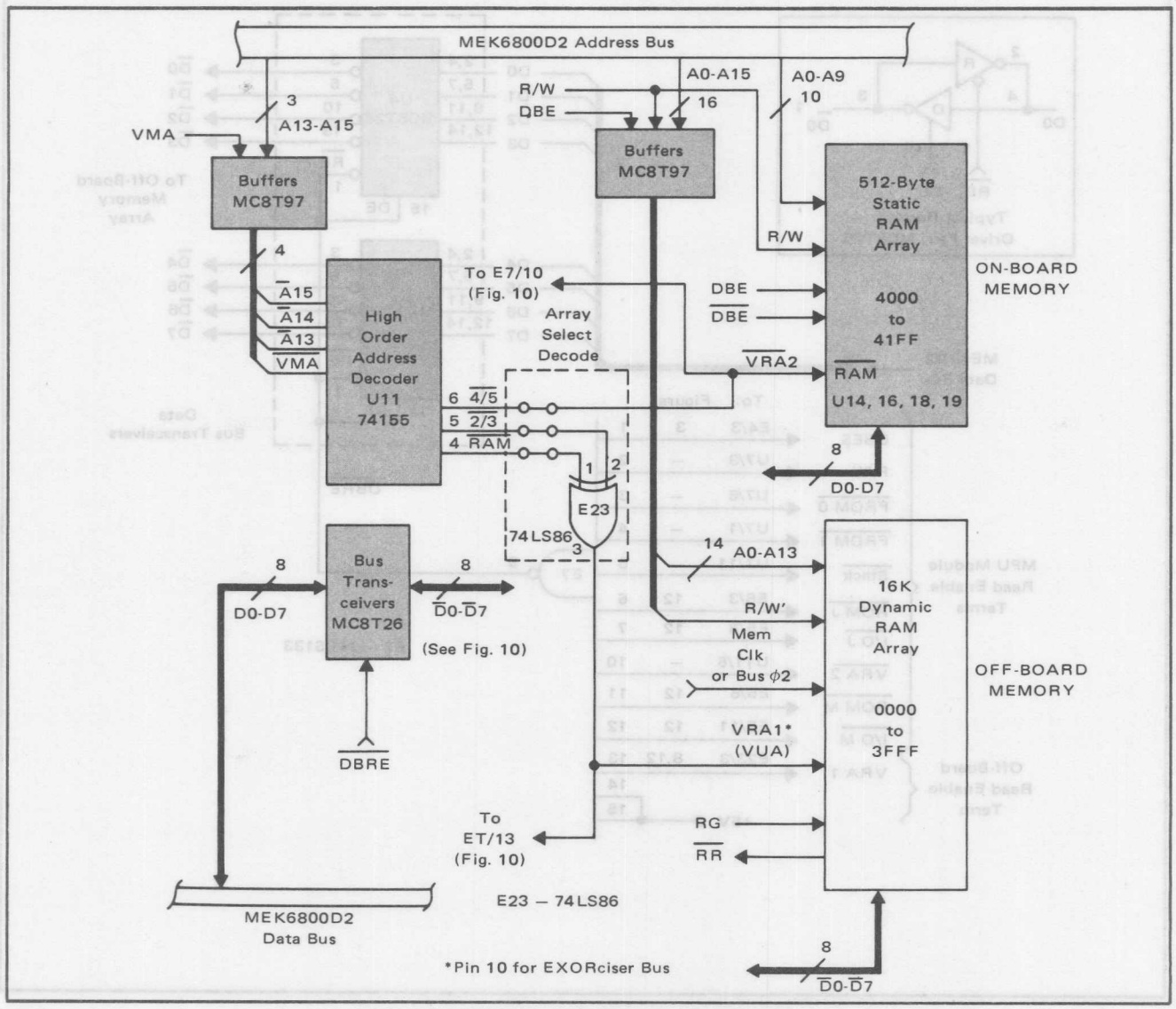


FIGURE 9 – Addressing for 16.5K Memory Configuration

select from S_R. With S_R in position J (for JBUG enable), the ROM and I/O chip-select signals (ROM and I/O are steered, respectively, only to the JBUG ROM or ACIA, while the MINIBUG ROM and ACIA are held deselected. The converse actions occur for S_R at position M (for MINIBUG enable). Each toggle of S_R generates an MPU Reset pulse via the State Change Detect Logic. This has the effect of automatically initializing each monitor ROM when manually selected. Nine standard data terminal baud rates may be derived from existing MEK/D2 logic and are used to provide transmit and receive clocks for the MINIBUG ACIA.

Logic Design

Logic realizations of the system functions depicted in Figure 11 are presented in Figures 12, 14, 15, 16 and 17.

Figure 12 shows the Chip Select Steering Logic, MPU Cycle-Sync Logic and State Change Detect Logic. Chip-select steering is accomplished by the network composed of gates E5 and E1C. The clocked-latch network (E3A – E3B) which generates the chip-select steering control signal, provides two design benefits. First, monitor switching occurs only after MPU reset is asserted and prior to a $\phi 2$ cycle, thus assuring that data will not be erroneously written or read as a result of a manual monitor select. In addition, latch E3A, under the control of the PIA, provides an asynchronous-override to the manual select switch control. This feature allows direct access to subroutines in either ROM or addresses in either ACIA from the user program. A subroutine to accomplish this access is described in a following section.

