The Engineering Staff Of
TEXAS INSTRUMENTS INCORPORATED
Semiconductor Group

TM 990/201 AND TM 990/206 EXPANSION MEMORY BOARDS

JANUARY 1979

TEXAS INSTRUMENTS
INCORPORATED

PREFACE

This document describes two Texas Instruments memory expansion boards: the TM 990/201 EPROM RAM expansion board and the TM 990/206 RAM-only expansion board. Essentially, the TM 990/206 board is the TM 990/201 board with only the latter board's RAM circuitry and without its EPROM circuitry. The RAM circuitry is the same for both boards. The TM 990/201 is presented in detail in Sections 1 to 4, and the differences between the TM 990/201 and TM 990/206 are described in Section 5.

TABLE OF CONTENTS

1. INTRODUCTION 1-1
   1.1 General 1-4
   1.2 Manual Organization 1-4
   1.3 Specifications 1-4
   1.4 Applicable Documents 1-4

2. INSTALLATION AND OPERATION
   2.1 General 2-1
   2.2 Required Equipment 2-1
   2.3 Unpacking 2-1
   2.4 Power and Terminal Hookup 2-1
   2.5 Memory Mapping 2-2
   2.6 Memory Access Speed 2-2
   2.7 Operation 2-2
   2.8 Example 2-2
   2.8.1 Configure Memory Map 2-4
   2.8.2 Select Wait State 2-4

3. MEMORY PLACEMENT AND SELECTION
   3.1 General 3-1
   3.2 Memory Placement 3-1
   3.2.1 Memory Configuration Switch Array (S1) 3-1
   3.2.2 Memory Placement by Blocks 3-5

4. THEORY OF OPERATION
   4.1 General 4-1
   4.2 Static RAM Section 4-1
   4.3 EPROM Section 4-2
   4.4 Address Map Options 4-2
   4.5 EPROM Decode Logic 4-5
   4.6 RAM Decode Logic 4-5
   4.7 Addressing Summary 4-5
   4.8 Memory Speed and Timing 4-5
   4.8.1 Memory Speed 4-5
   4.8.2 Memory Timing 4-7
   4.8.3 Ready Logic 4-9
   4.9 RAM Precedence Logic 4-9
   4.10 Interface Description 4-9

5. TM 990/206 RAM-ONLY MEMORY EXPANSION BOARD
   5.1 General 5-1
   5.2 Specifications 5-1
   5.3 Installation and Opertion 5-3
   5.4 Memory Placement and Selection 5-3
   5.5 Operation 5-3
   5.6 Example 5-4
   5.6.1 Configure Memory Map 5-4
   5.6.2 Select Wait State 5-4

APPENDICES

A. Programming Address Decode PROM's for Alternate Address Maps
B. TM 990/201 Schematics
C. PROM Program Sheets
D. TM 990/422 Demonstration Software

LIST OF ILLUSTRATIONS

Figure 1-1. TM 990/201 Memory Expansion Board 1-2
Figure 2-3. Board Dimensions 1-3

Figure 2-1. TM 990/201 Memory Map Example 2-3

Figure 3-1. Memory Configuration Switch 3-2
Figure 3-2. TM 990/201 EPROM Memory Configuration 3-3
Figure 3-3. TM 990/201 RAM Memory Configuration 3-4
Figure 3-4. Memory Block Locations 3-7

Figure 4-1. TM 990/201 Block Diagram 4-1
Figure 4-2. TM 990/201 Address Decode Logic Block Diagram 4-3
Figure 4-3. EPROM Decode Logic 4-4
Figure 4-4. RAM Decode Logic 4-6
Figure 4-5. SLOW/FAST Memory Jumper Placement 4-7
Figure 4-6. TM 990/201 Memory Timing 4-8
Figure 4-7. TM 990/201 RAM Ready Logic 4-10
Figure 4-8. RAM Precedence Logic 4-11
Figure 4-9. TM 990/510 OEM Chassis Backplane Schematic 4-11

Figure 5-1. TM 990/206 Memory Board 5-2
Figure 5-2. TM 990/206 Memory Map Example 5-5
Figure 5-3. RAM (ONLY) Configuration for Model 990/201-4 5-6

Figure A-1. Decode PROM Functional Diagram A-2
Figure A-2. TM 990/201 RAM Decode PROM Program A-4
Figure A-3. TM 990/201 EPROM Decode PROM Program A-5

LIST OF TABLES

Table 1-1. TM 990/201 Product Matrix 1-1
Table 1-2. TM 990/201 Power Consumption vs Sizes 1-4

Table 4-1. FAST/SLOW Jumper (J1/J2) Positions versus Memory Access Time 4-7
Table 4-2. Backplane/P1 Pin Assignments Used by TM 990/201 Board 4-12

Table 5-1. TM 990/206 Product Matrix 5-1
Table 5-2. TM 990/206 Power Consumption vs Sizes 5-3

Table A-1. RAM PROM Decode Programming Example A-3

SECTION 1

INTRODUCTION

1.1 GENERAL

Sections 1 to 4 present detailed information on the TM 990/201 EPROM/RAM memory expansion board. Section 5 covers the TM 990/206 RAM-only memory expansion board and how it differs from the TM 990/201 board. Information applicable to the RAM configurations in Sections 1 to 4 is applicable to the TM 990/206.

The Texas Instruments TM 990/201 is an expansion memory board (shown in Figure 1-1) for use with the TM 990/100M microcomputer. Its features include:

- Up to 8K words of TMS 4045 static RAM (1024 X 4 bits each)
- Up to 16K words of TMS 2716 EPROM (2048 X 8 bits each)
- TTL compatible interface
- 4 MHz operating capability

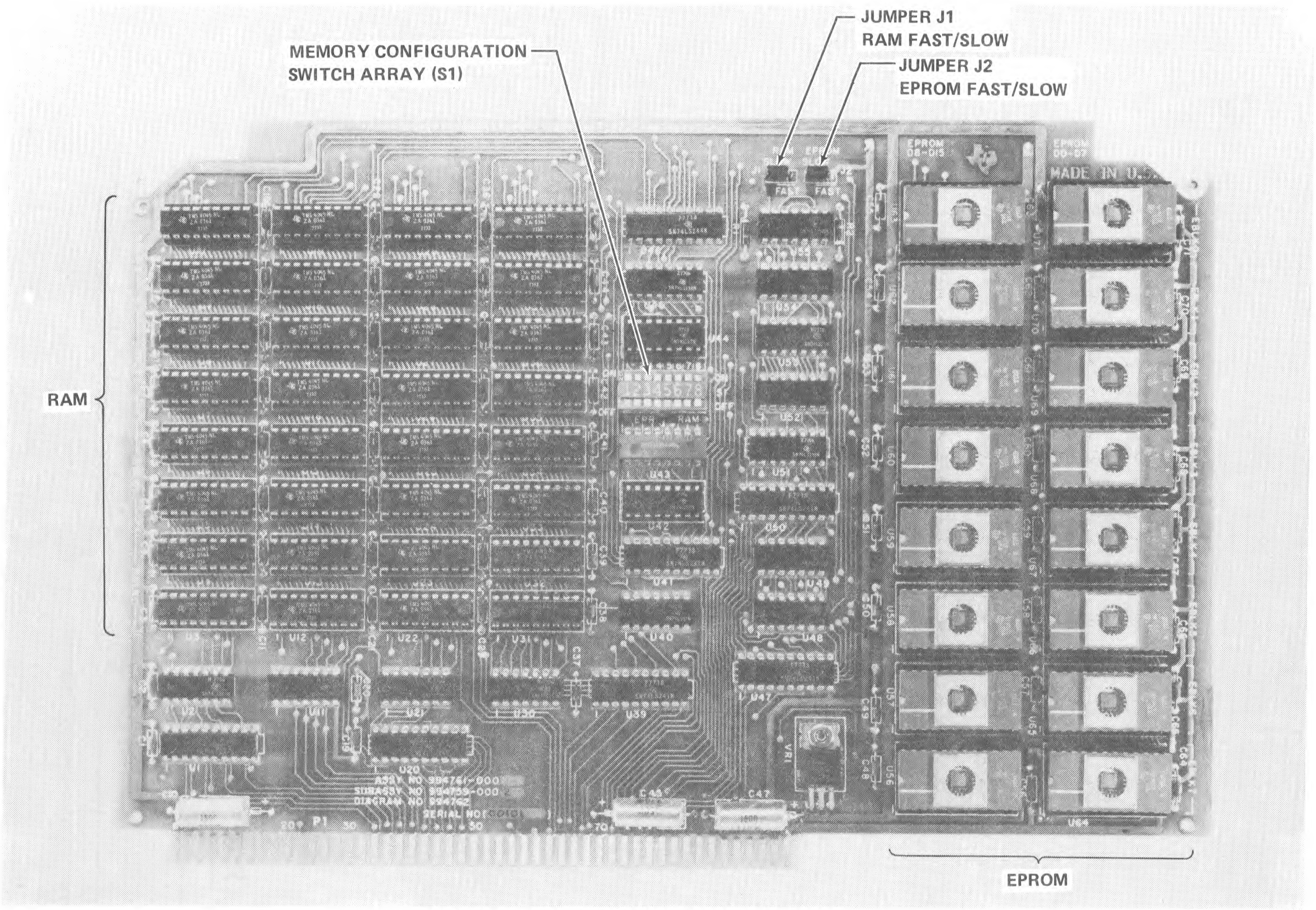
The TM 990/201 is available in three versions as shown in Table 1-1. Access to the board is through the edge connector which mates to the backplane of the TM 990/510 OEM chassis. The TM 990/201 board is not compatible with the TM 990/180M board which operates with an 8-bit data bus.

On Model TM 990/201-41, sockets are provided for 4K words of static RAM and 8K words of EPROM; however, only 2K words of RAM and 4K words of EPROM are populated. The TM 990/201-42 and -43 boards are totally socketed for up to 8K words of RAM and 16K words of EPROM and are populated in accordance with Table 1-1. Information in parentheses in Table 1-1 refers to the name of the memory block populated at the factory. Figure 1-2 shows memory board dimensions (for the TM 990/201 and /206). The TM 990/206 product matrix is shown in Section 5-1, Table 5-1.

TABLE 1-1. TM 990/201 PRODUCT MATRIX

MODEL	SOCKETS PROVIDED		SOCKETS POPULATED		RAM ACCESS TIME
	RAM	EPROM	RAM	EPROM	
TM 990/201-41	4K X 16 (RBLKO-RBLK1)	8K X 16 (EBLK4-EBLK7)	2K X 16 (RBLKO)	4K X 16 (EBLK6, EBLK7)	450 ns
TM 990/201-42	8K X 16 (RBLKO-RBLK3)	16K X 16 (EBLKO-EBLK7)	4K X 16 (RBLKO-RBLK1)	8K X 16 (EBLK4-EBLK7)	450 ns
TM 990/201-43	8K X 16 (RBLKO-RBLK3)	16K X 16 (EBLKO-EBLK7)	8K X 16 (RBLKO-RBLK3)	16K X 16 (EBLKO-EBLK7)	450 ns

NOTE: Block nomenclature explained in Section 3.



MEMORY CONFIGURATION SWITCH ARRAY (S1)
JUMPER J1 RAM FAST/SLOW
JUMPER J2 EPROM FAST/SLOW
RAM
EPROM

FIGURE 1-1. TM 990/201 MEMORY EXPANSION BOARD

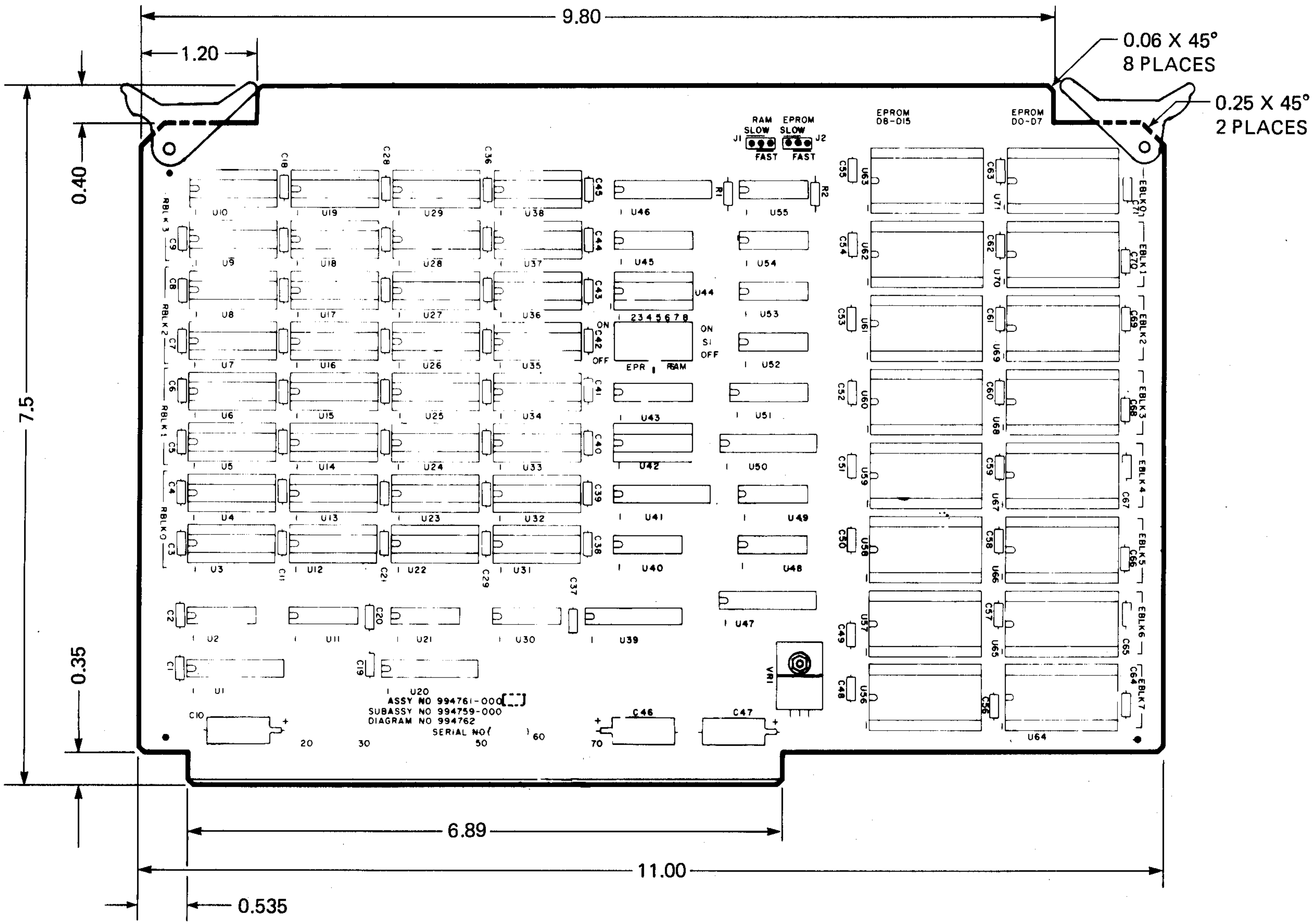


FIGURE 1-2. BOARD DIMENSIONS (IN INCHES)

1.2 MANUAL ORGANIZATION

Section 2 of this manual describes the correct procedure for installation, power up, and operation of the TM 990/201 memory expansion board. Section 3 discusses memory mapping and operation of switch array S1 as well as jumpers J1 and J2. Section 4 discusses the theory of operation, including timing considerations and addressing. Section 5 outlines the differences between the TM 990/206 and the TM 990/201.

1.3 SPECIFICATIONS

Board dimensions:	See Figure 1-2.
Temperature range:	Operating 0°C to 70°C Storage -40°C to 70°C
Clock rate:	The TM 990/201 memory expansion board is compatible with the TM 990/100M CPU at 3 MHz and can be operated at 4 MHz.
Devices utilized:	TM 4045-45 static RAM, 1K X 4 TM 2716 EPROM, 2K X 8, 450 nsec access time
Power:	See Table 1-2.

1.4 APPLICABLE DOCUMENTS

- TM 990/100 Microcomputer User's Guide
- TMS 9900 Microprocessor Data Manual

TABLE 1-2. TM 990/201 POWER CONSUMPTION VS. SIZES

MODEL	MEMORY SIZE	+5 V		+12 V		-12 V	
		MAX	TYP	MAX	TYP	MAX	TYP
TM 990/201-41	2K RAM, 4K EPROM	2.5A	1.0	.18A	.16	.5A	.05
TM 990/201-42	4K RAM, 8K EPROM	3.0A	1.4	.38A	.225	.55A	.125
TM 990/201-43	8K RAM, 16K EPROM	5.5A	2.15	.75A	.475	.7A	.225

NOTE: Voltage tolerance ±5% for all supplies.

SECTION 2

INSTALLATION AND OPERATION

2.1 GENERAL

This section explains the procedure for unpacking and setting up the TM 990/201 board for operation with a TM 990/ 10X microcomputer.*

2.2 REQUIRED EQUIPMENT

- TM 990/510 OEM chassis
- Power supply that is capable of supplying the power requirements of the memory board (Table 1-2), CPU, and other installed user equipment
- Terminal
- TM 990/ 10X microcomputer.

2.3 UNPACKING

Take the TM 990/201 board from its carton and remove the protective wrapping. Check the board for any abnormalities that could have occurred in shipping, and report any discrepancies to your supplier.

2.4 POWER AND TERMINAL HOOKUP

This procedure for hooking up a terminal and system power assumes a system of a TM 990/ 10XM microcomputer, a TM 990/510 chassis, and a suitable terminal. (See the *TM 990/100M Microcomputer User's Guide* for description of proper terminals.) The power supply must provide all the necessary power requirements for the CPU board, the memory board, and any other boards the user may be using.

The use of the TM 990/510 chassis is recommended because it offers protection from the abuse that a loose board would receive. It also provides termination resistors for the open collector signals used on the bus and allows system flexibility and hookup convenience.

There are two requirements that have to be met for proper operation of the TM 990/201:

- Proper selection of memory map
- Proper hookup

If the TM 990/510 chassis is used, the hookup is simple. Place the microcomputer in slot 1 of the chassis and place the memory board in any of the remaining slots. This positions the memory board between the CPU and the termination resistors on the backplane.

CAUTION

Always remove and insert boards with the power off. Do not insert or remove any board when the power is on as significant damage may result.

*TM 990/10X refers to the TM 990/100M, TM 990/101M, and other CPU boards in the TM 990/10X series.

2.5 MEMORY MAPPING

Care in selection of the memory map is important before powerup. Refer to Section 3 for details in memory placement and selection of address configuration using switch array S1.

2.6 MEMORY ACCESS SPEED

Jumpers J1 and J2 (Figure 1-1) must be set to FAST or SLOW to indicate respectively the access time of the RAM or EPROM memories used. The following table lists access time and J1/J2 settings.

Section 4.8.1 explains these timing constraints in detail.

MEMORY ACCESS TIME	J1 (RAM) AND J2 (EPROM) SETTING AT CLOCK RATE	
	3 MHz	4 MHz
450 ns	SLOW	SLOW
300 ns	FAST	SLOW
200 ns	FAST	FAST
150 ns	FAST	FAST

2.7 OPERATION

Essentially the user needs only to choose the correct memory configuration (Section 3), insert the board into the chassis, and apply power to set up the system for operation.

The operation of the TM 990/201 memory board should be transparent to the user in that no special signals are required other than those supplied through the backplane. If the TM 990/510 chassis is not used, refer to Section 4.10 for interface information.

2.8 EXAMPLE

This example assumes the following configurations:

(1) TM 990/10X microcomputer

- 4K X 16 EPROM in memory address (M.A.) 0000_{16} to $1FFF_{16}$.
- 512 X 16 RAM in M.A. $FC00_{16}$ to $FFFF_{16}$.

(2) TM 990/201 expansion board

- 4K X 16 EPROM
- 2K X 16 RAM

Figure 2-1 depicts the desired memory map. Note that expansion EPROM resides at address 2000_{16} to $3FFF_{16}$ while expansion RAM on the TM 990/201-41 is to reside in locations $E000_{16}$ to $EFFF_{16}$ of the TM 990/10X address map.