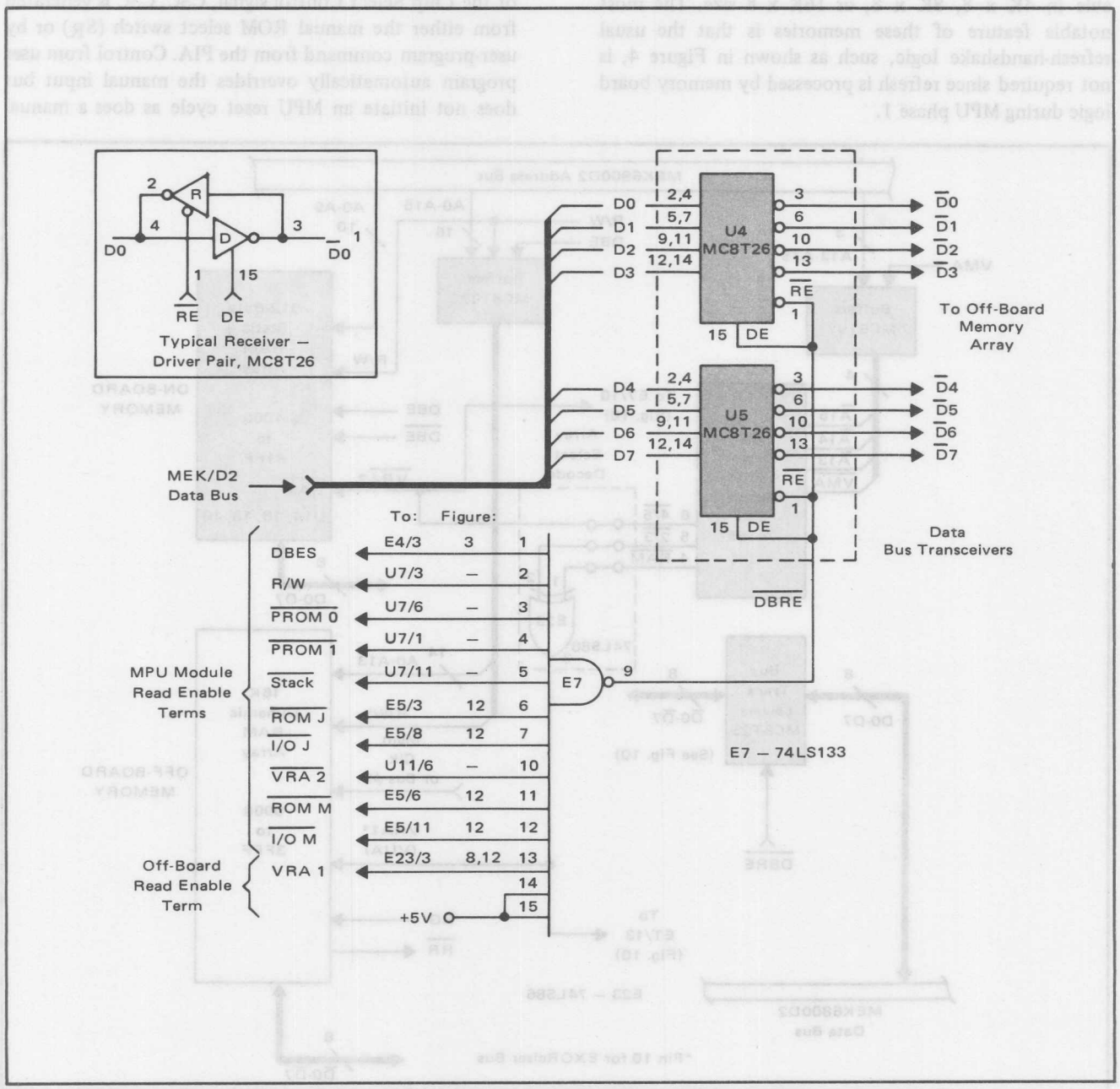


FIGURE 10 – Data Bus Expansion Control Logic

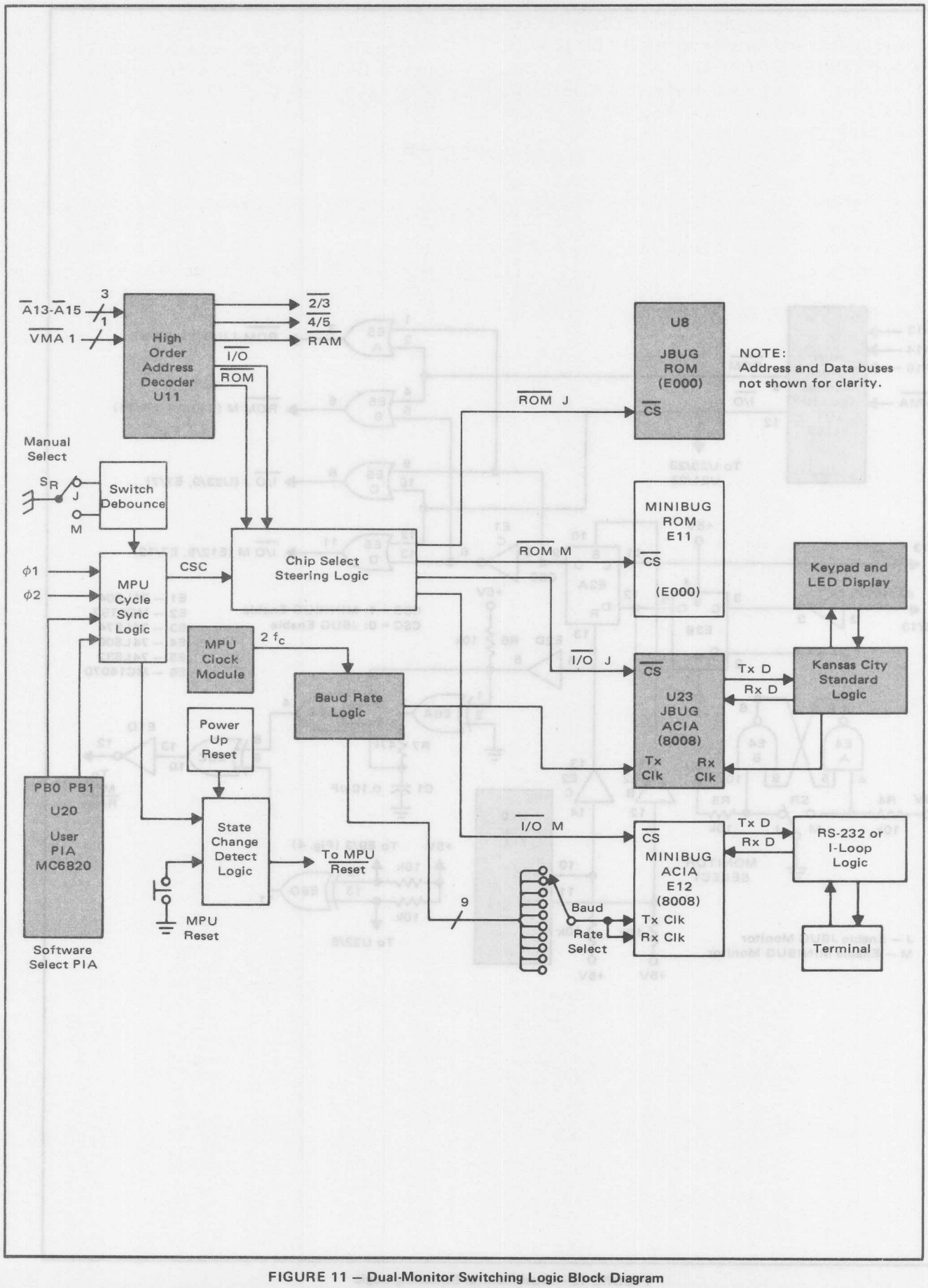


FIGURE 11 – Dual-Monitor Switching Logic Block Diagram

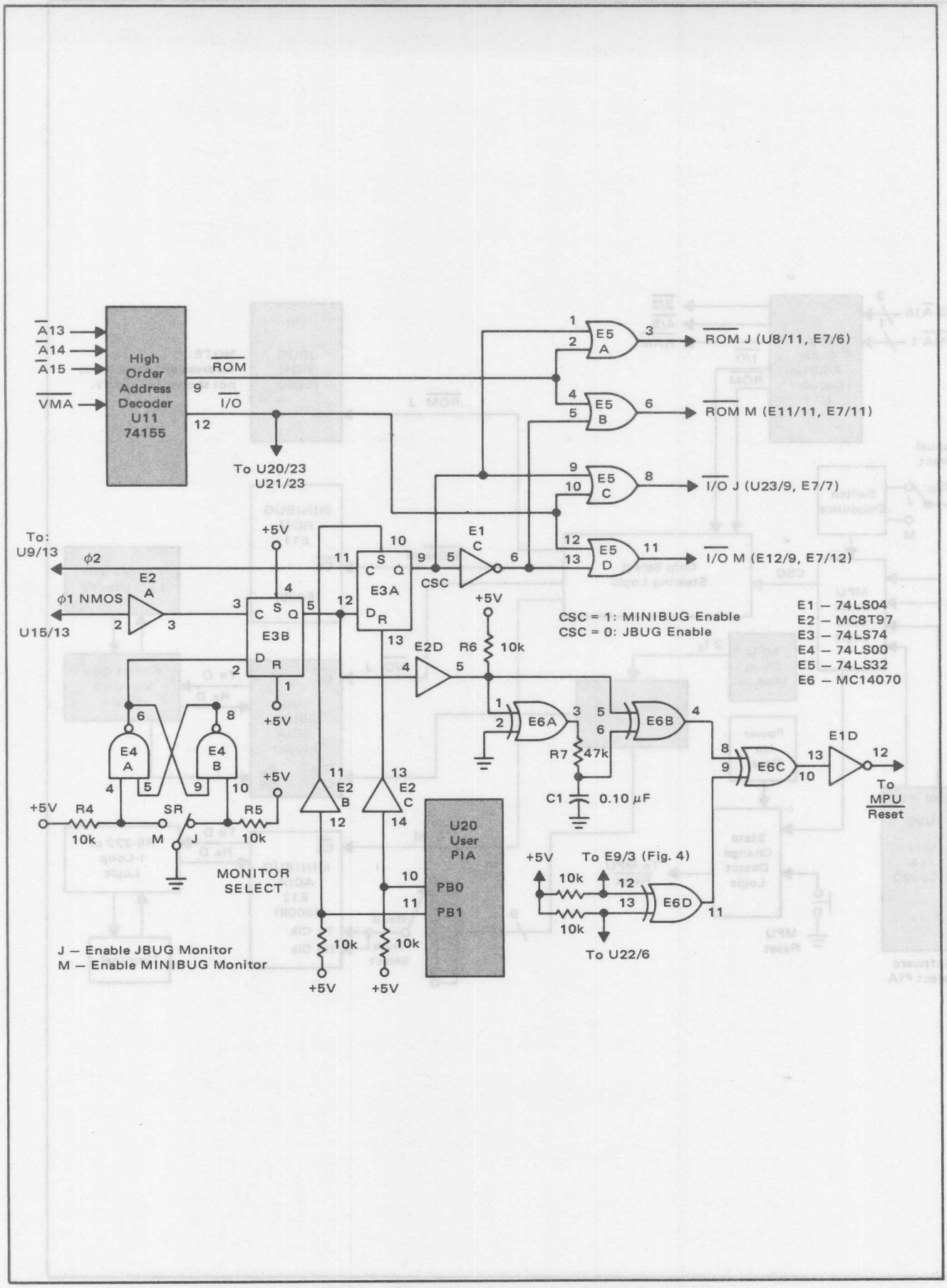


FIGURE 12 – Dual-Monitor Switching Logic

Figure 13 shows the chip-select timing for a manual command conversion from JBUG to MINIBUG via toggle switch S_R. Exclusive-OR gates E6 in Figure 12 form the state-change detection circuit which generates a 4 ms reset pulse for automatic MPU initialization whenever the monitor select switch is thrown in either direction. Note that provision for direct push-button reset of the MPU is also retained via E6D to pin 6 of U22.

Figure 14 shows address, data and control signal interconnection to the MINIBUG ROM and its ACIA. Note that even though these peripherals reside at the same bus address as the JBUG pair, the two pairs are never simultaneously selected due to the complementary control nature of the chip select steering logic.