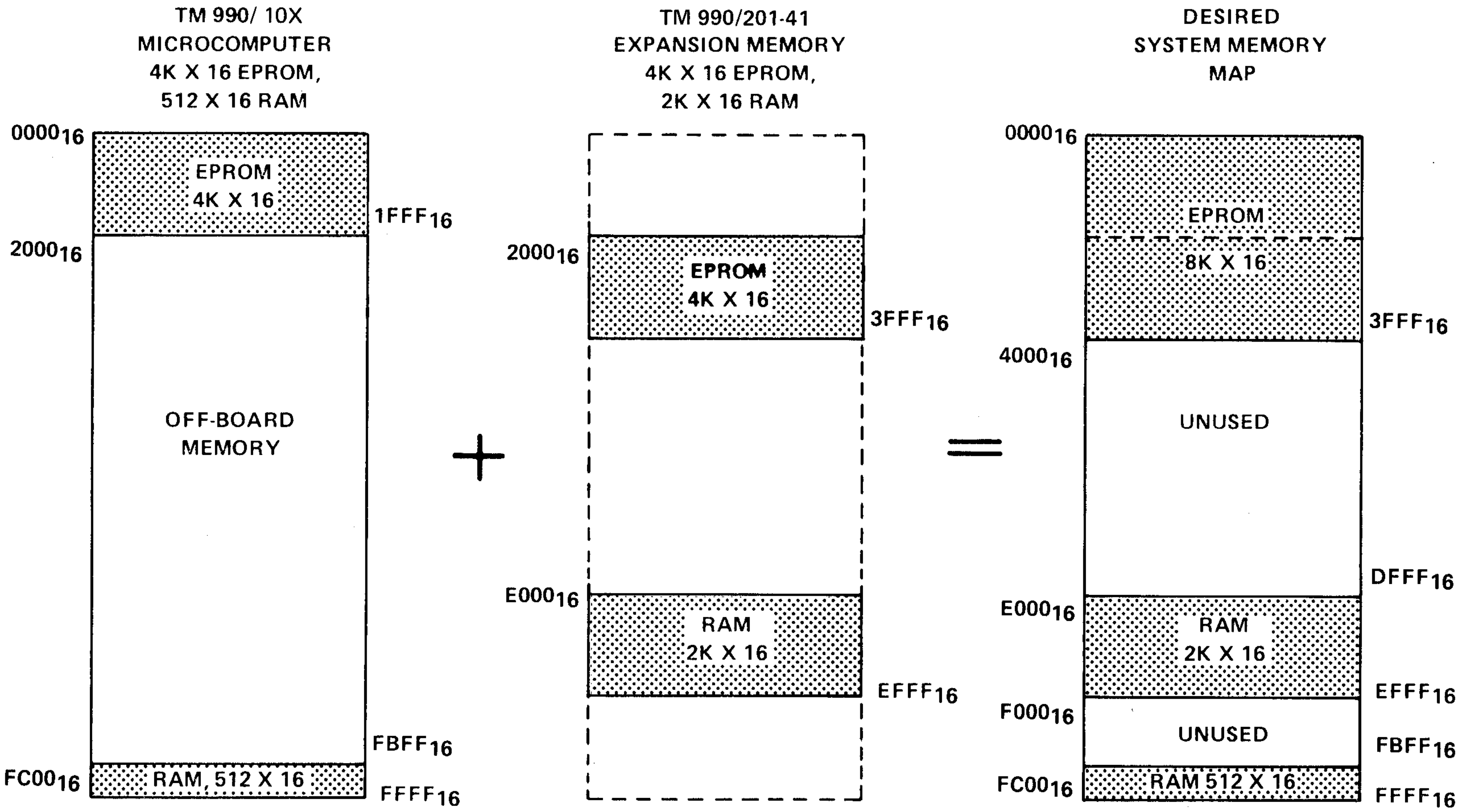


FIGURE 2-1. TM 990/201 MEMORY MAP EXAMPLE

The user must do four things to the TM 990/201-41 prior to interfacing the unit to the microcomputer:

(1) Configure the expansion RAM into the TM 990/10X memory map using switch S1 and memory placement on the board.

(2) Configure the expansion EPROM into the TM 990/10X memory map using switch S1 and memory placement on the board.

(3) Select the wait state for RAM using jumper J1.

(4) Select the wait state for EPROM using jumper J2.

2.8.1 CONFIGURE MEMORY MAP

Populate the EPROM and RAM as explained in paragraph 3.2.2 and Figure 3-4.

To map EPROM into the desired address bounds, set the memory configuration switch array (S1) to ON-ON-OFF-OFF as shown in Figure 3-2 (switches 1 to 4).

To map RAM into the desired address bounds, set switch array S1 to OFF-ON-OFF-OFF as shown in Figure 3-3 (switches 5 to 8).

Section 3 explains memory placement, mapping and selection of S1 switches.

2.8.2 SELECT WAIT STATE

The TM 990/10X operates at 3 MHz. The TM 990/201-41 is shipped with TMS 4045-45 RAM's, 450 nsec access time. Thus, place the RAM FAST/SLOW jumper (J1) in the "SLOW" position. The TM 990/201-41 is shipped with TMS 2716 EPROM's which have a 450 nsec access time. Place the EPROM FAST/SLOW jumper (J2) in the "SLOW" position.

The switch array and the FAST/SLOW jumpers are shown in Figure 1-1. Note that each switch of the array is numbered and each switch position is designated as either "ON" (a zero) or "OFF" (a one). The FAST/SLOW jumper positions are also marked "FAST" or "SLOW".

SECTION 3

MEMORY PLACEMENT AND SELECTION

3.1 GENERAL

This section describes the procedures used to map memory located on the TM 990/201 memory board into the available address space. Switch S1 and address decode PROM's U42 and U44 determine the address space occupied by both EPROM and RAM on the TM 990/201 board. To select a memory address configuration which is compatible with the system memory map, the user must first do the following:

(1) Determine the quantity of EPROM and RAM to be populated on the board.

(2) Place the EPROM and RAM devices in their correct physical locations on the board.

(3) Set the memory configuration switch array (S1) so that the memory on the TM 990 201 board is mapped into an available address space unoccupied on another board in the system. This address space must be large enough to contain the amount of memory on the TM 990/201 board, and it must not conflict with the same addresses populated on another board.

CAUTION

If there are overlaying duplicate addresses on different boards in the configured system, the resulting data bus conflict may cause damage to the data bus drivers on these boards.

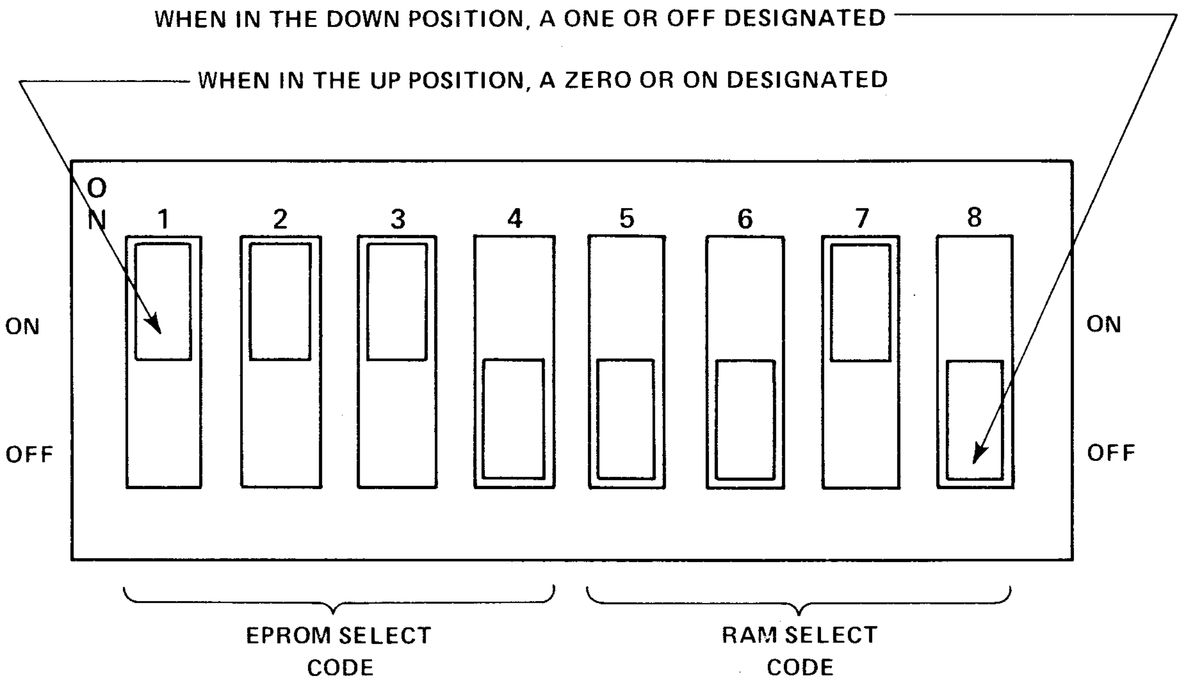
3.2 MEMORY PLACEMENT

Settings of the memory configuration switch array (S1, center of the board, see Figure 1-1) determine the address configuration that will be decoded by the address decoding circuitry — two SN74S287 PROM's programmed at U42 for decode EPROM and U44 for decode RAM. Switch array S1 selects the memory starting address and the quantity of memory starting at that address.

3.2.1 MEMORY CONFIGURATION SWITCH ARRAY (S1)

The memory configuration switch array is divided into two parts. The left four slide switches (1 to 4) designate the EPROM configuration. The right four slide switches (5 to 8) designate the RAM configuration. If the switch is set to the ON position, a binary "0" is encoded at that switch. If the switch is set in an OFF position, a binary "1" is encoded. These switch settings designate the four-bit codes that select the EPROM and RAM address configuration. Figure 3-1 shows the memory configuration switch array.

Note that when read right to left (4 to 1 and 8 to 5), the switches form the binary code 0_{16} to F_{16} as used at the top of Figures 3-2 and 3-3 to explain switch settings and memory addresses. Note again that OFF refers to a binary "1" and ON refers to a binary "0". As shown in Figures 3-2 and 3-3, switch array S1 selects the starting address and the quantity of memory decoded. For example, if switches 1 to 4 are set to all ON's, the memory decoder will select all 16 EPROM sockets (16K words) using addresses beginning at 0000_{16} (0000_{16} to $7FFF_{16}$). If switches 1 to 4 are set to OFF-ON-ON-ON respectively, the memory decoder will decode all EPROM sockets populated using a starting address of 2000_{16} (2000_{16} to $9FFF_{16}$).


WHEN IN THE DOWN POSITION, A ONE OR OFF DESIGNATED
WHEN IN THE UP POSITION, A ZERO OR ON DESIGNATED
ON
1
2
3
4
5
6
7
8
ON
OFF
ON
OFF
EPROM SELECT CODE
RAM SELECT CODE

NOTE: SWITCH SETTINGS FURTHER EXPLAINED IN FIGURES 3-2 (EPROM) AND 3-3 (RAM)

FIGURE 3-1. MEMORY CONFIGURATION SWITCH

Also shown in Figure 3-2 are EPROM memory address configurations for 8K words and 4K words, all switch selectable.

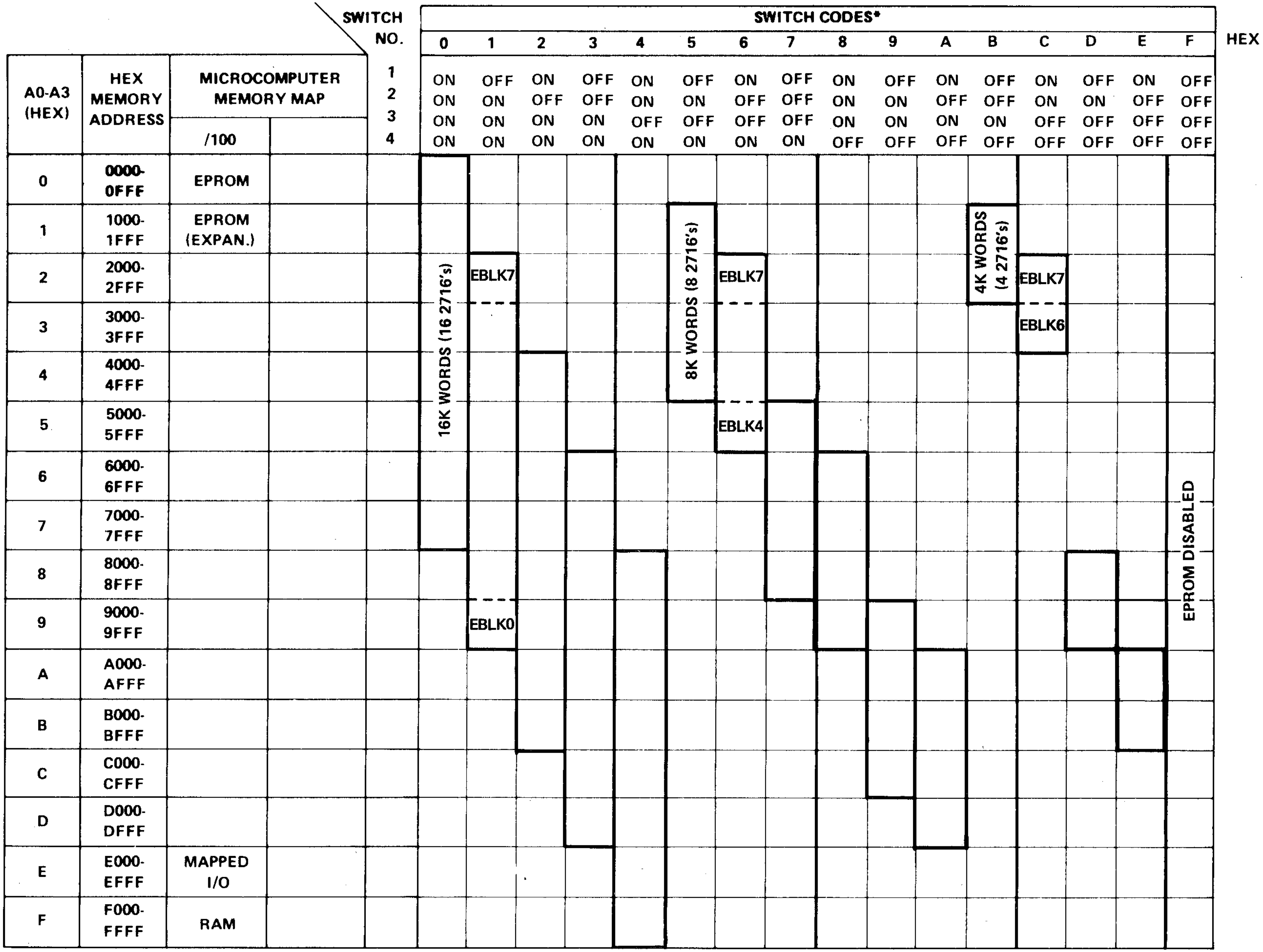
Switch decoding for RAM selection is shown in Figure 3-3. For example, if switches 5 to 8 are set to OFF-OFF-OFF-ON, the memory decoder will select 6K of RAM using addresses 0000_{16} to $2FFF_{16}$.

Not all EPROM's or RAM's populated on the board need be selected by switch S1. Memory not selected by the S1 setting will not be enabled although populated on the board. Conversely, if less memory is populated on the board than designated by switch array S1, the decode logic will address memory as if it was populated as shown by S1.

CAUTION

> The user must exercise care in configuring the TM 990/201 memory into a system. The memory map of the TM 990/201 must not overlay memory on other boards in the system. The resulting data bus conflict may cause damage to data bus drivers on the TM 990/201 or other boards in the system. Note that Figures 3-2 and 3-3 contain blocks to show the memory configuration on the microcomputer board. Consider this memory when selecting expansion memory configurations.

Figures 3-2 and 3-3 show that setting switches 1 to 4 or 5 to 8 to all OFF (OFF-OFF-OFF-OFF) will disable all EPROM or RAM.

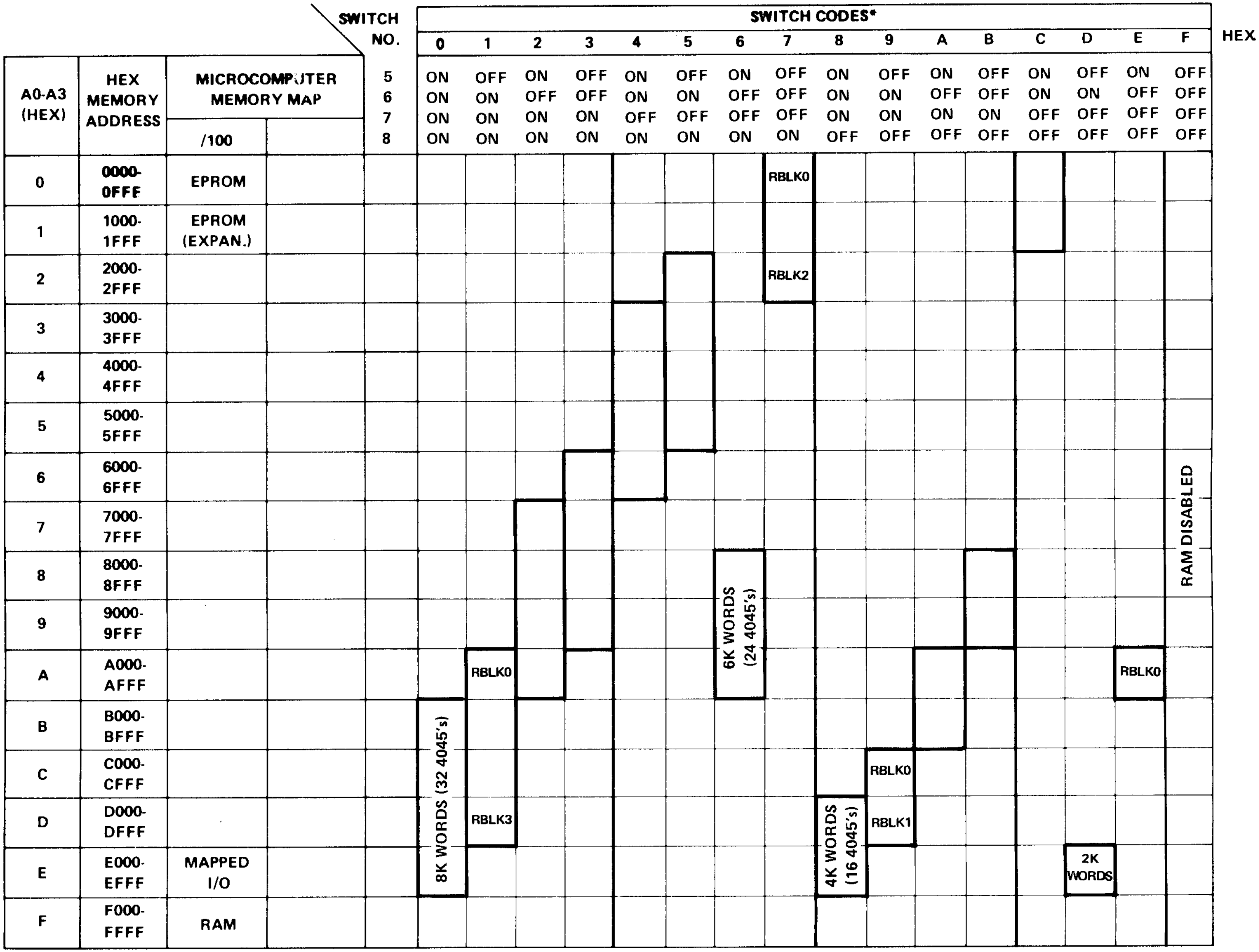


SWITCH NO.	SWITCH CODES* 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F HEX
1	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF
2	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF
3	ON	ON	ON	ON	OFF	OFF	OFF	OFF	ON	ON	ON	ON	OFF	OFF	OFF	OFF
4	ON	ON	ON	ON	ON	ON	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

A0-A3 (HEX)	HEX MEMORY ADDRESS	MICROCOMPUTER MEMORY MAP /100	
0	0000- 0FFF	EPROM	
1	1000- 1FFF	EPROM (EXPAN.)	
2	2000- 2FFF		
3	3000- 3FFF		
4	4000- 4FFF		
5	5000- 5FFF		
6	6000- 6FFF		
7	7000- 7FFF		
8	8000- 8FFF		
9	9000- 9FFF		
A	A000- AFFF		
B	B000- BFFF		
C	C000- CFFF		
D	D000- DFFF		
E	E000- EFFF	MAPPED I/O	
F	F000- FFFF	RAM	

*OFF = binary "1"
ON = binary "0"

FIGURE 3-2. TM 990/201 EPROM MEMORY CONFIGURATIONS



A0-A3 (HEX)	HEX MEMORY ADDRESS	MICROCOMPUTER MEMORY MAP		SWITCH NO.	SWITCH CODES*																
					0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	HEX
		/100		5	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	
				6	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	
				7	ON	ON	ON	ON	OFF	OFF	OFF	OFF	ON	ON	ON	ON	OFF	OFF	OFF	OFF	
				8	ON	ON	ON	ON	ON	ON	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	
0	0000- 0FFF	EPROM										RBLK0									
1	1000- 1FFF	EPROM (EXPAN.)																			
2	2000- 2FFF											RBLK2									
3	3000- 3FFF																				
4	4000- 4FFF																				
5	5000- 5FFF																				
6	6000- 6FFF																			RAM DISABLED	
7	7000- 7FFF																				
8	8000- 8FFF										6K WORDS (24 4045's)										
9	9000- 9FFF																				
A	A000- AFFF					RBLK0													RBLK0		
B	B000- BFFF				8K WORDS (32 4045's)																
C	C000- CFFF													RBLK0							
D	D000- DFFF					RBLK3							4K WORDS (16 4045's)	RBLK1							
E	E000- EFFF	MAPPED I/O																2K WORDS			
F	F000- FFFF	RAM																			

*OFF = 1, ON = 0

FIGURE 3-3. TM 990/201 RAM MEMORY CONFIGURATIONS

3.2.2 MEMORY PLACEMENT BY BLOCKS

EPROM is organized into eight 2K-word blocks designated EBLKO to EBLK7; RAM is organized into four 2K-word blocks designated RBLKO to RBLK3 as shown in Figure 3-4.