Figures 15 and 16 show circuitry necessary for interfacing with data terminals using either RS-232 or current-loop I/O configuration. Data terminal baud-rate clocks may be taken from the existing MC14040 binary counter (U17) outputs as shown in Figure 17. An MC1455 connected as an astable multivibrator (E13) is utilized to generate a baud-rate clock consistent with current-loop TTYs.

Software Control Considerations

Software access to addresses in either Monitor ROM or ACIA is gained through a subroutine which controls the output states of PB0 and PB1 of the user PIA. The four possible states of PB0 – PB1 produce the following control functions with respect to latch E3A, Figure 12:

PB1	PB0	Monitor Control Function
0	0	Illegal state
0	1	Enable MINIBUG ROM/ACIA user addressing
1	0	Enable JBUG ROM/ACIA user addressing
1	1	Addressing controlled by Monitor Select Switch, S_R

The 1-1 state is automatically entered upon system power-up or manual reset, since following the power-up reset pulse the PIA Data-Direction-Registers are programmed as inputs (all registers cleared). PB0 – PB1 appear as high-impedance inputs and both terms are held at logic 1 by the 10 kΩ pullup resistors.

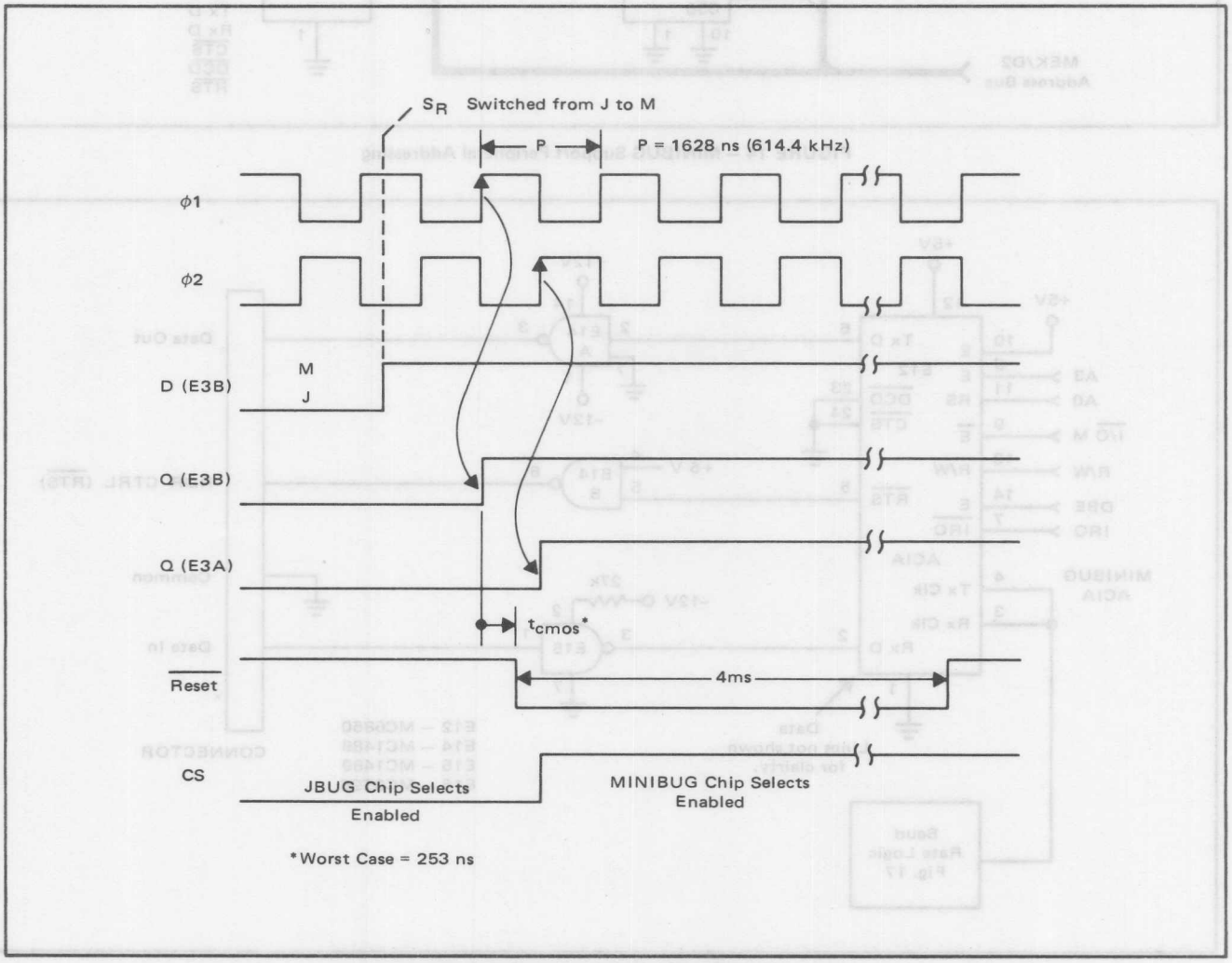


FIGURE 13 – Monitor Chip-Select Timing – Manual Select Select MINIBUG, Deselect JBUG