Encoded as the beginning of on-board memory is EPROM block EBLK7 and RAM block RBLKO. In other words, memory decode logic will map memory from low to high in the following order of blocks:

EPROM	RAM
EBLK7 (lowest address)	RBLKO (lowest address)
EBLK6	RBLK1
EBLK5	RBLK2
EBLK4	RBLK3 (highest address)
EBLK3	
EBLK2	
EBLK1	
EBLKO (highest address)	

As shown in Figure 3-4, each block of RAM consists of two 4-chip rows of TMS 4045's. Each row consists of 1K by 16-bits, with the bottom row at the lower addresses and the upper row at the higher addresses.

3.2.2.1 EPROM Examples

If the memory configuration switch is set to a code of OFF-ON-OFF-ON, indicating 8K words of EPROM, the following will be mapped by the memory decode logic:

BLOCK	MEMORY ADDRESS
EBLK7	1000_{16} to $1FFF_{16}$
EBLK6	2000_{16} to $2FFF_{16}$
EBLK5	3000_{16} to $3FFF_{16}$
EBLK4	4000_{16} to $4FFF_{16}$

NOTE

Even though other EPROM blocks may be populated, only those denoted in Figure 3-2 will be selected for a given setting of switch S1.

If the memory configuration switch is set to a code of OFF-OFF-ON-OFF, indicating 4K of EPROM, the following will be mapped by the memory decode logic:

BLOCK	MEMORY ADDRESS
EBLK7	8000_{16} to $8FFF_{16}$
EBLK6	9000_{16} to $9FFF_{16}$

3.2.2.2 RAM Examples

If the memory configuration switch is set to a code of OFF-OFF-OFF-ON, the following will be mapped by the memory decode logic:

BLOCK	MEMORY ADDRESS
RBLK0	0000_{16} to $0FFF_{16}$
RBLK1	1000_{16} to $1FFF_{16}$
RBLK2	2000_{16} to $2FFF_{16}$

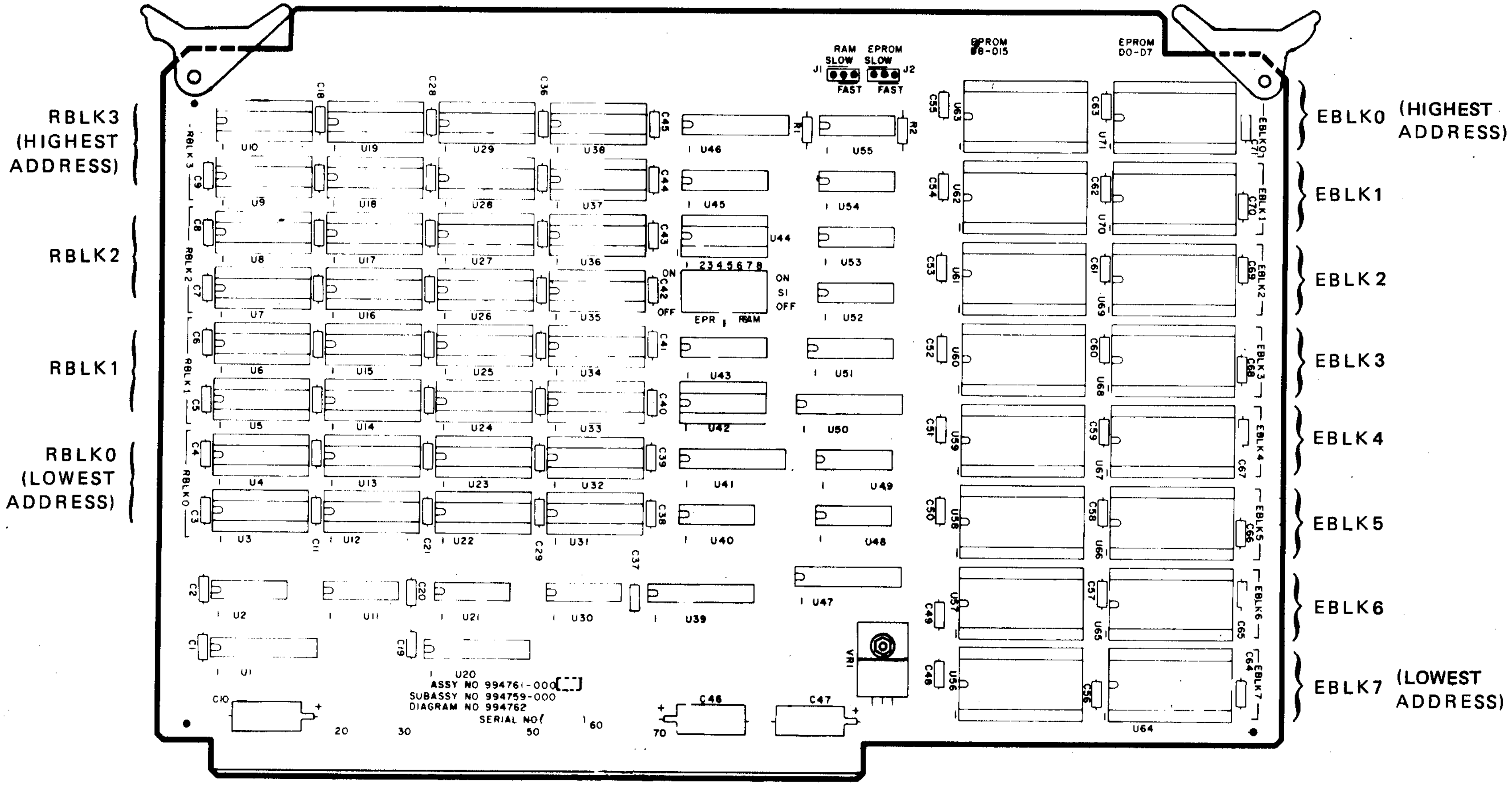


FIGURE 3-4. MEMORY BLOCK LOCATIONS

SECTION 4

THEORY OF OPERATION

4.1 GENERAL

This section covers the theory of operation of the TM 990/201. Information in the *TMS 9900 Microprocessor Data Manual* can supplement the material in this section. Figure 4-1 is a block diagram of the TM 990/201 memory board. Figure 1-1 is a picture of the TM 990/201 detailing the position of its primary components.

4.2 STATIC RAM SECTION

The static RAM section of the TM 990/201 expansion memory board utilizes TMS 4045, 1K X 4 bit static RAM. Table 1-1 defines the product matrix and the amount of RAM on each board. A fully populated TM 990/201 consists of four 2K X 16 blocks of RAM. These blocks are designated RBLKO, RBLK1, RBLK2, and RBLK3.

For the TM 990/201, RBLKO always appears first in the RAM address space followed by RBLK1. RBLK3 is always the last 2K word block of RAM decoded. The block numbers are designated in silkscreen on the TM 990/201.

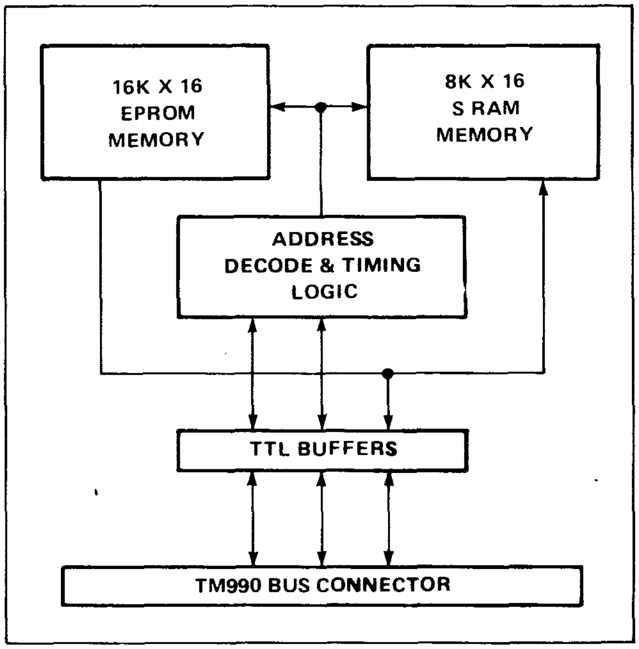


FIGURE 4-1. TM 990/201 BLOCK DIAGRAM

4.3 EPROM SECTION

The EPROM section of the TM 990/201 memory board utilize TMS 2716, 2K X 8 bit EPROM. Table 1-1 defines the product matrix and the amount of EPROM on each board. A fully populated TM 990/201 consists of eight 2K X 16 blocks of EPROM. These are designated EBLK0 through EBLK7.

EBLK7 is always decoded first in the EPROM address space followed by EBLK6. EBLK0 always appears as the last 2K word block of memory on the TM 990/201. The block numbers are designated in the board's silkscreen.

4.4 ADDRESS MAP OPTIONS

The TM 990/201 can be configured in a variety of ways into the address map of a TM 990/100 series microcomputer system. The switch states of switch array S1, an 8-position DIP switch array, uniquely determine the mapping of the TM 990/201 EPROM and RAM memory arrays. Switches 1 through 4 select the mapping of the EPROM memory array. Switches 5 through 8 select the RAM mapping. Figure 4-2 summarizes in block diagram form the address decode logic. Selection of memory mapping using S1 is explained in Section 3.

The possible maps for the EPROM and RAM arrays are shown in Figures 3-2 and 3-3. As explained in Section 3, each switch code corresponds to a unique location of the EPROM or RAM in the memory address space. The switch code implies the starting address of the entire block of EPROM or RAM on the TM 990/201 and the amount of EPROM or RAM selected on the TM 990/201.

The EPROM decode logic maps the EPROM into a contiguous memory space. EPROM block 7 (designated EBLK7 in silkscreen) is mapped into the first 2K word block of this address space followed by block 6 and so on. Block 0 is the last block mapped. This is true of all EPROM mapping options. For code 5, EBLK7 is mapped into 1000_{16}—$1FFF_{16}$; EBLK6 into 2000_{16}—$2FFF_{16}$; EBLK5 into 3000_{16}—$3FFF_{16}$ and EBLK4 into 4000_{16}—$4FFF_{16}$. All other EPROM blocks are disabled for code 5 even if they are populated.

The RAM decode logic also maps the RAM array into a contiguous address space. RAM block 0 (RBLK0 designated in silkscreen) is always mapped into the first 2K words of the space. This is always followed by block 1 and so on. Block 3 is mapped into the last 2K word block.

The decode logic permits RAM precedence over EPROM if both RAM and EPROM are configured in the same address space. This feature is very convenient in debugging programs which will be ROM resident. They may be debugged in RAM on the TM 990/201. Once they are "clean", they may be programmed directly into EPROM without relocation and the attendant relinking.

4.5 EPROM DECODE LOGIC

Figure 4-3 depicts the EPROM decode logic. Switch positions 1 through 4 select a 16-"nibble"* block of memory in the 74S287. Each block corresponds to 1 of 16 possible EPROM address maps. Each nibble in the block determines:

(1) If a block of EPROM is to be selected during the current memory cycle given the current state of address bits A0-A3.

(2) Which block of EPROM (EBLK7 through EBLK0) is selected.

The three least significant data bits from the 74S287 PROM (D01-D03) select the block of EPROM. A state of 0 corresponds to EPROM block 0 while a state of 7 corresponds to EPROM block 7. The most significant bit (D04) enables the 1-of-8 selector along with the memory enable signal (MEMEN.M) from the microcomputer. The 1-of-8 selector (74LS138) develops the EPROM select signals.

*Nibble — refers to 4 bits (half a byte).

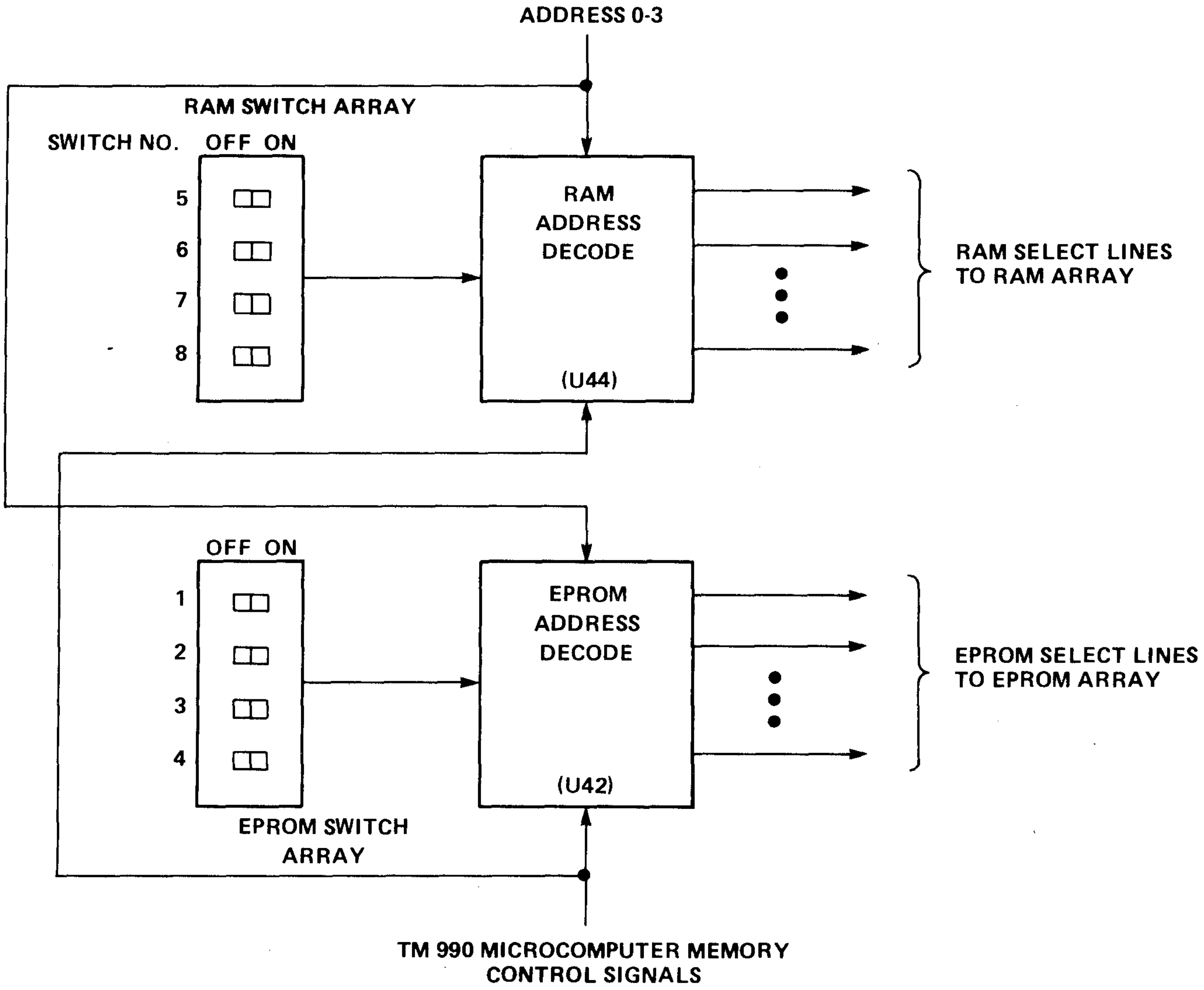


FIGURE 4-2. TM 990/201 ADDRESS DECODE LOGIC BLOCK DIAGRAM

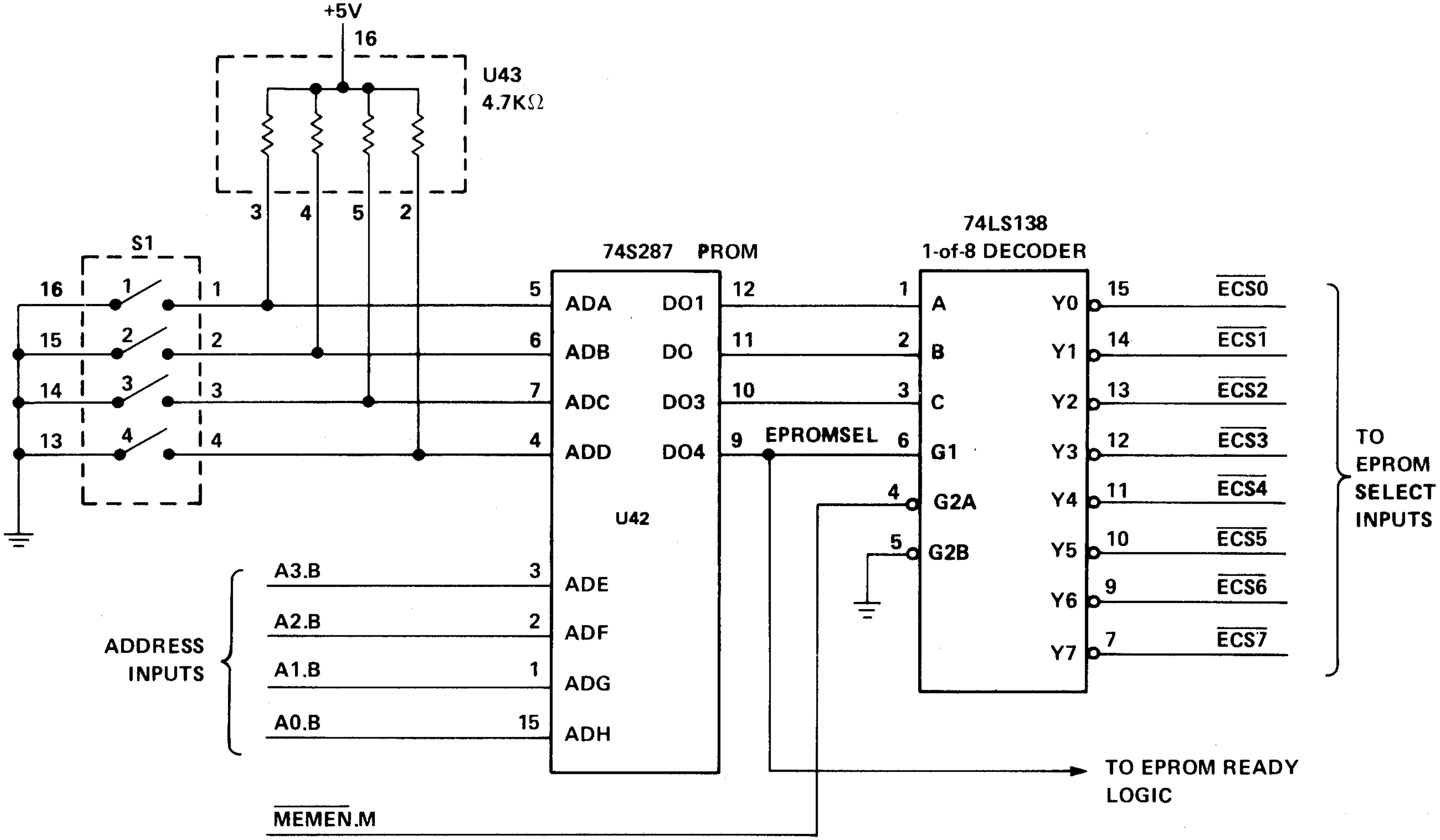


FIGURE 4-3. EPROM DECODE LOGIC

4.6 RAM DECODE LOGIC

Figure 4-4 is the logic diagram of the RAM decode logic. S1 switch positions 5 through 8 select a 16 by 4 block of memory in the 74S287 PROM. Each of the 16 blocks in the PROM corresponds to one of the 16 RAM address maps. Each nibble in the block determines:

(1) If a block of RAM is to be selected during the current memory cycle given the state of address lines A0-A3.

(2) Which block of RAM (RBLK0-RBLK3) is selected.

The two least significant bits of the nibble (D01 and D02) coupled with address bit 4 (EA4) from the microcomputer drive the 1-of-8 decoder in selecting one of the eight 1K X 16-bit banks of RAM. The most significant data bit of the nibble, D04, enables the decoder along with the memory enable signal (MEMEN.M) from the microcomputer. The outputs of the decoder are the select lines to the RAM banks.

The G2B input to the 74LS138 gates the RAM select lines so that no bus conflicts occur between the EPROM and RAM data buffers on the TM 990/201. This is accomplished with NORing the write strobe (WE.M) and data bus in (DBIN) from the microcomputer (detail is shown in Figure 4-8).

4.7 ADDRESSING SUMMARY

- The user has an option of 16 configurations each for RAM and EPROM, and a code OFF-OFF-OFFOFF at S1 allows the user to disable the memories. These options are explained in detail in Section 3.
- The user has the option of programming his own decode configuration, placing memory on any 2K word boundary. See Appendix A for details.
- An overlap of RAM and EPROM on board results in RAM dominance. See caution on page 3-2 for overlap from board to board.

4.8 MEMORY SPEED AND TIMING

This section describes memory speed and outlines timing for the TM 990/201 memory board.

4.8.1 MEMORY SPEED

The TMS 9900 interfaces easily with slow memories. This is accomplished through the use of the "wait state" concept. During each memory cycle, the microprocessor samples the READY signal. When READY is active high, it indicates to the microprocessor that memory will be ready to read or write during the next clock cycle. When not-ready is indicated during a memory operation, the TMS 9900 enters a wait state and suspends internal operation until the memory system indicates it is ready to proceed.

The READY signal is generated on the TM 990/201 expansion memory board separately for RAM and EPROM. The board will be populated with TMS 2716 EPROMs that have an access time of 450 ns and TMS 4045 static RAMs that have an access time in accordance with Table 1-1. At 3 MHz, the EPROMs and RAMs with access times in excess of 300 ns require one wait state. There are jumper provisions on the board to disable READY for RAM, EPROM, or both. Jumper J1 designates the memory speed for RAM used, and jumper J2 designates the EPROM speed. Table 4-1 shows the necessary jumper setting according to clock used and memory access time.

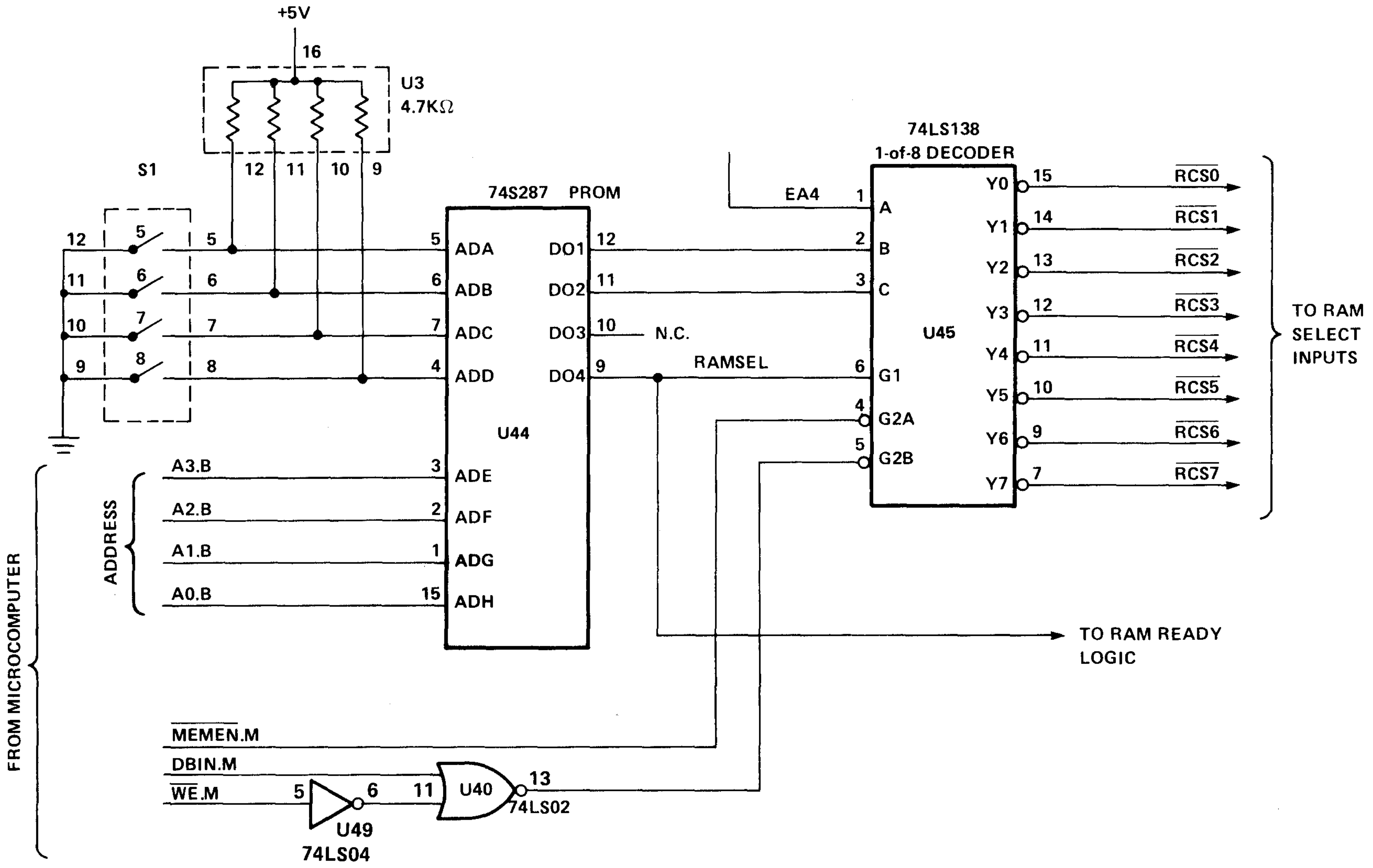


FIGURE 4-4. RAM DECODE LOGIC

TABLE 4-1. FAST/SLOW JUMPER (J1/J2) POSITIONS VERSUS MEMORY ACCESS TIME

ACCESS TIME	CPU OPERATION	
	3 MHz	4 MHz
450 ns	SLOW	SLOW
300 ns	FAST	SLOW
200 ns	FAST	FAST
150 ns	FAST	FAST

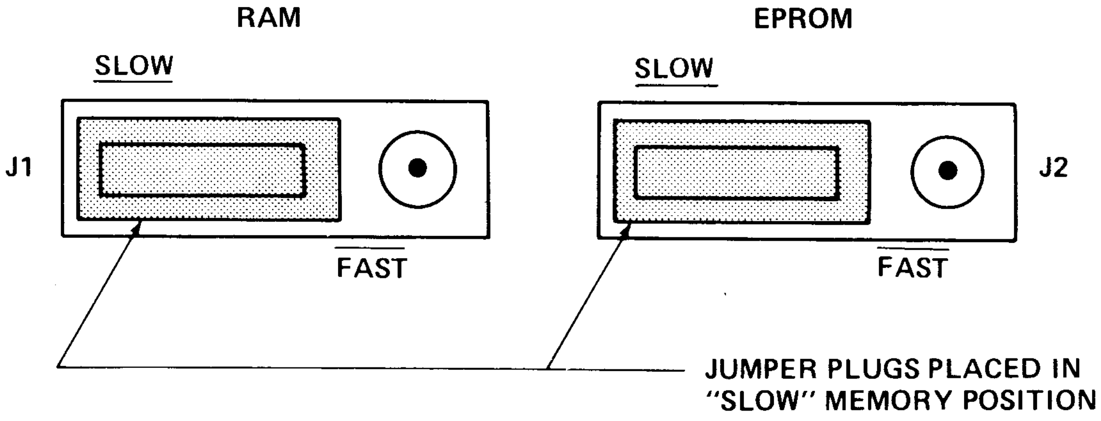
Figures 1-1 and 4-5 shows the jumper positions on the board for RAM (J1) and EPROM (J2). The jumpers are in the SLOW position; this is how the board is shipped.

The speed setting for RAM and EPROM are for *all* RAM or *all* EPROM; therefore, if the addition or replacement of memories becomes necessary, the user must take into account the speed of the devices used. If the access time of the slowest RAM being added is more than 300 ns, the RAM jumper must be placed in the SLOW position. This assumes 3 MHz operation. Refer to Table 4-1 for the proper settings. If the RAM supplied with the board operates in the "fast" mode and the jumper is in the "SLOW" position, system performance will be less than optimal.

4.8.2 MEMORY TIMING

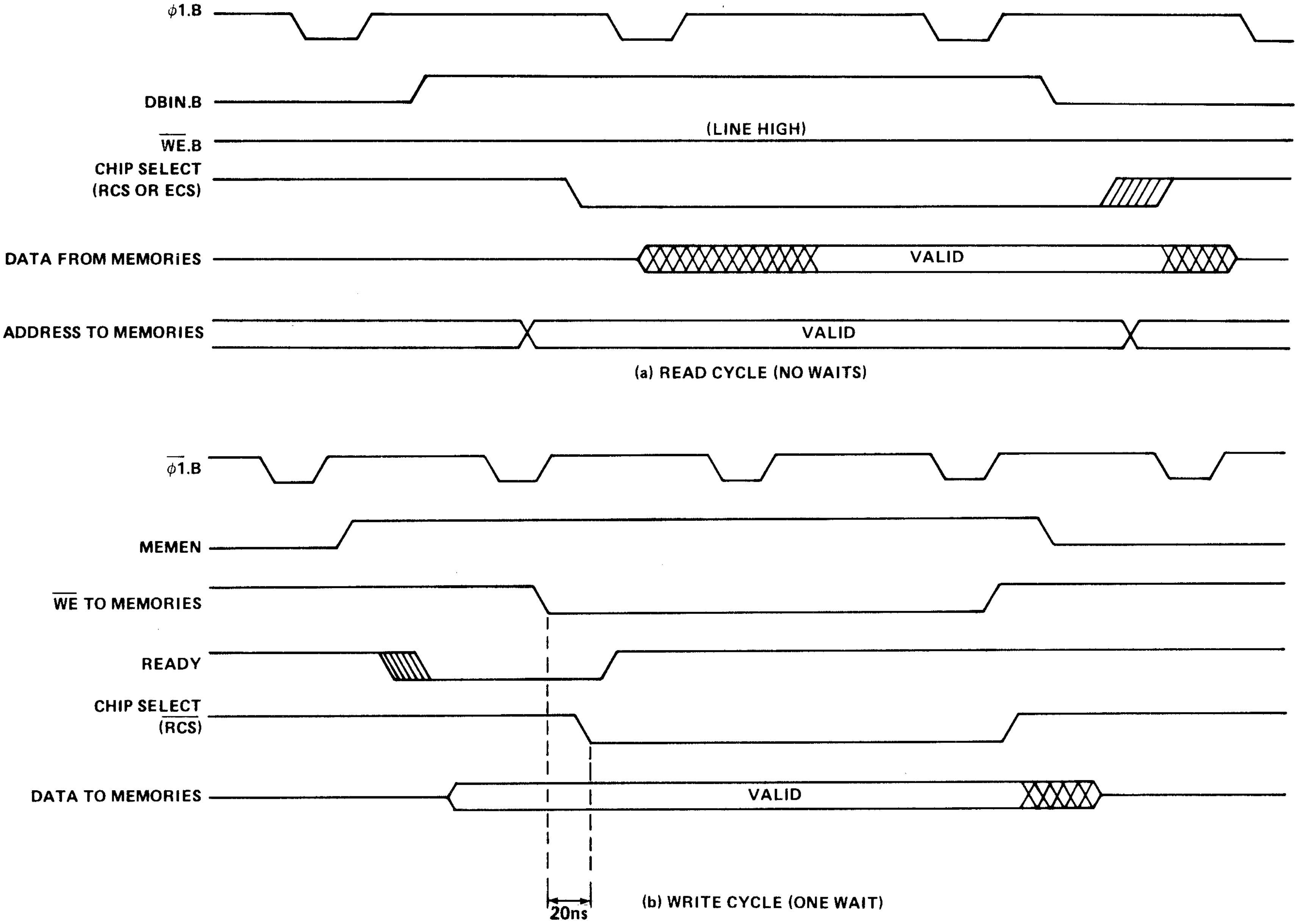
The memory timing for the TM 990/201 board is shown in Figure 4-6. Memory write timing is shown with one wait state and read cycle timing is shown with none.

Care must be taken when interfacing the TMS 4045 static RAM to the TMS 9900. During a write cycle, the chip select ($\overline{S}$) to the RAM's must be held high (inactive) until after $\overline{WE}$ goes low. Otherwise the RAM's enter a read mode and enable their output buffers. The output buffers in the RAM's would fight against the data bus drivers and data would be lost. This condition would persist for approximately 1 clock cycle until $\overline{WE}$ is output by the TMS 9900 microprocessor.



NOTE: Only RAM jumper J1 provided on the TM 990/206 board.

FIGURE 4-5. SLOW/FAST MEMORY JUMPER PLACEMENT



(a) READ CYCLE (NO WAITS)

(b) WRITE CYCLE (ONE WAIT)

FIGURE 4-6. TM 990/201 MEMORY TIMING

The hardware on the TM 990/201 memory board resolves this problem. In essence the $\overline{S}$ signal to the RAM's is an "OR"ed function of WE and DBIN. This method is recommended whenever devices with common I/O pins are used.

4.8.3 READY LOGIC

Figure 4-7 depicts the circuitry for the RAM READY logic. The EPROM READY logic is identical. The READY logic forces one wait state during each memory cycle the RAM is accessed.

If Jumper J1 is in the FAST position, READY is never asserted low since the left flip-flop's CLR input is always low. Thus READY is never forced low. If J1 is in the SLOW position, READY is forced low for the first ϕ1 clock period of every RAM memory cycle. The two flip-flops force READY high for the second and third clock cycles as shown in the timing diagram in Figure 4-6.

4.9 RAM PRECEDENCE LOGIC

Figure 4-8 details the RAM precedence logic. During any memory cycle that both RAM and EPROM are selected on the TM 990/201, the EPROM data buffers are placed in the high impedance state. Thus the RAM data buffers are the only buffers allowed to utilize the data bus.

4.10 INTERFACE DESCRIPTION

All of the interface functions for the TM 990/201 memory board are through the chassis backplane. A pin assignment chart is shown in Table 4-2. The signals used are shown with their corresponding pin number on the P1 connector tab.

Figure 4-9 shows the TM 990/510 chassis backplane.