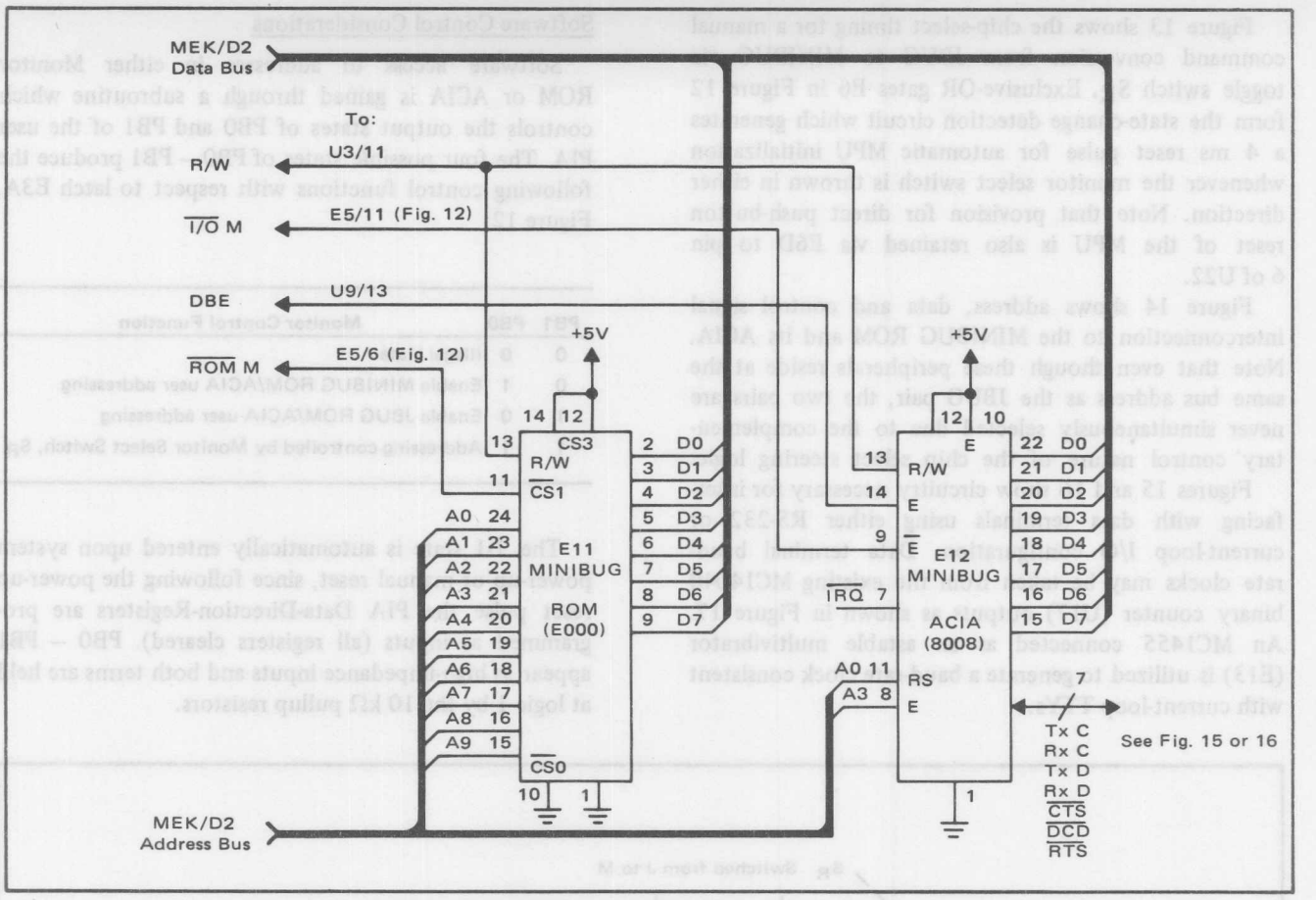


FIGURE 14 – MINIBUG Support Peripheral Addressing

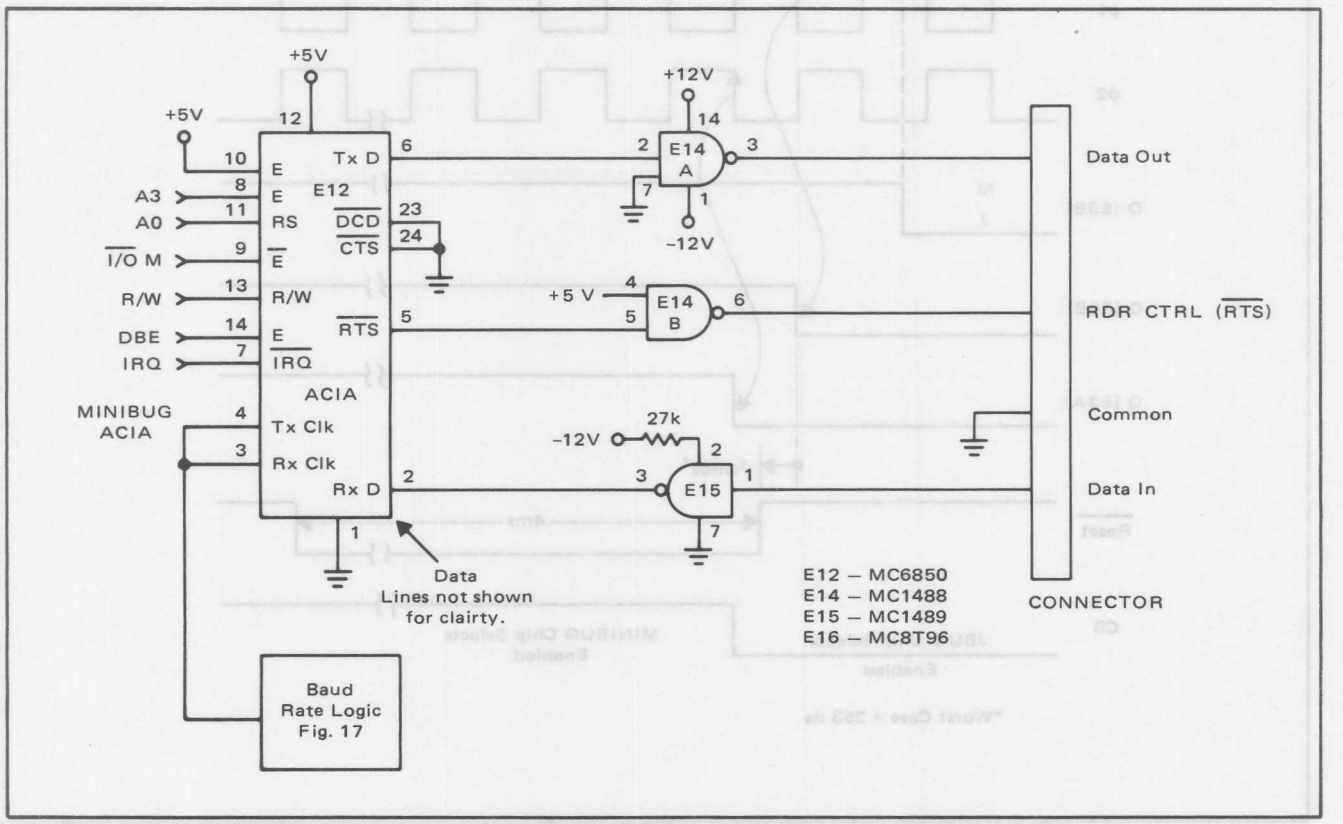


FIGURE 15 – Terminal Interface Logic
RS-232 Channel