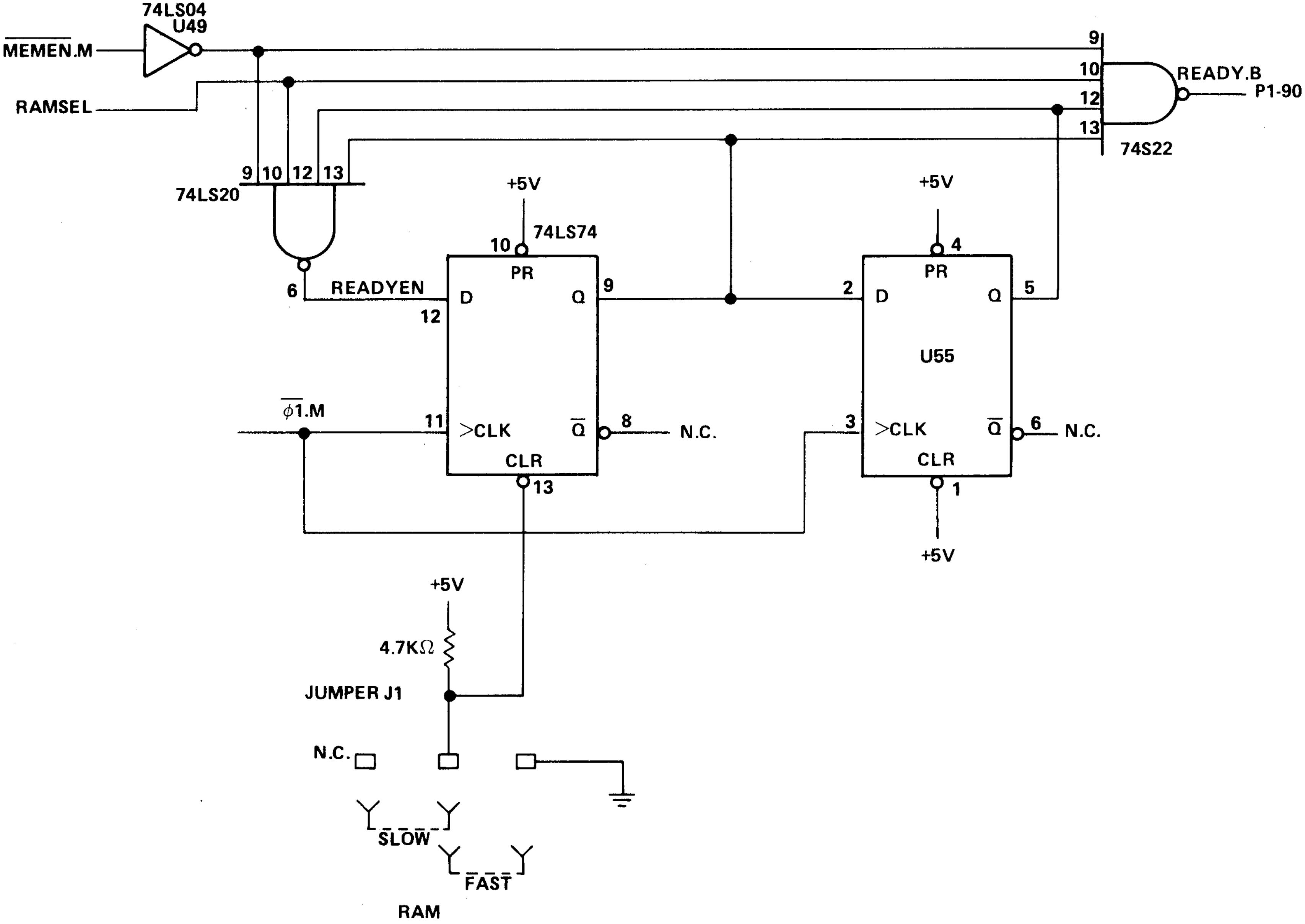


FIGURE 4-7. TM 990/201 RAM READY LOGIC

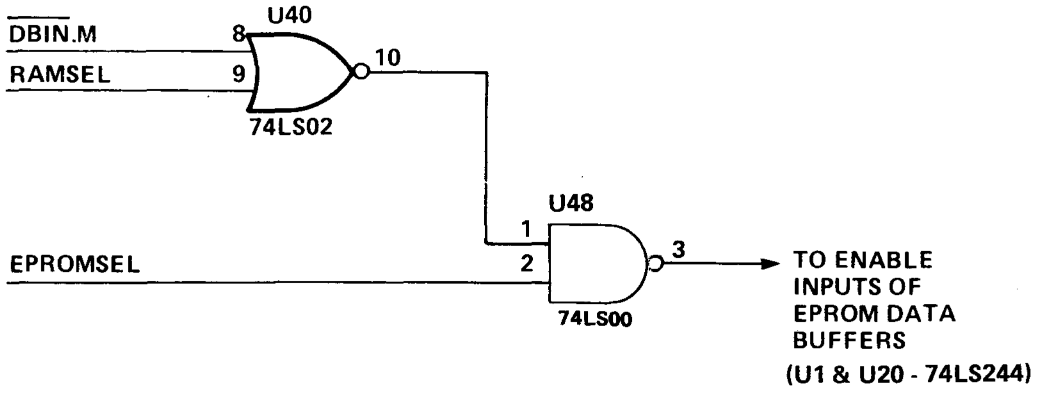


FIGURE 4-8. RAM PRECEDENCE LOGIC

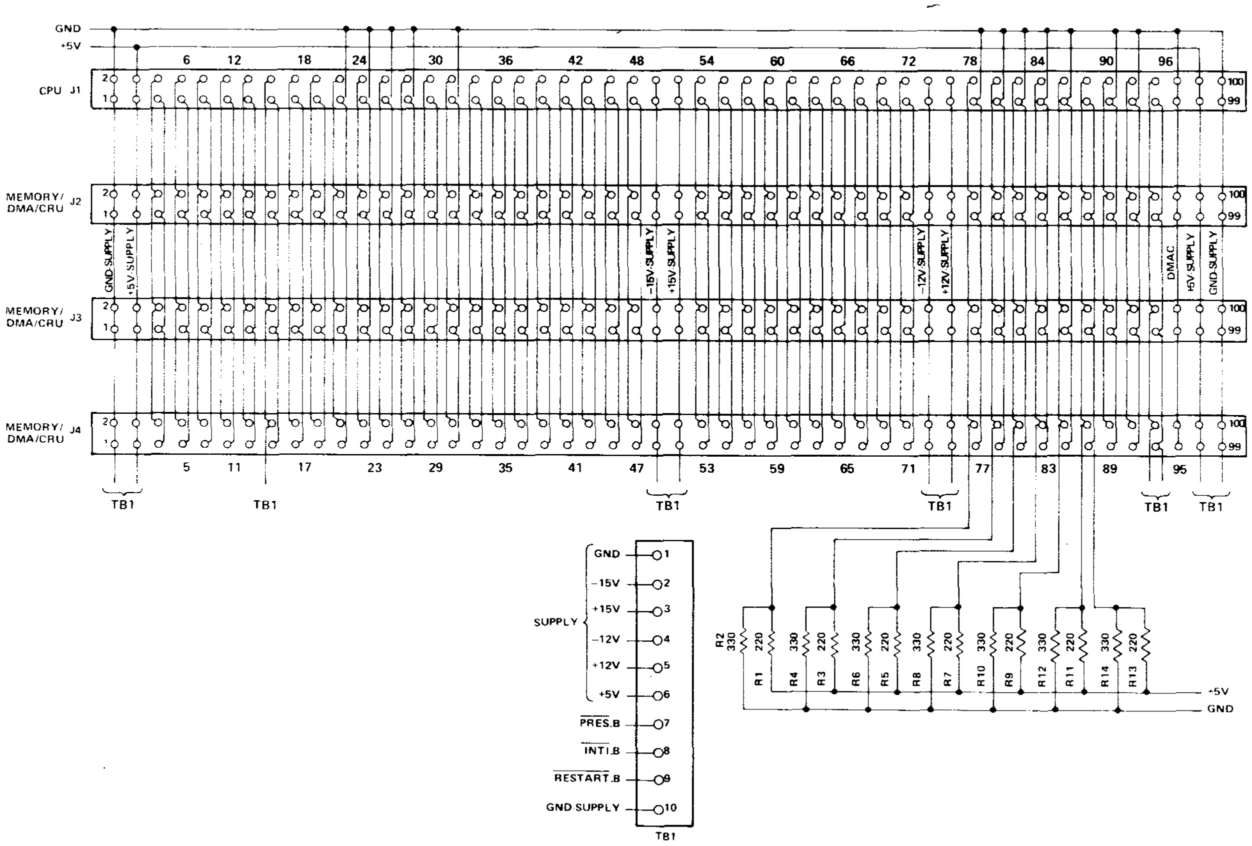


FIGURE 4-9. TM 990/510 OEM CHASSIS BACKPLANE SCHEMATIC

TABLE 4-2. BACKPLANE/P1 PIN ASSIGNMENTS USED BY TM 990/201 BOARD

PIN	SIGNAL	PIN	SIGNAL
1	GND	62	A5.B
2	GND	63	A6.B
3	+5	64	A7.B
4	+5	65	A8.B
21	GND	66	A9.B
22	$\overline{\Phi}$1.B	67	A10.B
23	GND	68	A11.B
25	GND	69	A12.B
27	GND	70	A13.B
31	GND	71	A14.B
33	D0.B	73	-12V
34	D1.B	74	-12V
35	D2.B	75	+12V
36	D3.B	76	+12V
37	D4.B	77	GND
38	D5.B	78	$\overline{\text{WE}}$.B
39	D6.B	79	GND
40	D7.B	80	$\overline{\text{MEMEN}}$.B
41	D8.B	81	GND
42	D9.B	82	DBIN.B
43	D10.B	83	GND
44	D11.B	85	GND
45	D12.B	89	GND
46	D13.B	90	READY.B
47	D14.B	91	GND
48	D15.B	97	+5
57	A0.B	98	+5
58	A1.B	99	GND
59	A2.B	100	GND
60	A3.B		
61	A4.B		

SECTION 5

TM 990/206 RAM-ONLY MEMORY EXPANSION BOARD

5.1 GENERAL

The Texas Instruments TM 990/206 is a RAM-only expansion memory board (shown in Figure 5-1) for use with the TM 990/10XM microcomputer. Its features include:

- Up to 8K words of TMS 4045 static RAM (1024 X 4 bits each)
- TTL compatible interface
- 4 MHz operating capability
- Single power supply (+5 V)

The operation of the TM 990/206 is essentially the same as the TM 990/201. The differences between the two include:

- The TM 990/206 has no EPROM memory
- The memory (RAM) configurations differ (the program in the RAM address decode PROM, U44, is also different for the two boards).
- The power requirements differ.

These differences are explained in more detail in Sections 5.2 and 5.3.

Table 5-1 defines the product matrix for the TM 990/206 RAM memory board.

TABLE 5-1. TM 990/206 PRODUCT MATRIX

MODEL	SOCKETS PROVIDED	SOCKETS POPULATED	RAM ACCESS TIME
TM 990/206-41	8K X 16 (RBLKO-RBLK3)	4K X 16 (RBLKO-RBLK1)	450 ns
TM 990/206-42	8K X 16 (RBLKO-RBLK3)	8K X 16 (RBLKO-RBLK3)	450 ns

NOTE: Block nomenclature is shown in Figure 3-4.

5.2 SPECIFICATIONS

Board dimensions:	See Figure 1-2.
Temperature range:	Operating: 0°C to 70°C Storage: —40°C to 70°C

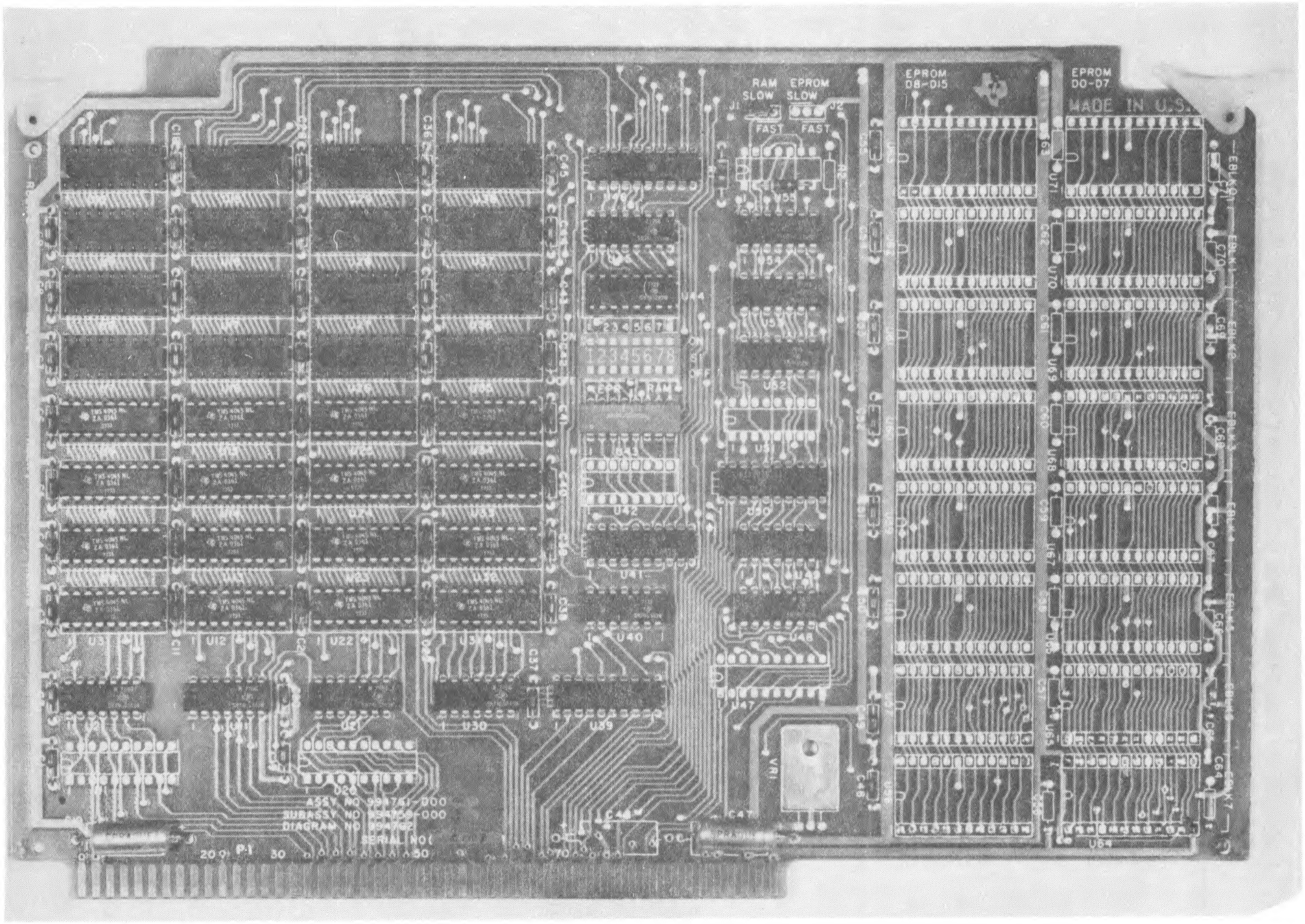


FIGURE 5-1. TM 990/206 MEMORY EXPANSION BOARD

Clock rate:	The TM 990/206 memory expansion board is compatible with the TM 990/10XM CPU at 3 MHz and can be operated at 4 MHz.
Devices utilized:	TM 4045-45 static RAM, 1K X 4
Power:	See Table 5-2.

TABLE 5-2. TM 990/206 POWER CONSUMPTION VS. SIZES

MODEL	MEMORY SIZE	+5 V (±5%)	
		MAX	TYP
TM 990/206-41	4K X 16 RAM	2.5A	1.3A
TM 990/206-42	8K X 16 RAM	5.5A	2.15A

5.3 INSTALLATION AND OPERATION

Installation and operation of the TM 990/206 board is the same as explained in Section 2 except for the example in paragraph 2.8. An example using the TM 990/206 memory and memory mapping is provided in paragraph 5.6.

5.4 MEMORY PLACEMENT AND SELECTION

The TM 990/206 RAM memory board is decoded in a similar manner as for the TM 990/201 EPROM/RAM board as explained in Section 3 except that the TM 990/206 does not contain EPROM and its RAM configuration differs.

S1 is essentially the same as described in paragraph 3.2.1 except that only switches 5 to 8 are used since this is a RAM-only board. The TM 990/206 memory map is shown in Figure 5-2. RAM block placement is the same as in paragraph 3.2.2:

RBLKO (lowest address)
RBLK1
RBLK2
RBLK3 (highest address)

Only Jumper J1 to select RAM speed is provided on the TM 990/206 board. J1 operates the same for either board as explained in paragraph 2.6.

5.5 OPERATION

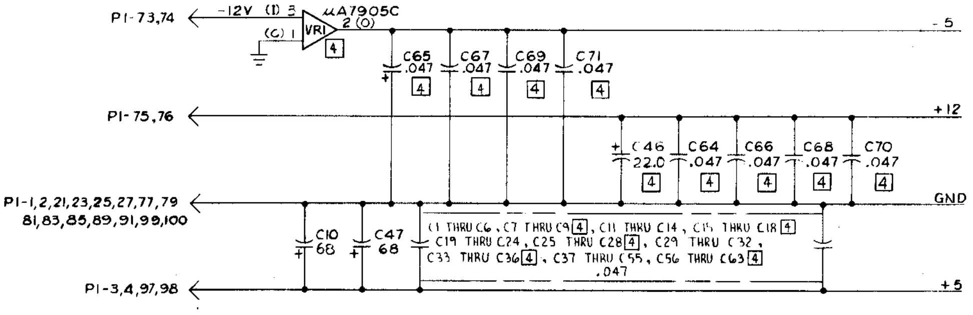
Essentially the user needs only to choose the correct memory configuration (Section 3), insert the board into the chassis, and apply power to set up the system for operation.

The operation of the TM 990/206 memory board should be transparent to the user in that no special signals are required other than those supplied through the backplane. If the TM 990/510 chassis is not used, refer to Section 4.10 for interface information.

NOTES: UNLESS OTHERWISE SPECIFIED:

1. CAPACITANCE VALUES ARE IN MICROFARADS
2. RESISTANCE VALUES ARE IN OHMS
3. ALL RESISTORS ARE 1/4W, ± 5 %

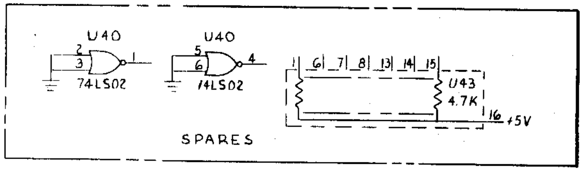
[4] THE COMPONENTS ARE INSTALLED IN ASSEMBLIES 994761-0001 THRU 994761-0005 PER TABLE I



REFERENCE DESIGNATOR	DEVICE	ASSEMBLY TM990/201-41 (994761-0001)	TM990/201-42 (994761-0002)	TM990/201-43 (994761-0003)	TM990/206-41 (994761-0004)	TM990/206-42 (994761-0005)
U5 - U6	TMS 4045		X	X	X	X
U14 - U15	TMS 4045		X	X	X	X
U24 - U25	TMS 4045		X	X	X	X
U33 - U34	TMS 4045		X	X	X	X
U7 - U10	TMS 4045			X		X
U16 - U19	TMS 4045			X		X
U26 - U29	TMS 4045			X		X
U35 - U38	TMS 4045			X		X
U56 - U57	TMS 2716	X	X	X		
U64 - U65	TMS 2716	X	X	X		
U58 - U59	TMS 2716		X	X		
U66 - U67	TMS 2716		X	X		
U60 - U63	TMS 2716			X		
U68 - U71	TMS 2716			X		
U1, U20	74LS244	X	X	X		
U47	74LS241	X	X	X		
U51	74LS138	X	X	X		
U55	74LS74	X	X	X		
VR1	μA7905	X	X	X		
U42	74S287	X	X	X		
R2	4.7KΩ 1/4 W	X	X	X		
C46	22 μf	X	X	X		
R3	68Ω 1/4 W				X	X
C7-C9	.047 μf		X	X	X	X
C15-C18	.047 μf		X	X	X	X
C25-C28	.047 μf		X	X	X	X
C33-C36	.047 μf		X	X	X	X
C60-C63	.047 μf		X	X		
C68-C71	.047 μf		X	X		
C56-C59	.047 μf	X	X	X		
C64-C67	.047 μf	X	X	X		

TABLE I

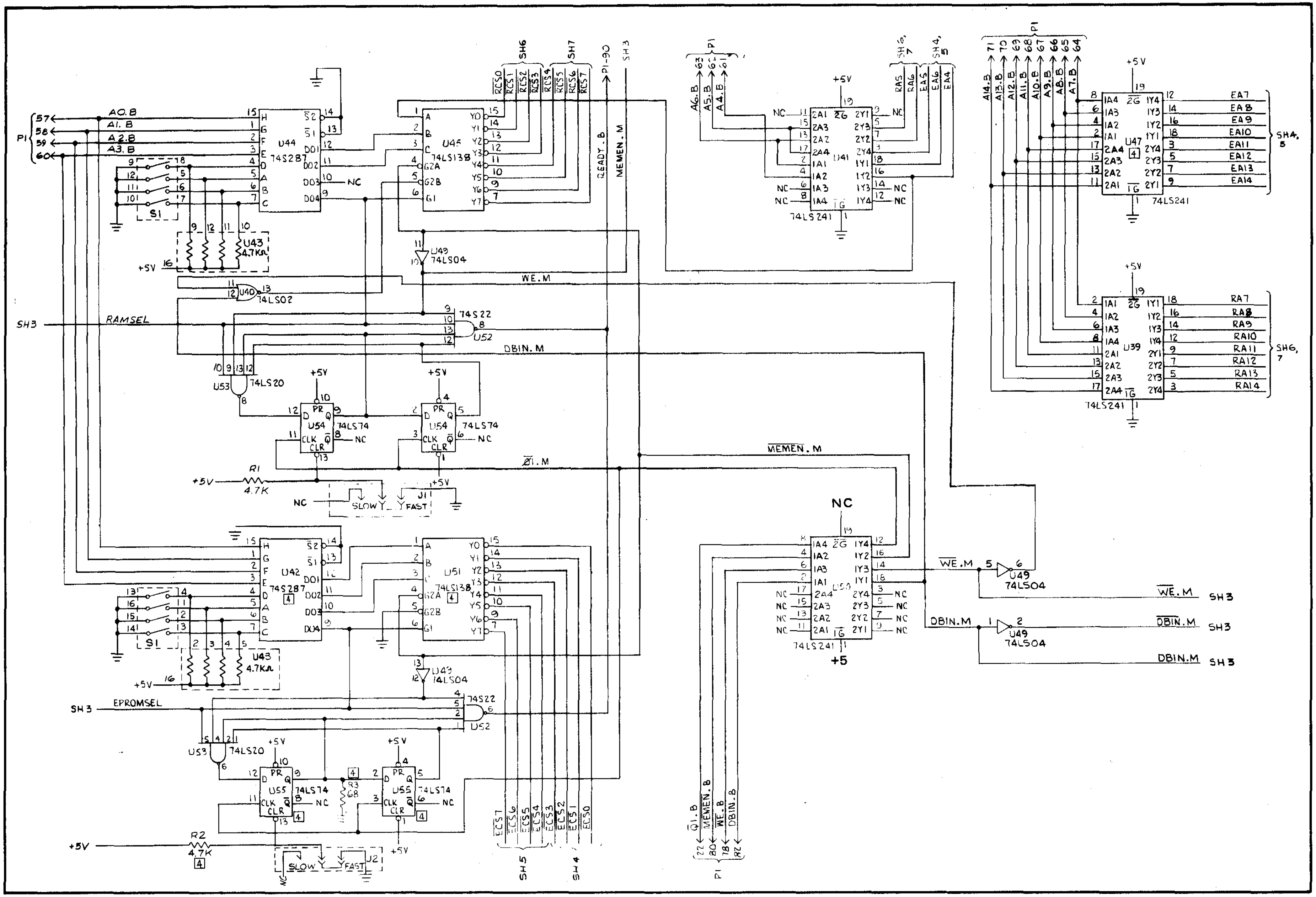
DEVICE	REF DESIGNATOR	VCC(+5V) PIN	GND PIN
74LS00N	U48	14	7
74LS02N	U40		
74LS04N	U49		
74LS20N	U53		
74S22N	U52		
74LS74N	U54, U55		
74LS243N	U2, U11, U21, U30	14	7
74S287N	U42, U44	16	8
74LS138N	U45, U51	16	8
74LS241N	U39, U41, U47, U50	20	10
74LS244N	U1, U20, U46	20	10
TMS2716	U56 THRU U71	24	12