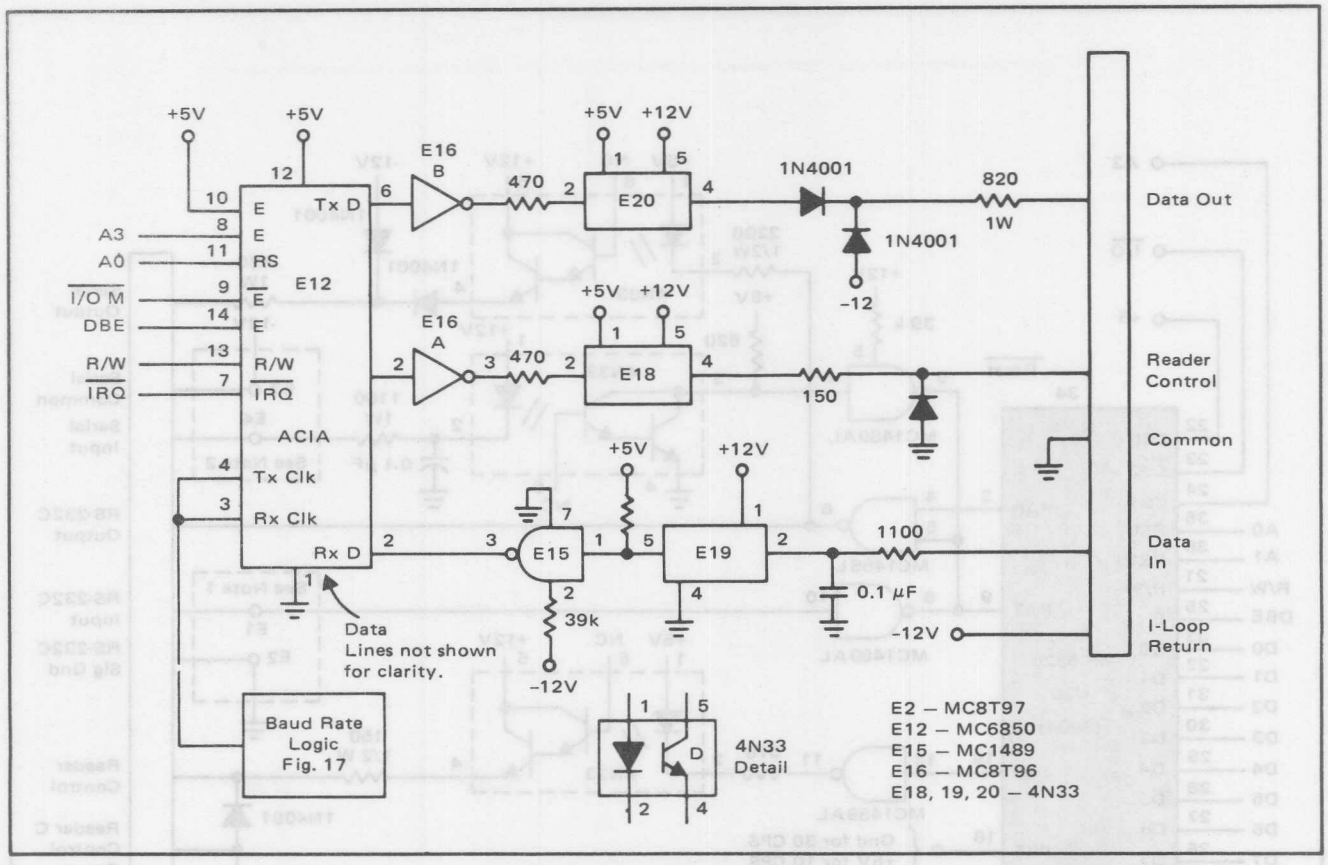


FIGURE 16 – Terminal Interface Logic
Current Loop Channel

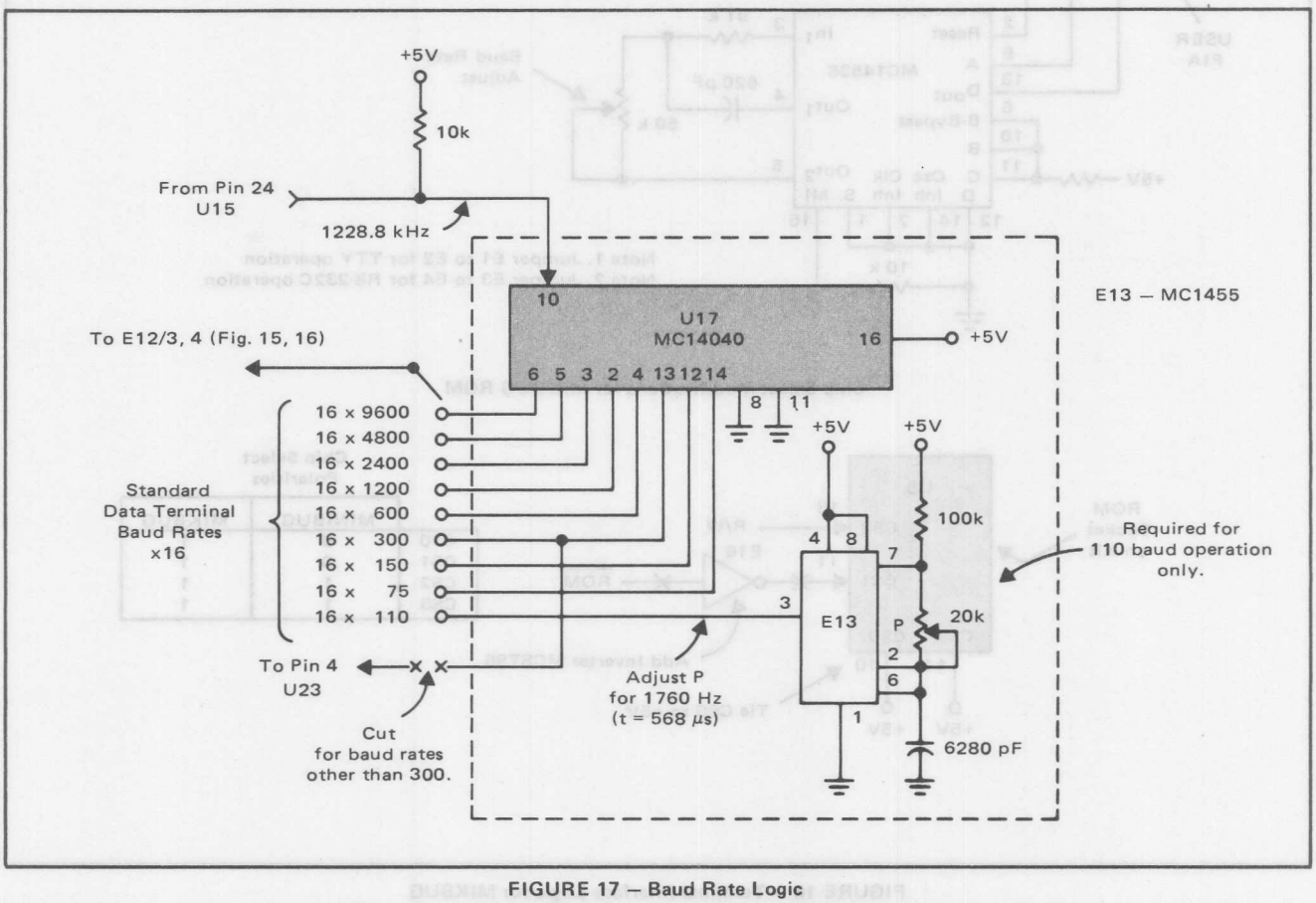
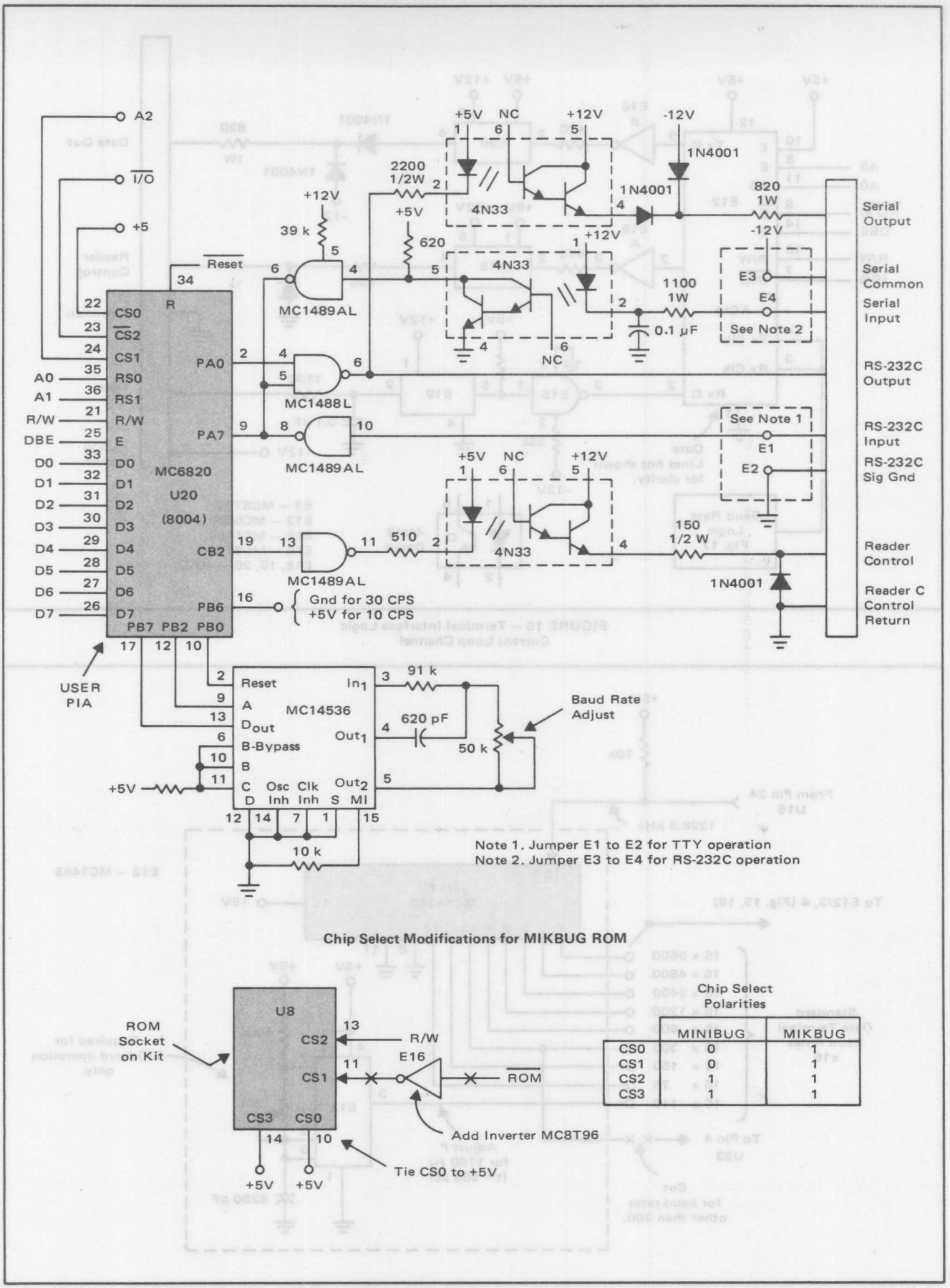


FIGURE 17 – Baud Rate Logic



Chip Select Modifications for MIKBUG ROM

Chip Select Polarities

	MINIBUG	MIKBUG
CS0	0	1
CS1	0	1
CS2	1	1
CS3	1	1

FIGURE 18 – Terminal Interface Logic for MIKBUG

Data Terminal-Only Configurations

A configuration which employs data-terminal communication interface only may be easily implemented by inserting MINIBUG or MIKBUG Monitor ROMs into the JBUG ROM socket (U8). Foil path modifications and additional logic necessary to support these ROMs are as follows:

Modifications for MINIBUG

1. Cut foil path at U17, pin 13.
2. Connect pins 3 and 4 of U23 (ACIA for Audio Cassette).
3. Add terminal I/O interface logic as shown by Figures 15 or 16 and Figure 17. Connect U17 output to pin 3 or 4 of U23 as shown. U17/3 need not be cut (as shown in Figure 17) if 300 baud operation is desired.

Modifications for MIKBUG

1. Add terminal I/O interface logic to the user PIA (U20) as described by the schematic of Figure 18.
2. Cut foil paths at U8/10 and U8/11 and connect per Figure 18.

The I/O logic and discrete components described in these figures may be mounted in the wire-wrap area provided on the microcomputer module board.

SYSTEM APPLICATION CONSIDERATIONS

A subroutine which controls the monitor-selection latch (E3, Figure 12) through the PIA is presented in Figure 19. User program access to subroutines in ROM or addresses in ACIA is accomplished by first calling the monitor access subroutine (MONACC) shown in Figure 19 and then executing a memory reference instruction to the ROM or ACIA address desired. As an example, the subroutine calling sequence:

```
        .
        .
        .
LDAA # $ 41 Form ASCII "A"
LDAB # $ 01 Get subroutine constant
JSR MONACC Enable MINIBUG ROM/
                ACIA addressing
JSR $ E108 Output ASCII char to
                terminal
        .
        .
        .
```

causes the character "A" to be printed on a terminal as a result of MINIBUG monitor access from the subroutine MONACC. In this example, the hex address E108 is the start vector of the MINIBUG II subroutine OUTCH which outputs one ASCII character to a terminal. The following is a list of useful data-moving subroutines contained in MINIBUG II and III along with their starting addresses, entry and exit conditions:

* $ is Motorola Resident Assembler syntax for a hexadecimal number.

MINIBUG ROUTINES

( ) – Addresses in MINIBUG II
⟨ ⟩ – Addresses in MINIBUG III

BADDR ($E0D9) ⟨$E0F8⟩* – Build a 16-bit hexadecimal address from four digits entered from the keyboard.
Entry requirements: none
Exit: X-register contains the 16-bit address. The A & B registers are destroyed.

BYTE ($E0E7) ⟨$E106⟩ – Input two hex characters from the keyboard and form a 1-byte number.
Entry requirements: none
Exit: A-register contains the 8-bit number. B-register is destroyed.

OUTHL ($E0FA) ⟨$E118⟩ – Output left digit of hex number to console.
Entry requirements: A-register contains hex number.
Exit: A-register is destroyed.

OUTHR ($E0FE) ⟨E11C⟩ – Output right digit of hex number to console.
Entry requirements: A-register contains hex number.
Exit: A-register is destroyed.

OUTCH ($E108) ⟨$E126⟩ – Output one ASCII character to terminal.
Entry requirements: A-register contains ASCII character to output.
Exit: No change

INCHP ($E115) ⟨$133⟩ – Input one character, with parity, from terminal to A-register.
Entry requirements: None
Exit: A-register contains character input.

INCH ($E11F) ⟨$E133⟩ – Input one character from terminal to A-register and set parity bit = 0. If character is a delete ($7F) it is ignored. Location $A00C should be equal to zero if the character should be echoed (MINIBUG II only).
Entry requirements: none
Exit: A-register contains character without parity.

PDATA1 ($E130) ⟨$E14B⟩ – Print at terminal the ASCII data string pointed to by X-register. Data string must contain an ASCII EOT ($04) as a terminator.
Entry requirements: X-register contains the address of the 1st byte of the data string. The data string is terminated with a $04 character.
Exit: A-register is destroyed. X-register contains address of $04 character.

OUT2H ($E173) ⟨$E18D⟩ – Output two hex characters, pointed to by X-register to the terminal.
Entry requirements: X-register contains the address of the characters to be output.
Exit: A-register is destroyed. X-register is incremented.