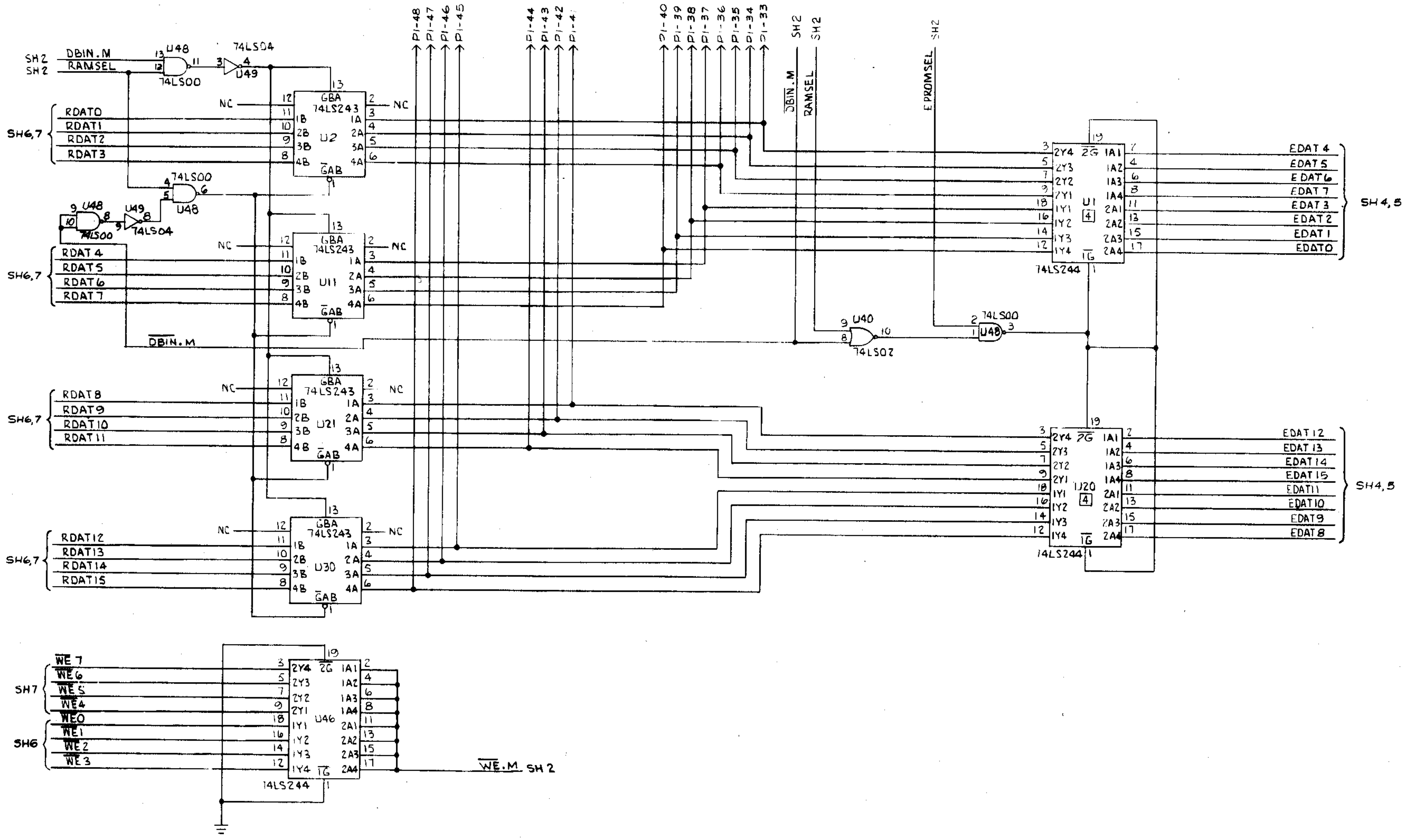
SPARES

DRAWING 994762 Rev. A

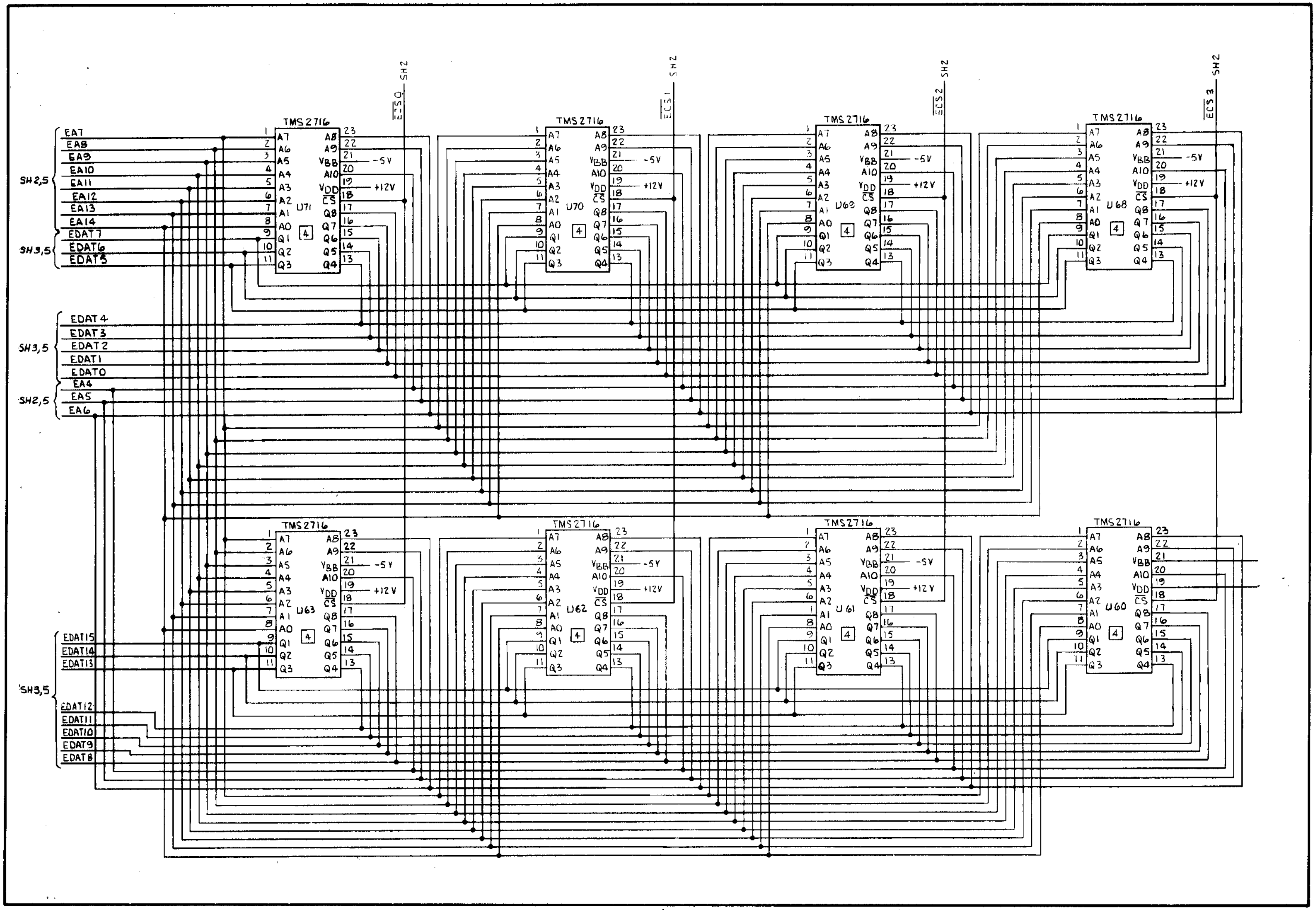
LOGIC DIAGRAM, TM 990/201
SHEET 1 OF 7



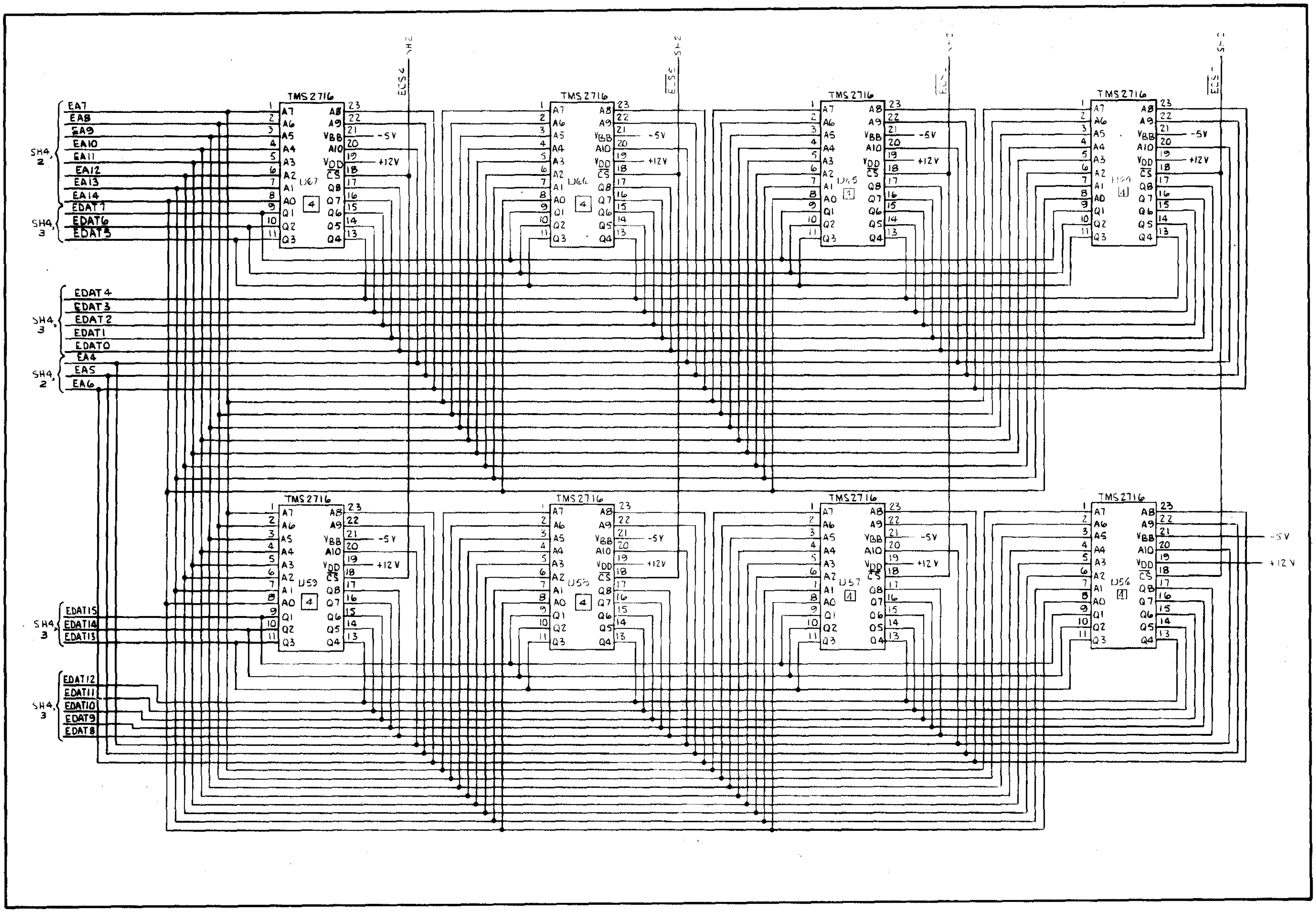
LOGIC DIAGRAM, TM 990/201
SHEET 2 OF 7



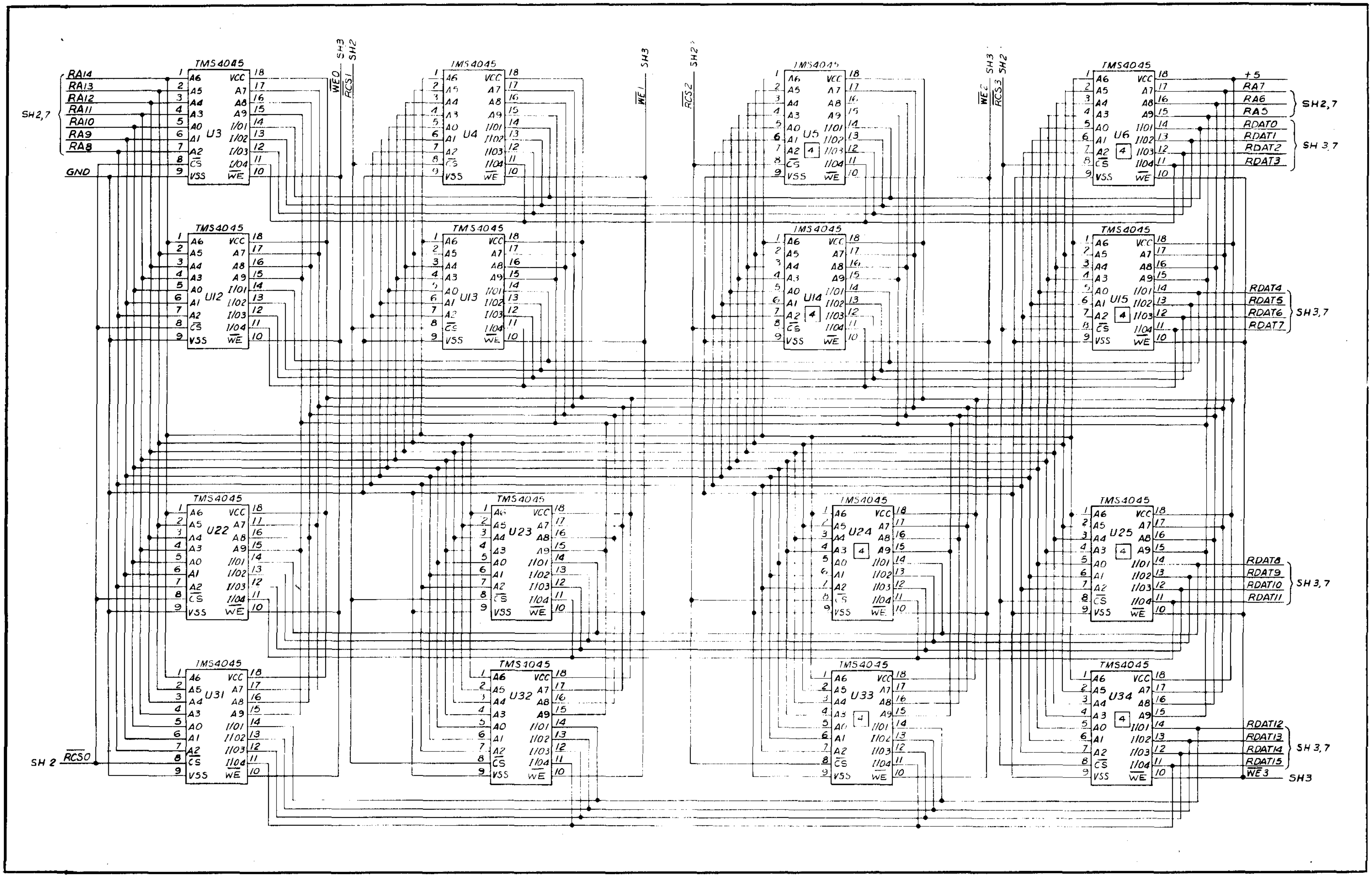
LOGIC DIAGRAM, TM 990/201
SHEET 3 OF 7



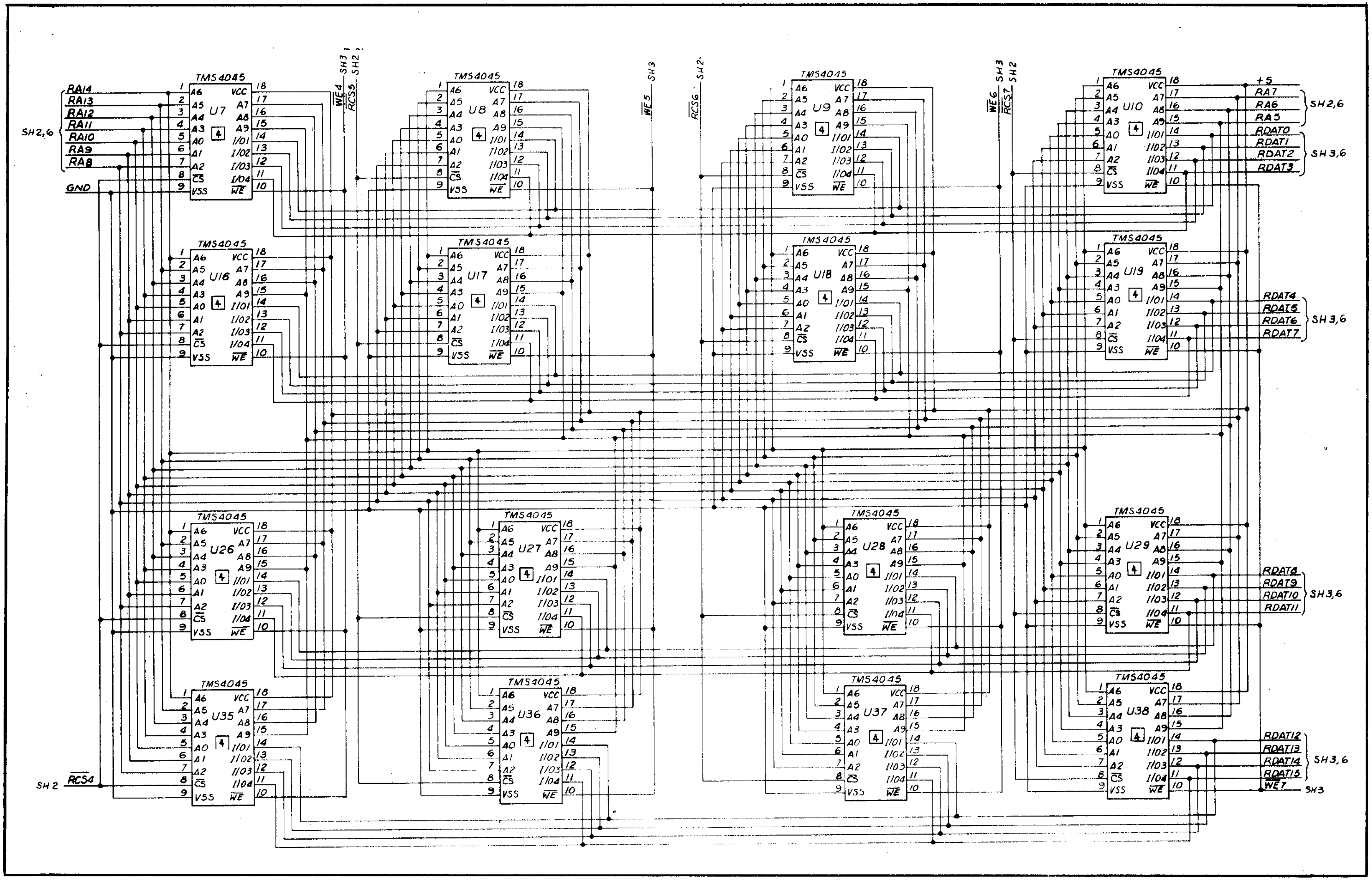
LOGIC DIAGRAM, TM 990/201
SHEET 4 OF 7



LOGIC DIAGRAM, TM 990/201
SHEET 5 OF 7



LOGIC DIAGRAM, TM 990/201
SHEET 6 OF 7



LOGIC DIAGRAM, TM 990/201
SHEET 7 OF 7

APPENDIX C
PROM PROGRAM SHEETS

				SWITCH NO.	SWITCH CODES*															
					0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	HEX
A0-A3 (HEX)	HEX MEMORY ADDRESS	MICROCOMPUTER MEMORY MAP			ON ON ON ON	OFF ON ON ON	ON OFF ON ON	OFF OFF ON ON	ON ON OFF ON	OFF ON OFF ON	ON OFF OFF ON	OFF OFF OFF ON	ON ON ON OFF	OFF ON ON OFF	ON OFF ON OFF	OFF OFF ON OFF	ON ON OFF OFF	OFF ON OFF OFF	ON OFF OFF OFF	OFF OFF OFF OFF	
		/100																			
0	0000- 0FFF	EPROM																			
1	1000- 1FFF	EPROM (EXPAN.)																			
2	2000- 2FFF																				
3	3000- 3FFF																				
4	4000- 4FFF																				
5	5000- 5FFF																				
6	6000- 6FFF																				
7	7000- 7FFF																				
8	8000- 8FFF																				
9	9000- 9FFF																				
A	A000- AFFF																				
B	B000- BFFF																				
C	C000- CFFF																				
D	D000- DFFF																				
E	E000- EFFF	MAPPED I/O																			
F	F000- FFFF	RAM																			

*OFF = LOGIC "1"
ON = LOGIC "0"

				SWITCH NO.	SWITCH CODES*																
					0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	HEX
A0-A3 (HEX)	HEX MEMORY ADDRESS	MICROCOMPUTER MEMORY MAP			ON ON ON ON	OFF ON ON ON	ON OFF ON ON	OFF OFF ON ON	ON ON OFF ON	OFF ON OFF ON	ON OFF OFF ON	OFF OFF OFF ON	ON ON ON OFF	OFF ON ON OFF	ON OFF ON OFF	OFF OFF ON OFF	ON ON OFF OFF	OFF ON OFF OFF	ON OFF OFF OFF	OFF OFF OFF OFF	
		/100																			
0	0000- 0FFF	EPROM																			
1	1000- 1FFF	EPROM (EXPAN.)																			
2	2000- 2FFF																				
3	3000- 3FFF																				
4	4000- 4FFF																				
5	5000- 5FFF																				
6	6000- 6FFF																				
7	7000- 7FFF																				
8	8000- 8FFF																				
9	9000- 9FFF																				
A	A000- AFFF																				
B	B000- BFFF																				
C	C000- CFFF																				
D	D000- DFFF																				
E	E000- EFFF	MAPPED I/O																			
F	F000- FFFF	RAM																			

*OFF = LOGIC "1"
ON = LOGIC "0"

				SWITCH NO.	SWITCH CODES*																
					0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	HEX
A0-A3 (HEX)	HEX MEMORY ADDRESS	MICROCOMPUTER MEMORY MAP			ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	
					ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	
					ON	ON	ON	ON	OFF	OFF	OFF	OFF	ON	ON	ON	ON	OFF	OFF	OFF	OFF	
		/100			ON	ON	ON	ON	ON	ON	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	
0	0000- 0FFF	EPROM																			
1	1000- 1FFF	EPROM (EXPAN.)																			
2	2000- 2FFF																				
3	3000- 3FFF																				
4	4000- 4FFF																				
5	5000- 5FFF																				
6	6000- 6FFF																				
7	7000- 7FFF																				
8	8000- 8FFF																				
9	9000- 9FFF																				
A	A000- AFFF																				
B	B000- BFFF																				
C	C000- CFFF																				
D	D000- DFFF																				
E	E000- EFFF	MAPPED I/O																			
F	F000- FFFF	RAM																			

*OFF = LOGIC "1"
ON = LOGIC "0"

A0-A3 (HEX)	HEX MEMORY ADDRESS	MICROCOMPUTER MEMORY MAP		SWITCH NO.	SWITCH CODES*																
					0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	HEX
					ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	
					ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	
					ON	ON	ON	ON	OFF	OFF	OFF	OFF	ON	ON	ON	ON	OFF	OFF	OFF	OFF	
		/100			ON	ON	ON	ON	ON	ON	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	
0	0000- 0FFF	EPROM																			
1	1000- 1FFF	EPROM (EXPAN.)																			
2	2000- 2FFF																				
3	3000- 3FFF																				
4	4000- 4FFF																				
5	5000- 5FFF																				
6	6000- 6FFF																				
7	7000- 7FFF																				
8	8000- 8FFF																				
9	9000- 9FFF																				
A	A000- AFFF																				
B	B000- BFFF																				
C	C000- CFFF																				
D	D000- DFFF																				
E	E000- EFFF	MAPPED I/O																			
F	F000- FFFF	RAM																			

*OFF = LOGIC "1"
ON = LOGIC "0"

APPENDIX D

TM 990/422 DEMONSTRATION SOFTWARE

D.1 GENERAL

Optional TM 990/422 demonstration software is available for verifying RAM memory on the TM 990/201 and TM 990/206 memory boards. The software is provided on two TMS 2716 EPROM chips which can be plugged into the EPROM memory areas on the TM 990/10X M microcomputer board or the TM 990/201 memory board. When plugged into the TM 990/201 memory board, it can verify operation of the EPROM array on that board. This software is executed under the TIBUG monitor and uses the utilities provided by the monitor. A source listing of this software is provided as part of this appendix.

The software is completely relocatable in that it can be plugged into any location on the address map (except those locations which are reserved such as TIBUG workspace, interrupt vectors, or load vectors). The entry point is the first address occupied by the EPROM module.

NOTE

> During testing, locations FF90$_{16}$ to FFFF$_{16}$ will be protected from data written to these addresses. This is to ensure protection of the monitor and demonstration software workspaces.

D.2 INSTALLATION

The EPROM's can be installed on either the TM 990/10X M microcomputer board or on the TM 990/201 memory board.

D.2.1 INSTALLATION ON MICROCOMPUTER BOARD

a. Turn off power to the system. Remove TM 990/10X'M board.

b. Remove any EPROM's installed in sockets U43 or U45. Leave the TIBUG EPROM's installed in sockets U42 and U44.

c. Set jumper J2 to 2716.

d. Set jumper J4 to 16 (indicating 2716). Jumper J3 should remain set at 08 (setting for TIBUG EPROM's).

e. Install the EPROM marked U43 in socket U43. Install the EPROM marked U45 in socket U45.

f. Install the board into the system. Reapply power.

g. Call up the TIBUG monitor (toggle the microcomputer board RESET switch, press the character A on the keyboard).

h. Using the R command, set the Program Counter (P=) to 1000 (hexadecimal).

i. Using the E command, execute the software demonstration program:

```
?R
W=FFC6
P=2000  1000
?E
```

D.2.2 INSTALLATION ON TM 990/201 MEMORY BOARD

a. Turn off power to the system. Remove the memory board.

b. On the memory board, place the two TMS 2716 demonstration software chips on adjacent horizontal EPROM sockets (e.g., U56 and U64) with the lowest numbered chip going to the lowest numbered socket (e.g., the U43-marked chip in U56 and the U45-marked chip in U64).

c. Install jumper J2 on the memory board to the SLOW position.

d. Set switches 1 to 4 at S1 to a configuration corresponding to placement of the chips in the memory map. For example, if sockets U56 and U64 are used (this is EBLK7), then any one of the settings can be used as shown in Figure 3-2 of the memory board user's guide. As shown in the figure, EBLK7 is mapped into every memory map configuration selected by switches 1 to 4. The only difference is the beginning address. For example, with switches 1 to 4 set to OFF-ON-ON-ON, EBLK7 starts at address 2000_{16}. This being the case, the demonstration software can be executed with the following interaction with TIBUG:

```
?R
W=FFC6
P=01A4  2000
?E
```

D.3 DEMONSTRATION SOFTWARE COMMANDS

When the demonstration software is executed, it outputs an opening message that asks for which addresses to check. After these initial inputs, a double question mark prompt (??) is output asking for one of the six commands explained in the following paragraphs.

The opening message is shown below:

```
?E
TM990/201-/206 DEMO SOFTWARE      REV.A      8/02/78

INPUT HEX START ADDRESS, DEFAULT = 2000=>
```

The input hex start address is the starting memory location at which the demonstration software will begin checking. Enter the desired start address (in hexadecimal); if the default address of 2000_{16} is desired, enter only a carriage return. Next a prompt will ask for the address at which demonstration software will end its check routine:

```
INPUT HEX END ADDRESS, DEFAULT = F000=>
```

Enter the desired end address; if the default address of $F000_{16}$ is desired, enter only a carriage return. These start and end address prompts are the same as if the I command was issued (paragraph D.3.2).

If the start address is greater than the end address the following message is output:

```
**ERROR**
```

To correct, re-enter the values in the proper order. After these interactive messages, a double question mark (??) prompt asks for one of the six commands explained below.

NOTE

> Address inputs to the demonstration software prompts should be even hexadecimal values. If an odd value is input, the resulting address will be the odd value minus one (e.g., FE01$_{16}$ will be interpreted as FE00$_{16}$).

D.3.1 HELP COMMAND (H)

To obtain a list of the six one-character commands observed by the demonstration software, enter the H command. The following list will be output:

```
?? H
COMMANDS:

H - HELP, PRINTS THIS HELP LIST
Q - QUIT, BACK TO TIBUG
S - SEARCH FOR RAM BOUNDS
    SEARCHES FOR THE FIRST CONTIGUOUS BLOCK OF RAM
I - INITIALIZE MEMORY BOUNDS
P - PATTERN MEMORY
    WRITE PATTERN TO ALL LOCATIONS UNDER TEST
V - VERIFY RAM OPERATION
    ADDRESSING AND DATA TEST; THE DATA TEST CHECKS EVERY BIT.
    THE ADDRESS TEST CHECKS TO SEE THAT ALL THE ADDRESSES ARE UNIQUE
```

D.3.2 INITIALIZE MEMORY BOUNDS COMMAND (I)

This command sets the memory address bounds. The demonstration software will check memory as defined by start and end memory address bounds. These addresses are first defined in the opening message (paragraph D.3); the I command provides the same function so that these bounds can be modified as desired. Two interactive messages ask for the starting and end address with default values noted. The start address must be less than the end address; if not, the error message **ERROR** will be output as described in paragraph D.3. To accept the default value (2000$_{16}$ start address and F000$_{16}$ end address), enter a carriage return. To change the address, enter the new address followed by a carriage return.

The following example changes the start and end addresses to B800$_{16}$ and C000$_{16}$ respectively.

```
?? I
INPUT HEX START ADDRESS, DEFAULT = 2000=> B800
INPUT HEX END ADDRESS, DEFAULT = F000=> C000
```

D.3.3 SEARCH FOR RAM BOUNDS COMMAND (S)

The software checks memory for a contiguous block of RAM. When the block is found, the beginning and ending addresses of the block are printed out. The data in the RAM area is not disturbed. The area of memory searched will be the bounds set at program initialization or by the I command. Even though several different RAM blocks may be present in the search area, only the first block (lowest address) will be located by this command. After one RAM block is found and a message output, control reverts back to the demonstration software command scanner and the double question mark is printed. If a RAM area is not found, the message:

```
NO MEMORY FOUND
```

will be printed out. If the RAM area exceeds the bounds of memory to be searched, the search will stop at the search-end location and a message will write out the beginning RAM address and the end address of the search as well as a ****TEST COMPLETE**** message. In the following example, a RAM block was found from addresses B800$_{16}$ to C000$_{16}$.

```
?? S
MEMORY UNDER TEST =>B800 TO C000
 ****TEST COMPLETE****
```

Note that the search routine is also part of the verify command (paragraph D.3.5).

D.3.4 WRITE/READ HEXADECIMAL PATTERN TO RAM COMMAND (P)

The command writes and reads a hexadecimal value (pattern) to and from each location found during the opening message or I command (paragraphs D.3 and D.3.2) or RAM address area found during the search routine (paragraphs D.3.3 and D.3.5). The data read from the memory address is compared to the data pattern sent to verify RAM operation. The data in the memory locations will remain changed to the pattern written.

Following an opening message, the user can choose to use the default pattern value of 0000$_{16}$ by entering a carriage return or can designate a four-digit hexadecimal pattern to be used followed by a carriage return. A non-hexadecimal input will result in issuing the message **ERROR** and the beginning pattern prompt reissued. If the pattern read from a memory location is different than the pattern written, an error message is output showing the location in error, this pattern subroutine is terminated, and program control is returned to the demonstration program command scanner. Error message is as follows:

```
**ERROR**LOCATION 2000
```

The following example sends a pattern of AAAA$_{16}$ to RAM without an error occurring. Then the user checked three memory addresses, using TIBUG, showing that AAAA$_{16}$ was written to each.

```
?? P
INPUT HEX PATTERN, DEFAULT = 0000=> AAAA
 ****TEST COMPLETE****

?? Q
?M B800
B800=AAAA
B802=AAAA
B804=AAAA
B806=AAAA
?
```

D.3.5 VERIFY MEMORY COMMAND (V)

There are three parts to the verify subroutine:

a. Search routine. This is the same as the search command explained in paragraph D.3.3.

b. A data check. This is a bit-by-bit check of each RAM memory address. Both a zero and also a one are written to each bit, then read back and checked. If the check showed an error, a message is output. The following example is a message indicating that 02E0$_{16}$ was read back from location 2000$_{16}$ when the value read was expected to be 7FFF$_{16}$.

```
DATA BUS IN ERROR, 7FFF WAS WRITTEN, BUT 02E0 WAS READ BACK
LOCATION 2000
```

c. A memory address check. The address value of each memory address location is written to each corresponding memory address. The value is read back and compared with the actual address. When the comparison shows an error (not equal), a message is written. The following example is a message indicating that the value 2008$_{16}$ was read back from memory location 2000$_{16}$ (2000$_{16}$ should have been read).

```
ADDRESS PROBLEM FOUND. LOCATION 2000 WAS IN ERROR WITH ADDRESS 2008
```

The verify command will conduct its initial RAM block search within the bounds set by the I command or by initial demonstration software bounds prompts. The following example shows that a RAM block was found from A000$_{16}$ to C000$_{16}$, the entire memory under the bounds in the I command. A second block was found at FC00$_{16}$ to FF00$_{16}$ after the bounds were reset to conduct a search from D000$_{16}$ to FF00$_{16}$. Both tests were completed without finding errors.

```
?? I
INPUT HEX START ADDRESS, DEFAULT = A000=> 8000
INPUT HEX END ADDRESS, DEFAULT = C000=> C000

?? V
MEMORY UNDER TEST =>A000 TO C000
ONE MOMENT PLEASE
 ♦♦♦♦TEST COMPLETE♦♦♦♦

?? I
INPUT HEX START ADDRESS, DEFAULT = A000=> D000
INPUT HEX END ADDRESS, DEFAULT = C000=> FF00

?? V
MEMORY UNDER TEST =>FC00 TO FF00
ONE MOMENT PLEASE
 ♦♦♦♦TEST COMPLETE♦♦♦♦

??
```

D.3.6 QUIT, RETURN TO MONITOR COMMAND (Q)

To return to the monitor, enter the character Q. The monitor will respond with its single question-mark prompt.