OUT2HA ($E175) ⟨$E10F⟩ – Output two hex character in A-register to the terminal.

Entry requirements: A-register contains the characters to output.

Exit: A-register is destroyed. X-register is incremented.

OUT4HS ($E17C) ⟨$E196⟩ – Output four hex characters (2 bytes) plus a space to the terminal.

Entry requirements: X-register contains address of first byte.

Exit: A-register is destroyed. X-register contains address of second byte.

OUT2HS ($E17E) ⟨$E198⟩ – Output two hex characters (1 byte) and a space to the terminal.

Entry requirements: X-register contains address of byte to output.

Exit: A-register is destroyed. X-register is incremented.

OUTS ($E180) ⟨$E19A⟩ – Output a space.

Entry requirements: none

Exit: A-register destroyed.

The ability to gain access to two co-residing monitor ROMs via manual or software commands combined with keyboard, audio cassette, or data terminal I/O capability provides opportunity for moving program data between various storage media. It is possible, for instance, to create and assemble a program under the control of MINIBUG II or III using an RS-232-compatible digital cassette terminal. The resulting object code is loaded to MEK/D2 RAM using the MINIBUG "L" command. The Monitor Control Switch may now be used to initialize the JBUG Monitor in order to move the object code in RAM to an audio cassette tape with a JBUG "P" command.

```
00001                    NAM     MONACC
00002                    OPT     O,S
00003                  * SUBROUTINE TO CONTROL ROM ACCESS PIA
00004                  * FROM USER PROGRAM. ROM ACCESS CONSTANT
00005                  * IS REQUIRED IN ACC-B ON SUBROUTINE
00006                  * ENTRY AS FOLLOWS :
00007                  *       $01 = ENABLE MINIBUG ROM/ACIA ACCESS
00008                  *       $02 = ENABLE JBUG ROM/ACIA ACCESS
00009                  *       $03 = ENABLE TOGGLE SWITCH ACCESS
00010      8006      IODDB   EQU     $8006
00011      8007      CRB     EQU     $8007
00012 0000 36        MONACC  PSH A
00013 0001 4F                CLR A
00014 0002 B7 8007           STA A   CRB       ENABLE DDB ACCESS
00015 0005 43                COM A
00016 0006 B7 8006           STA A   IODDB     MAKE ALL PB'S OUTPUTS
00017 0009 86 04             LDA A   #$04
00018 000B B7 8007           STA A   CRB       ENABLE IO ACCESS
00019 000E 86 03             LDA A   #$03
00020 0010 B7 8006           STA A   IODDB     PRE-SET E3 S,R INPUTS
00021 0013 F7 8006           STA B   IODDB     WRITE ACCESS WORD TO E3
00022 0016 32                PUL A
00023 0017 39                RTS
00024                  *     DDB = PIA DATA DIRECTION REGISTER-B SIDE
00025                  *     CRB = PIA CTRL REGISTER-B SIDE
00026                  *     IODDB = PIA I/O,DIRECTION REG-B SIDE
00027                        END
IODDB   8006
CRB     8007
MONACC  0000

TOTAL ERRORS 00000
```

FIGURE 19 – ROM Access Subroutine

* $ is Motorola Resident Assembler syntax for a hexadecimal number.

Figure 20 presents a tabular comparison of command sets for JBUG, MINIBUG and MIKBUG monitors. Any two pairs of these monitors may be used to configure the MEK/D2 computer to maximum advantage to suit the application through use of the dual monitor access logic described in Figure 12. A comparison of the commands of Figure 20 reveals that an excellent combination might be a MINIBUG II/MINIBUG III configuration. This would provide capability for memory test, punching and loading of binary tapes as well as access to the powerful software edit functions of Trace and Breakpoint insertion.

Monitor Function	JBUG	MINIBUG II	MINIBUG III	MIKBUG	Notes
Display Internal Registers	R	R	R	R	1
Load RAM from Tape	L	L	L	L	
Dump RAM to Tape (Punch)	P	P	P	P	2
Memory Examine/Change	M	M	M	M	3
Go to Entered Address and Execute	G	G	G	G	4
Set Terminal Baud Rate	—	S	S	—	S
Test Memory	—	W	—	—	6
Punch Binary Tape from RAM	—	Y	—	—	7
Load Binary Tape to RAM	—	Z	—	—	7
Abort Program Execution (Escape)	E	—	—	—	
Trace One Instruction	N	—	N	—	
Set a Breakpoint	V	—	V	—	8
Reset a Breakpoint	V	—	U	—	
Continue Execute from Breakpoint	E, G	—	C	—	
Delete All Breakpoints	V	—	D	—	8
Print Addresses of All Breakpoints	—	—	B	—	
Trace N Instructions	—	—	T	—	

NOTES

1. Order of Display: JBUG (PC,X,A,B,CC,SP); MINIBUG II and III, MIKBUG (PC,SP,CC,B,A,X).
2. Before executing, load beginning and ending address of range in locations A002 to A005.
3. For JBUG: Enter address, type M for contents. For MINIBUG: Enter M followed by address. Contents are displayed after typing last address character. For MIKBUG: Enter M, space, address. Address and data are printed.
4. For JBUG: Enter starting address, type G. For MINIBUG: Type G, followed by address. Execution begins after type of last character. For MIKBUG: Load start address in A048/A049, type G.
5. For 110 Baud: Type S1. For 300 Baud: Type S3.
6. Performs six memory tests: walking address, write/read all ones, all zeros, AA, 55 and "Walking Bit."
7. Data is in binary (not ASCII) format. Requires a terminal with DC2, DC4 character recognition.
8. For JBUG: Type address where breakpoint is desired, followed by V. A total of five may be entered. Removal of all breakpoints executed by typing V not preceded by address. For MINIBUG III: Same as JBUG except eight breakpoints may be entered.
9. IRQ vector must be stored at A000/A001, NMI must be stored at A006/A007 for all monitors.

FIGURE 20 – Comparison of Monitor Commands

Figure 21 presents a brief test program for evaluating user-program access to monitor subroutines through the monitor switching logic. The program should be executed from JBUG, i.e. with the monitor select switch in the J-poisition. Upon execution, MINIBUG addressing is enabled and a string of control characters are transmitted to the terminal. Following this, any character typed at the terminal is echoed to the terminal. When the character "ESC" is typed, the program jumps from the echo loop, JBUG addressing is software enabled and program control passes from the user program to the JBUG monitor. This action may be checked by viewing the dash "prompt" in the keyboard LED display immediately after typing the "ESC" character on the terminal keyboard.