D.4 DEMONSTRATION SOFTWARE LISTING

A listing of the demonstration software follows.

```
                          IDT  'DEMO'
           *=============================================================
           *
           *     TITLE:        TM990/201-/206 DEMO TEST SOFTWARE
           *
           *     DATE:         11/16/77
           *
           *     FUNCTION:     TO DEMONSTRATE/TEST RAM MEMORY ON THE
           *                   TM990/201 AND TM990/206 EXPANSION
           *                   MEMORY BOARDS
           *
           *     DESCRIPTION:  THIS PROGRAM PROMPTS THE USER FOR RAM
           *                   BOUNDS. THESE BOUNDS SET THE ADDRESS
           *                   LIMITS FOR TESTING.
           *
           *      ***********THIS PROGRAM IS RELOCATABLE***********
           *
           *       THE WORKSPACE FOR THIS MODULE STARTS AT >FF90
           *
           *=============================================================
           *======REGISTER EQUATES======*
     0000  R0     EQU  0
     0001  R1     EQU  1
     0002  R2     EQU  2
     0003  R3     EQU  3
     0004  R4     EQU  4
     0005  R5     EQU  5
     0006  R6     EQU  6
     0007  R7     EQU  7
     0008  R8     EQU  8
     0009  R9     EQU  9
     000A  R10    EQU  10
     000B  R11    EQU  11
     000C  R12    EQU  12
     000D  R13    EQU  13
     000E  R14    EQU  14
     000F  R15    EQU  15
           ***************************
                  DXOP HEXI,9
                  DXOP HEXO,10
                  DXOP READ,11
                  DXOP PRNT,14
           *************************
           *XXXX THIS SECTION ALLOWS RELOCATION XXXX
           START
0000 02E0         LWPI >FF90                LOAD W.P.
0002 FF90
0004 0201         LI   R1,>045B             R1 = RT INSTRUCTION
0006 045B
0008 0681         BL   R1                   LOAD R11 WITH ADD. OF NEXT INS
000A 022B         AI   R11,-10              ADJUST TO START ADD.
000C FFF6
000E C28B         MOV  R11,R10              R10 = BASE ADDRESS
           *XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
 0053 0010 0208          LI   R8,>2000          LOAD DEFAULT START ADD.
      0012 2000
 0054 0014 0209          LI   R9,>F000          LOAD DEFAULT END ADD.
      0016 F000
 0055 0018 0207          LI   R7,>0000          LOAD DEFAULT PATTERN
      001A 0000
 0056 001C 2FAA          PRNT @CRLF(10)
      001E 01F8'
 0057 0020 2FAA          PRNT @BANNER(10)       OUTPUT BANNER/DATE
      0022 02A4'
 0058               *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 0059               INIT
 0060 0024 06AA          BL   @BOUNDS(10)       SET MEMORY BOUNDS
      0026 0497'
 0061               *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 0062               INSTR
 0063 0028 2FAA          PRNT @PROMPT(10)       OUTPUT PROMPT
      002A 02D9'
 0064 002C 2EC5          READ R5                READ THE COMMAND CHARACTER
 0065 002E 0285          CI   R5,>5600          IF IT IS A 'V'=>VERIFY
      0030 5600
 0066 0032 1317          JEQ  VERIFY
 0067 0034 0285          CI   R5,>4900          IF IT IS A 'I'=>INITIALIZE
      0036 4900
 0068 0038 13F5          JEQ  INIT
 0069 003A 0285          CI   R5,>5100          IF IT IS A 'Q'=>QUIT
      003C 5100
 0070 003E 130F          JEQ  QUIT
 0071 0040 0285          CI   R5,>4800          IF IT IS A 'H'=>HELP
      0042 4800
 0072 0044 1309          JEQ  HELP
 0073 0046 0285          CI   R5,>5000          IF IT IS A 'P'=>PATTERN
      0048 5000
 0074 004A 131B          JEQ  PAT
 0075 004C 0285          CI   R5,>5300          IF IT IS A 'S'=>SEARCH
      004E 5300
 0076 0050 1311          JEQ  SEARCH
 0077 0052 2FAA          PRNT @INVCMD(10)       OTHERWISE PRINT ERROR
      0054 0563'
 0078 0056 10E8          JMP  INSTR             GET NEW COMMAND
 0079               *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 0080               HELP
 0081 0058 2FAA          PRNT @HLIST(10)        OUTPUT HELP LIST
      005A 02FE'
 0082 005C 10E5          JMP  INSTR             GET NEW COMMAND
 0083               *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 0084               QUIT
 0085 005E 0460          B    @>0080            EXIT, TO TIBUG !
      0060 0080
 0086               *++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 0087               VERIFY
 0088 0062 2FAA          PRNT @CRLF(10)
      0064 01F8'
 0089 0066 06AA          BL   @SCAN(10)         FIND THE RAM
```

```
      0068 018C'
 0090 006A 06AA          BL   @TEST(10)          RUN FUNCTIONAL TEST
      006C 00F0'
 0091 006E 2FAA          PRNT @MSG5(10)          PRINT TEST COMPLETE
      0070 01FB'
 0092 0072 10DA          JMP  INSTR              GET NEXT INSTRUCTION
 0093               *************************************************************
 0094               SEARCH
 0095 0074 2FAA          PRNT @CRLF(10)
      0076 01F8'
 0096 0078 06AA          BL   @SCAN(10)          ENTER SEARCH
      007A 018C'
 0097 007C 2FAA          PRNT @MSG5(10)          PRINT TEST COMPLETE
      007E 01FB'
 0098 0080 10D3          JMP  INSTR              GET NEXT INSTRUCTION
 0099               *************************************************************
 0100               PAT
 0101 0082 2FAA          PRNT @CRLF(10)
      0084 01F8'
 0102 0086 06AA          BL   @PATERN(10)        WRITE PATTERN TO MEMORY
      0088 0090'
 0103 008A 2FAA          PRNT @MSG5(10)          PRINT TEST COMPLETE
      008C 01FB'
 0104 008E 10CC          JMP  INSTR              GET NEXT INSTRUCTION
 0105               *************************************************************
 0106               PATERN
 0107 0090 0200          LI   R0,>0460           R0 = BRANCH
      0092 0460
 0108 0094 C080          MOV  R0,R2              R2 = BRANCH
 0109 0096 0201          LI   R1,ZIP             <CR><SPACE>OR<-> GO TO ZIP
      0098 00B4'
 0110 009A A04A          A    R10,R1             ADD BASE ADDRESS
 0111 009C 0203          LI   R3,ERROR2          IF ERR BRANCH TO ERROR2
      009E 00D8'
 0112 00A0 A0CA          A    R10,R3             ADD BASE ADDRESS
 0113               OK
 0114 00A2 2FAA          PRNT @PROMT(10)         ASK FOR PATTERN
      00A4 02E0'
 0115 00A6 2E87          HEXO R7                 OUTPUT DEFAULT
 0116 00A8 2FAA          PRNT @TIC(10)
      00AA 0504'
 0117 00AC 2E44          HEXI R4                 PICK UP PATTERN
 0118 00AE FF90          DATA >FF90              POINTS TO R0
 0119 00B0 FF94          DATA >FF94              NON-NUMERIC OR INVALID CHAR
 0120 00B2 C1C4          MOV  R4,R7              R7 = PATTERN
 0121               ZIP
 0122 00B4 2FAA          PRNT @CRLF(10)
      00B6 01F8'
 0123 00B8 C088          MOV  R8,R2              R2 = START ADDRESS
 0124               PAT2
 0125 00BA CC87          MOV  R7,*R2+            MOV PATTERN TO LOCATION
 0126 00BC C112          MOV  *R2,R4             SAVE NEXT LOCATION
 0127 00BE 0547          INV  R7                 INVERT DATA
 0128 00C0 C487          MOV  R7,*R2             DISTURB DATA LINES
 0129 00C2 0642         DECT R2                 POINT TO PREV. LOCATION
 0130 00C4 C172         MOV  *R2+,R5            SAVE DATA
 0131 00C6 C484         MOV  R4,*R2             RESTORE LOCATION
 0132 00C8 0547         INV  R7                 RESTORE PATTERN
 0133 00CA 0642         DECT R2                 RESTORE POINTER
 0134 00CC 81C5         C    R5,R7              COMPARE DATA
 0135 00CE 1607         JNE  ERR                IF NOT INDICATE
 0136 00D0 05C2         INCT R2                 INCREMENT ADDRESS
 0137 00D2 8242         C    R2,R9              DONE?--R9 = END ADDR.
 0138 00D4 1AF2         JL   PAT2               IF NOT CONTINUE
 0139 00D6 100A         JMP  RET                OTHERWISE GET NEXT INSTR.
 0140              ERROR2
 0141 00D8 2FAA         PRNT @ERRMSG(10)        PRINT ERROR MSG.
      00DA 0555'
 0142 00DC 045B         RT
 0143              ERR
 0144 00DE 2FAA         PRNT @ERRMSG(10)        PRINT ERROR
      00E0 0555'
 0145 00E2 2FAA         PRNT @LOC(10)           *
      00E4 0574'
 0146 00E6 2E82         HEXO R2                 PRINT ADDRESS
 0147 00E8 2FAA         PRNT @CRLF(10)          CLEAR LINE
      00EA 01F8'
 0148              RET
 0149 00EC 045B         RT
 0150 00EE 1000         NOP
 0151              *==============================================================
 0152              *
 0153              *  CHECKS DATA BIT FAILURES
 0154              *
 0155              *  THIS TEST WRITES TO EVERY INDIVIDUAL BIT IN THE
 0156              * MEMORY UNDER TEST TWICE, TO VERIFY THAT THE DATA
 0157              * BUS IS FUNCTIONAL.
 0158              *==============================================================
 0159              TEST
 0160 00F0 2FAA         PRNT @WAIT(10)          OUTPUT WAIT MESSAGE
      00F2 014E'
 0161 00F4 04C6         CLR  R6                 CLR LOOP FLAG
 0162 00F6 0203         LI   R3,>FFFE           LOAD DATA MASK
      00F8 FFFE
 0163              TOP
 0164 00FA C088         MOV  R8,R2              LOAD START ADDRESS
 0165              TOP2
 0166 00FC 0204         LI   R4,>10             SET COUNT TO 16
      00FE 0010
 0167              LOOP1
 0168 0100 0604         DEC  R4                 DECREMENT COUNT
 0169 0102 C104         MOV  R4,R4              CHECK FOR END OF LOOP
 0170 0104 1601         JNE  CONT               IF NOT DONE CONTINUE !
 0171 0106 1019         JMP  EOTCK              ***CHECK TO SEE IF DONE
 0172 0108 0B13    CONT SRC  R3,R1              SHIFT MASK TO NEXT BIT
 0173 010A CC83         MOV  R3,*R2+            WRITE MASK TO MEMORY
 0174 010C 0543         INV  R3                 INVERT DATA
 0175 010E C012         MOV  *R2,R0             SAVE FOR RESTORE
 0176 0110 C483          MOV  R3,*R2          WRITE DISTURB DATA
 0177 0112 0543          INV  R3              RESET DATA
 0178 0114 C480          MOV  R0,*R2          RESTORE DATA
 0179 0116 0642          DECT R2              RESET ADDRESS
 0180 0118 C152          MOV  *R2,R5          READ IT BACK
 0181 011A 80C5          C    R5,R3           COMPARE
 0182 011C 13F1          JEQ  LOOP1           IF CORRECT DO NEXT BIT
 0183 011E 2FAA          PRNT @MSG6(10)       IF NOT, OUTPUT FAIL MSG.
      0120 0214'
 0184 0122 2E83          HEXO R3              *
 0185 0124 2FAA          PRNT @MSG7(10)       *
      0126 0228'
 0186 0128 2E85          HEXO R5              *
 0187 012A 2FAA          PRNT @MSG2(10)       *
      012C 01EA'
 0188 012E 2FAA          PRNT @LOC(10)        *
      0130 0574'
 0189 0132 2E82          HEXO R2              *
 0190 0134 2FAA          PRNT @CRLF(10)       *
      0136 01F8'
 0191 0138 1028          JMP  EOT             AND EXIT TEST
 0192               EOTCK
 0193 013A 05C2          INCT R2              INCREMENT ADDRESS
 0194 013C 8242          C    R2,R9           IF IT IS LESS THAN END ADD.
 0195 013E 1ADE          JL   TOP2            JUMP BACK AND FINISH
 0196 0140 C186          MOV  R6,R6           GO ON ?
 0197 0142 1301          JEQ  BYPAS4          YES IF NO FAIL OCCURRED
 0198 0144 100E          JMP  TEST2           OTHERWISE GO TO TEST 2
 0199               BYPAS4
 0200 0146 0203          LI   R3,>0001        LOAD NEW MASK
      0148 0001
 0201 014A 0706          SETO R6              SET FLAG WHEN 1ST PAT DONE
 0202 014C 10D6          JMP  TOP
 0203 014E   4F     WAIT TEXT 'ONE MOMENT PLEASE'
 0204 015F   0D          BYTE >D,>A,>0
      0160   0A
      0161   00
 0205                    EVEN
 0206               *-----------------------------------------------------------
 0207               *
 0208               * CHECKS ADDRESSING FAILURES
 0209               *
 0210               *  THIS TEST WRITES THE ADDRES OF THE LOCATION
 0211               * TO THE LOCATION TO VERIFY THAT EACH ADDRESS
 0212               * IN THE AREA UNDER TEST IS UNIQUE.
 0213               *-----------------------------------------------------------
 0214               TEST2
 0215 0162 C088          MOV  R8,R2           LOAD STARTING ADDRESS
 0216               WRITE
 0217 0164 CC82          MOV  R2,*R2+         WRITE TO LOCATION
 0218 0166 8242          C    R2,R9           DONE ?
 0219 0168 1AFD          JL   WRITE           IF NOT JUMP BACK
 0220 016A C088          MOV  R8,R2           OTHERWISE READ IT BACK
 0221               READA
 0222 016C C152           MOV  *R2,R5              STORE
 0223 016E 8C82           C    R2,*R2+             AND COMPARE
 0224 0170 1603           JNE  ERR3                IF NOT =, INDICATE ERROR
 0225 0172 8242           C    R2,R9               DONE ?
 0226 0174 1AFB           JL   READA               IF NOT CONTINUE
 0227 0176 1009           JMP  EOT                 OTHERWISE END THE TEST
 0228               ERR3
 0229 0178 0642           DECT R2
 0230 017A 2FAA           PRNT @MSG9(10)           ERROR MESSAGE
      017C 023B'
 0231 017E 2E82           HEXO R2                  *
 0232 0180 2FAA           PRNT @MSG10(10)          *
      0182 025C'
 0233 0184 2E85           HEXO R5                  *
 0234 0186 2FAA           PRNT @CRLF(10)           *
      0188 01F8'
 0235               EOT
 0236 018A 045B           RT                       GET NEXT COMMAND
 0237               *##############################################################
 0238               *
 0239               *  DEFINES START AND STOP ADDRESSES
 0240               *
 0241               *  THIS ROUTINE SEARCHES BETWEEN START AND END
 0242               * FOR THE FIRST CONTIGUOUS BLOCK OF RAM MEMORY.
 0243               *
 0244               *   THIS ROUTINE DOES A NON-DESTRUCTIVE SEARCH
 0245               *##############################################################
 0246               SCAN
 0247 018C 0704           SETO R4                  DISTURB DATA = >FFFF
 0248 018E 04C3           CLR  R3                  DATA = 0
 0249 0190 C088           MOV  R8,R2               GET START ADDRESS
 0250 0192 0642           DECT R2                  ADJUST FOR TEST
 0251 0194 04C6           CLR  R6                  CLR FIRST FLAG
 0252               FIND
 0253 0196 8242           C    R2,R9               LAST ADD. ?
 0254 0198 1323           JEQ  TOOFAR              IF SO GO TO ERR. ROUTINE
 0255 019A 05C2           INCT R2
 0256 019C C012           MOV  *R2,R0              SAVE FOR RESTORE
 0257 019E CC83           MOV  R3,*R2+             WRITE TO MEMORY
 0258 01A0 C052           MOV  *R2,R1              SAVE FOR RESTORE
 0259 01A2 C484           MOV  R4,*R2              WRITE DISTURB DATA
 0260 01A4 0642           DECT R2                  MOV POINTER BACK
 0261 01A6 C152           MOV  *R2,R5              STORE WRITTEN DATA
 0262 01A8 8143           C    R3,R5               IS DATA = WHAT WAS WRITTEN ?
 0263 01AA 1316           JEQ  MOVE                IF SO CHECK IF FIRST TIME
 0264               FIND2
 0265 01AC C186           MOV  R6,R6               IF FLAG = 0,STRT ADD NOT FOUND
 0266 01AE 1310           JEQ   GO                 SO TRY TO FIND IT
 0267               SUPER
 0268 01B0 C242           MOV  R2,R9               OTHERWISE THIS IS THE ENDADD
 0269 01B2 CC80           MOV  R0,*R2+             RESTORE DATA
 0270 01B4 C481           MOV  R1,*R2              RESTORE DATA
 0271 01B6 2FAA           PRNT @MBOUND(10)
      01B8 028A'
 0272 01BA 2E88           HEXO R8                  OUTPUT MEMORY BOUNDS
 0273 01BC 2FAA           PRNT @TO(10)
      01BE 029F'
 0274 01C0 2E89           HEXO R9
 0275 01C2 2FAA           PRNT @CRLF(10)
      01C4 01F8'
 0276 01C6 1010           JMP  EOSRCH              AND JUMP TO END OF TEST
 0277              STRTAD
 0278 01C8 C186           MOV  R6,R6               CHECK FIRST FLAG
 0279 01CA 1602           JNE  GO                  IF R6 IS SET START ALREADY FOU
 0280 01CC C202           MOV  R2,R8               OTHERWISE THIS IS START ADD.
 0281 01CE 0706           SETO R6                  SET FLAG
 0282              GO
 0283 01D0 CC80           MOV  R0,*R2+             RESTORE DATA
 0284 01D2 C481           MOV  R1,*R2              RESTORE DATA
 0285 01D4 0642           DECT R2                  RESET FOR LOOP
 0286 01D6 10DF           JMP  FIND                CONTINUE
 0287              MOVE
 0288 01D8 C484           MOV  R4,*R2              CHECK TO SEE IF ITS EPROM
 0289 01DA 8112           C    *R2,R4
 0290 01DC 13F5           JEQ  STRTAD              IF EQ, CHECK FOR START ADD.
 0291 01DE 10E6           JMP FIND2                NOW TO FIND END ADD.
 0292              TOOFAR
 0293 01E0 C186           MOV  R6,R6               IF SET >F000 MAY BE THE END AD
 0294 01E2 16E6           JNE  SUPER               IF SO, THAT'S SUPER
 0295 01E4 2FAA           PRNT @MSG16(10)          OUTPUT 'NO RAM MEMORY FOUND'
      01E6 0278'
 0296              EOSRCH
 0297 01E8 045B           RT                       RETURN
 0298              *
 0299              ************************************************************
 0300              *SAGES  ***  MESSAGES  ***  MESSAGES  ***  MESSAGES  ***  ME
 0301              ************************************************************
 0302 01EA   20    MSG2   TEXT ' WAS READ BACK'
 0303 01F8   0D    CRLF   BYTE >D,>A,>0
      01F9   0A
      01FA   00
 0304 01FB   20    MSG5   TEXT ' ****TEST COMPLETE****'
 0305 0211   0D           BYTE >D,>A,>0
      0212   0A
      0213   00
 0306 0214   44    MSG6   TEXT 'DATA BUS IN ERROR, '
 0307 0227   00           BYTE >0
 0308 0228   20    MSG7   TEXT ' WAS WRITTEN, BUT '
 0309 023A   00           BYTE >0
 0310 023B   41    MSG9   TEXT 'ADDRESS PROBLEM FOUND. LOCATION '
 0311 025B   00           BYTE >0
 0312 025C   20    MSG10  TEXT ' WAS IN ERROR WITH ADDRESS '
 0313 0277   00           BYTE >0
 0314 0278   4E    MSG16  TEXT 'NO MEMORY FOUND'
 0315 0287   0D           BYTE >D,>A,>0
      0288   0A
      0289   00
 0316 028A   4D    MBOUND TEXT 'MEMORY UNDER TEST =>'
```

```
  0317 029E   00           BYTE >0
  0318 029F   20   TO      TEXT ' TO '
  0319 02A3   00           BYTE >0
  0320 02A4   54   BANNER  TEXT 'TM990/201-/206 DEMO SOFTWARE      '
  0321 02C5   52           TEXT 'REV.A      8/02/78'
  0322 02D6   0D           BYTE >D,>A,>0
       02D7   0A
       02D8   00
  0323 02D9   0D   PROMPT  BYTE >D,>A
       02DA   0A
  0324 02DB   3F           TEXT '?? '
  0325 02DE   07           BYTE >7,>0
       02DF   00
  0326 02E0   49   PROMT   TEXT 'INPUT HEX PATTERN, DEFAULT = '
  0327 02FD   00           BYTE >0
  0328 02FE   0D   HLIST   BYTE >D,>A
       02FF   0A
  0329 0300   43           TEXT 'COMMANDS:'
  0330 0309   0D           BYTE >D,>A,>A
       030A   0A
       030B   0A
  0331 030C   48           TEXT 'H - HELP, PRINTS THIS HELP LIST'
  0332 032B   0D           BYTE >D,>A
       032C   0A
  0333 032D   51           TEXT 'Q - QUIT, BACK TO TIBUG'
  0334 0344   0D           BYTE >D,>A
       0345   0A
  0335 0346   53           TEXT 'S - SEARCH FOR RAM BOUNDS'
  0336 035F   0D           BYTE >D,>A
       0360   0A
  0337 0361   20           TEXT '     SEARCHES FOR THE FIRST CONTIGUOUS BLOCK'
  0338 038C   20           TEXT ' OF RAM '
  0339 0394   0D           BYTE >D,>A
       0395   0A
  0340 0396   49           TEXT 'I - INITIALIZE MEMORY BOUNDS'
  0341 03B2   0D           BYTE >D,>A
       03B3   0A
  0342 03B4   50           TEXT 'P - PATTERN MEMORY'
  0343 03C6   0D           BYTE >D,>A
       03C7   0A
  0344 03C8   20           TEXT '     WRITE PATTERN TO ALL LOCATIONS UNDER TEST
  0345 03F5   0D           BYTE >D,>A
       03F6   0A
  0346 03F7   56           TEXT 'V - VERIFY RAM OPERATION'
  0347 040F   0D           BYTE >D,>A
       0410   0A
  0348 0411   20           TEXT '     ADDRESSING AND DATA TEST;'
  0349 042E   20           TEXT ' THE DATA TEST CHECKS EVERY BIT.'
  0350 044E   0D           BYTE >D,>A
       044F   0A
  0351 0450   20           TEXT '     THE ADDRESS TEST CHECKS TO SEE THAT ALL'
  0352 047B   20           TEXT ' THE ADDRESSES ARE UNIQUE'
  0353 0494   0D           BYTE >D,>A,>0
       0495   0A
```

```
      0496   00
 0354                 *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 0355                 * THIS SUBROUTINE ACCEPTS HEX CHARACTERS FROM THE
 0356                 * TERMINAL. R8 IS USED FOR THE BEGINNING ADDRESS AND
 0357                 * R9 IS USED FOR THE END ADDRESS
 0358                 *
 0359                 *     XOP HEXI REQUIRES TWO ARGUEMENTS (ADDRESSES) TO
 0360                 *     FOLLOW IT. THE FIRST IS THE ADDRESS IT WILL
 0361                 *     BRANCH TO IF A CARRIAGE RETURN, A SPACE BAR,OR
 0362                 *     A MINUS SIGN IS ENTERED.
 0363                 *     THE SECOND IS THE ADDRESS IT WILL BRANCH TO
 0364                 *     IF AN INVALID HEX DIGIT IS ENTERED.
 0365                 *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 0366                 BOUNDS
 0367 0498 0200             LI   R0,>0460          R0 = BRANCH
      049A 0460
 0368 049C 0201             LI   R1,ZIP1           IF <CR> BRANCH TO OK1
      049E 04D0'
 0369 04A0 A04A             A    R10,R1            ADD BASE ADDRESS
 0370 04A2 0202             LI   R2,>0460          R2 = BRANCH
      04A4 0460
 0371 04A6 0203             LI   R3,ERROR1         IF ERR BRANCH TO ERROR1
      04A8 0509'
 0372 04AA A0CA             A    R10,R3            ADD BASE ADDRESS
 0373 04AC 2FAA             PRNT @CRLF(10)
      04AE 01F8'
 0374 04B0 2FAA             PRNT @ENTRY1(10)       PRINT PROMPT 1
      04B2 0510'
 0375 04B4 2E88             HEXO R8                OUTPUT DEFAULT
 0376 04B6 2FAA             PRNT @TIC(10)
      04B8 0504'
 0377 04BA 2E44             HEXI R4                WAIT FOR START ADDRESS
 0378 04BC FF90             DATA >FF90             POINTS TO R0
 0379 04BE FF94             DATA >FF94             NON-NUMERIC OR INVALID CHAR
 0380 04C0 0284             CI   R4,>FF90          INFRINGE ON WP ?
      04C2 FF90
 0381 04C4 1202             JLE  BYPAS             IF NOT GO ON
 0382 04C6 0204             LI   R4,>FF90          IF SO ENTER >FF90
      04C8 FF90
 0383                 BYPAS
 0384 04CA C204             MOV  R4,R8             R8 = START ADDRESS
 0385 04CC 0248             ANDI R8,>FFFE          MAKE IT EVEN
      04CE FFFE
 0386                 ZIP1
 0387 04D0 0201             LI   R1,ZIP2           IF <CR> BRANCH TO OK2
      04D2 04FA'
 0388 04D4 A04A             A    R10,R1            ADD BASE ADDRESS
 0389                 OK2
 0390 04D6 2FAA             PRNT @CRLF(10)
      04D8 01F8'
 0391 04DA 2FAA             PRNT @ENTRY2(10)       PRINT PROMPT 2
      04DC 0534'
 0392 04DE 2E89             HEXO R9                OUTPUT DEFAULT
 0393 04E0 2FAA             PRNT @TIC(10)
```

```
      04E2 0504'
 0394 04E4 2E44          HEXI R4                WAIT FOR END ADDRESS
 0395 04E6 FF90          DATA >FF90             POINTS TO R0
 0396 04E8 FF94          DATA >FF94             NON-NUMERIC OR INVALID CHAR
 0397 04EA 0284          CI   R4,>FF90          INVADE WP ?
      04EC FF90
 0398 04EE 1202          JLE  BYPASS            IF NOT GO ON
 0399 04F0 0204          LI   R4,>FF90          OTHERWISE ENTER >FF90
      04F2 FF90
 0400           BYPASS
 0401 04F4 C244          MOV  R4,R9             R9 = END ADDRESS
 0402 04F6 0249          ANDI R9,>FFFE          MAKE IT EVEN
      04F8 FFFE
 0403           ZIP2
 0404 04FA 2FAA          PRNT @CRLF(10)
      04FC 01F8'
 0405 04FE 8248          C    R8,R9             ARE ENTRIES IN ORDER ?
 0406 0500 1B03          JH   ERROR1            IF ERR PRINT ERR MSG.
 0407 0502 045B          RT                     RETURN
 0408           TIC
 0409 0504   3D          TEXT '=> '
 0410 0507   07          BYTE >7,>0
      0508   00
 0411           ERROR1
 0412 0509 2FAA          PRNT @ERRMSG(10)       PRINT ERROR MSG.
      050B 0555'
 0413 050E 10C3          JMP  BOUNDS            AND JUMP BACK
 0414           ENTRY1
 0415 0510   49          TEXT 'INPUT HEX START ADDRESS, DEFAULT = '
 0416 0533   00          BYTE >0
 0417           ENTRY2
 0418 0534   49          TEXT 'INPUT HEX END.ADDRES, DEFAULT = '
 0419 0554   00          BYTE >0
 0420           ERRMSG
 0421 0555   0D          BYTE >D,>A
      0556   0A
 0422 0557   2A          TEXT '***ERROR***'
 0423 0562   00          BYTE >0
 0424           INVCMD
 0425 0563   20          TEXT ' INVALID COMMAND'
 0426 0573   00          BYTE >0
 0427           LOC
 0428 0574   4C          TEXT 'LOCATION '
 0429 057D   00          BYTE >0
 0430           *:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 0431                    END  START

 0000 ERRORS
```