The W command of MINIBUG II may be used to test all memory in the expanded system. Figure 20 describes the use of this command.

```
00001                            NAM     TEST1
00002                            OPT     O,S
00003                    *
00004                    * TEST PROGRAM TO EVALUATE SOFTWARE ACCESS TO
00005                    * JBUG AND MINIBUG SUBROUTINES THROUGH MONITOR
00006                    * SWITCHING LOGIC. TERMINAL IS 300 BAUD, RS-232
00007                    * CONFIGURED. PROGRAM DOES MINIBUG ADDRESS
00008                    * ENABLE, EXECUTES CR + 4LF'S AT TERMINAL,
00009                    * THEN JUMPS TO A CHARACTER ECHO MODE. EACH
00010                    * CHARACTER TYPED AT THE TERMINAL IS ECHOED
00011                    * AND PRINTED AT THE TERMINAL. WHEN AN "ESC"
00012                    * IS TYPED, THE PROGRAM JUMPS OUT OF THE ECHO
00013                    * LOOP AND ENABLES JBUG MONITOR ADDRESSING.
00014                    * CONTROL IS PASSED TO THE JBUG MONITOR. THIS
00015                    * ACTION MAY BE VIEWED BY OBSERVING THE JBUG
00016                    * "PROMPT" (A DASH) ON THE MEK/D2 KEYBOARD
00017                    * DISPLAY IMMEDIATELY FOLLOWING TYPE OF THE
00018                    * "ESC" ON THE TERMINAL. THE PROGRAM IS INITIATED
00019                    * FROM JBUG WITH THE "G" COMMAND.
00020                    *
00021          8004      IODDA   EQU     $8004
00022          8006      IODDB   EQU     $8006
00023          8005      CRA     EQU     $8005
00024          8007      CRB     EQU     $8007
00025          8008      ACIAC   EQU     $8008
00026          8009      ACIAD   EQU     $8009
00027 4000               ORG     $4000
00028 4000 CE 41FF       LDX     #$41FF
00029 4003 01            NOP
00030 4004 0F            SEI
00031 4005 C6 01         LDA B   #$01    GET MINIBUG ENABLE CONSTANT
00032 4007 BD 403A       JSR     MONACC  ENABLE MINIBUG ADDRESSING
00033 400A 86 03         LDA A   #$03
00034 400C B7 8008       STA A   ACIAC   CLEAR ACIA
00035 400F 86 09         LDA A   #$09    7BITS,EVN PRTY,1 STOP,/16
00036 4011 B7 8008       STA A   ACIAC   CONFIGURE ACIA
00037 4014 86 0D         LDA A   #$0D
00038 4016 5F            CLR B
00039 4017 BD E108       JSR     $E108   XMIT CR
00040 401A 86 0A         LDA A   #$0A
00041 401C BD E108 K1    JSR     $E108   XMIT LF
00042 401F 5C            INC B
00043 4020 C1 04         CMP B   #$04    4 LF'S ?
```

FIGURE 21 — Test Program

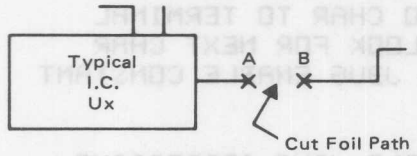
```
00044 4022 26 F8          BNE    K1          END LOOP
00045 4024 BD E115 K2     JSR    $E115       BRING IN TERMINAL CHAR
00046                     * ACCUMULATOR A CONTAINS THE ASCII CHARACTER
00047 4027 81 1B          CMP A  #$1B        IS IT AN "ESC" ?
00048 4029 27 05          BEQ    K3          IF YES LEAVE ECHO LOOP
00049 402B BD E108        JSR    $E108       ECHO CHAR TO TERMINAL
00050 402E 20 F4          BRA    K2          GO LOOK FOR NEXT CHAR
00051 4030 C6 02  K3      LDA B  #$02        GET JBUG ENABLE CONSTANT
00052 4032 01             NOP
00053 4033 01             NOP
00054 4034 BD 403A        JSR    MONACC      ENABLE JBUG ADDRESSING
00055 4037 7E E08D        JMP    $E08D       JUMP TO JBUG INIT VECTOR
00056                     *
00057                     *** SUBROUTINE ***
00058                     * ACCB CONTAINS ENABLING CONSTANT ON ENTRY
00059                     * $01=MINIBUG,$02=JBUG,$03=MANUAL SWITCH
00060                     *
00061 403A 36             MONACC PSH A
00062 403B 4F                    CLR A
00063 403C B7 8007               STA A  CRB         ENABLE DDB ACCESS
00064 403F 43                    COM A
00065 4040 B7 8006               STA A  IODDB       MAKE ALL PB'S OUTPUTS
00066 4043 86 04                 LDA A  #$04
00067 4045 B7 8007               STA A  CRB         ENABLE IO ACCESS
00068 4048 86 03                 LDA A  #$03
00069 404A B7 8006               STA A  IODDB       PRE-SET E3 S,R INPUTS
00070 404D F7 8006               STA B  IODDB       WRITE ACCESS WRD TO E3
00071 4050 32                    PUL A
00072 4051 39                    RTS
00073                            END
IODDA   8004
IODDB   8006
CRA     8005
CRB     8007
ACIAC   8008
ACIAD   8009
K1      401C
K2      4024
K3      4030
MONACC  403A


TOTAL ERRORS 00000
```

FIGURE 21 (Continued) – Test Program

SUMMARY OF MODIFICATIONS

A summary of foil-path modifications which account for both memory expansion and inclusion of multiple-monitor logic is tabulated in Figure 22.

Figure 23 presents a tabular summary of additional power supply capability required to support the expansion logic and memory. Data from this table may be used to estimate requirements for a specific system configuration.



Cut Foil Path At	Connect Cut Path	Term	See Figure
U6/36	A Side to E4/3, E7/1	$\overline{DBES}$	3
*U11/4	A Side to E23/1	$\overline{RAM}$	9
	B Side to U11/6	4/5 – $\overline{VRA}$ 2	9
U11/9	A Side to E5/2,4	$\overline{ROM}$	12
	B Side to E5/3	$\overline{ROM}$ J	12
U11/12A	A Side to E5/10,12	$\overline{I/O}$	12
	B Side to E5/8	$\overline{I/O}$ J	12
U23/3,4	B Side to E1/12	$\overline{Reset}$	12
U7/8	B Side to E7/9	$\overline{DBRE}$	10
U20/23	A Side to U11/12	$\overline{I/O}$ – User PIA	–
U21/23	A Side to U11/12	$\overline{I/O}$ – Keyboard PIA	–

*For system with 16.5K memory only. For 8.5K system, connect A side to E1/11 and B side to U11/5.

FIGURE 22 – MEK/D2 Foil Path Modification

EXPANSION LOGIC

Expansion Device	Type	Worst Case Supply Currents (mA)	Reference Figure
E1	74LS04	6.6	4, 8, 12
E2	MC8T97	98.0	12
E3	74LS74	8.0	12
E4	74LS00	4.4	3, 4, 12
E5	74LS32	9.8	12
E6	MC14507	0.008	12
E7	74LS133	1.1	10
E9	MC1455	6.0	4
E11	MC6830	130.0	14
E12	MC6850	105.0	14, 15, 16
E13	MC1455	6.0	17
E14	MC1488	25 (+12 V), 15 (-12 V)	15
E15	MC1489	26.0	15, 16
E16	MC8T96	89.0	16
E17	74LS74	8.0	4
E18	4N33	10.7 (+5 V), 80 (+12 V)	16
E19	4N33	20.0 (+12 V)	16
E20	4N33	10.7 (+5 V), 20 (+12 V)	16
E21	MC14503	0.004	3
E22	74LS08	8.8	4, 6
E23	74LS86	10.0	9

EXPANSION MEMORY ARRAYS

			Worst-Case Supply Currents (mA)			
Architecture	Type	Motorola Part	+5 V	+12 V	-5 V	-12 V
2k x 8	Static	MEX6812-1	1000	–	–	–
8k x 8	Dynamic	MEX6815-1	860	300	–	–
16k x 8	Dynamic	MMS68100*	1200	333	4	–
16k x 8	Dynamic	MMS68103	1200	333	–	1.7

*Not board-geometry compatible with EXORciser.

FIGURE 23 – DC Power Supply Requirements for MEK/D2 Expansion

CONCLUSION

The technqiues discussed in this note add the following capability to the basic MEK/D2 kit microcomputer

* Power-up auto-reset
* Switch-selectable monitor operation
* RS-232 or current-loop data terminal operation at all standard baud-rates
* RAM expansion to 16.5K bytes
* ROM-resident subroutine acquisition by user program
* Operation with JBUG, MINIBUG II and III or MINIBUG ROM monitors

MOTOROLA Semiconductor Products Inc.

Printed in Switzerland