As you are now the owner of this document which should have come to you for free, please consider making a donation of £1 or more for the upkeep of the (Radar) website which holds this document. I give my time for free, but it costs me money to bring this document to you. You can donate here https://blunham.com/Misc/Texas

Many thanks.

Please do not upload this copyright pdf document to any other website. Breach of copyright may result in a criminal conviction.

This Acrobat document was generated by me, Colin Hinson, from a document held by me. I requested permission to publish this from Texas Instruments (twice) but received no reply. It is presented here (for free) and this pdf version of the document is my copyright in much the same way as a photograph would be. If you believe the document to be under other copyright, please contact me.

The document should have been downloaded from my website https://blunham.com/, or any mirror site named on that site. If you downloaded it from elsewhere, please let me know (particularly if you were charged for it). You can contact me via my Genuki email page: https://www.genuki.org.uk/big/eng/YKS/various?recipient=colin

You may not copy the file for onward transmission of the data nor attempt to make monetary gain by the use of these files. If you want someone else to have a copy of the file, point them at the website. (https://blunham.com/Misc/Texas). **Please do not point them at the file itself as it may move or the site may be updated.**

It should be noted that most of the pages are identifiable as having been processed by me.

I put a lot of time into producing these files which is why you are met with this page when you open the file.

If you find missing pages, pages in the wrong order, anything else wrong with the file or simply want to make a comment, please drop me a line (see above).

It is my hope that you find the file of use to you.

Colin Hinson
In the village of Blunham, Bedfordshire.

TEXAS INSTRUMENTS

TM 990

TM 990/203 Dynamic RAM Expansion Memory Module

MICROPROCESSOR SERIES™

June 1980

Colin Hunson

TABLE OF CONTENTS

SECTION	TITLE	PAGE
1.	INTRODUCTION	
	1.1 General	1-1
	1.2 Manual Organization	1-1
	1.3 Specifications	1-3
	1.4 Applicable Documents	1-2
2.	INSTALLATION AND OPERATION	
	2.1 General	2-1
	2.2 Required Equipment	2-1
	2.3 Unpacking	2-1
	2.4 Initial Setup	2-1
	2.4.1 Module Installation and Hookup	2-1
	2.4.2 Memory Mapping	2-4
	2.4.2.1 Memory Configuration Example	2-7
	2.4.2.2 Memory Configuration Example: Extended Addressing	2-10
	2.4.3 Jumper Positions and Options	2-10
	2.4.3.1 Select Wait State	2-17
	2.4.3.2 Cycle Steal and Transparent Refresh	2-17
	2.4.3.3 Bank Select Circuitry	2-18
	2.4.3.4 Address Bus Jumpers	2-18
	2.5 Operation	2-18
	2.5.1 Operation Without Parity	2-18
	2.5.2 Operation With Parity	2-19
	2.5.3 General	2-19
3.	THEORY OF OPERATION	
	3.1 General	3-1
	3.1.1 RAM Area	3-1
	3.1.2 Parity Logic	3-1
	3.1.3 Memory Controller	3-1
	3.1.4 Data Buffers	3-1
	3.1.5 Data Multiplexer and Latch	3-1
	3.1.6 Wait State Logic	3-3
	3.1.7 Address and Select Logic	3-3
	3.1.8 Memory Refresh	3-3
	3.2 Memory Cycle Flow Chart	3-4
	3.3 Refresh	3-6
	3.3.1 Block Refresh	3-6
	3.3.2 Cycle Stealing Refresh	3-6
	3.3.3 Transparent Refresh	3-7
	3.3.4 Direct Memory Access (DMA)	3-7
	3.4 Circuit Descriptions	3-8
	3.4.1 Controller Section (Schematic Sheet 4)	3-8
	3.4.2 Parity Logic (Schematic Sheet 3)	3-10
	3.4.3 Data Multiplexing and Buffering	3-12
	3.4.4 Addressing and Select Logic (Schematic Sheet 4)	3-14
	3.4.5 Read/Write Timing	3-18

APPENDICES

A	TM 990/203 Jumper Descriptions
B	Schematics
C	CRU Parity Read/Reset User Option
D	Utilizing the Parity option
E	Parts List
F	TM 990 Memory Mapping

LIST OF ILLUSTRATIONS

FIGURE	TITLE	PAGE
1-1	Module Dimensions (In Inches)	1-2
2-1	Card Cage Terminal Block Connections	2-2
2-2	TM 990/510 Card Cage Backplane Schematic	2-4
2-3	Memory Mapping Switch Array	2-5
2-4	Memory Map for TM 990/101M (2K X 16 EPROM, 2K X 16 RAM)	2-8
2-5	Desired System Memory Map (For Text Example)	2-8
2-6	Location of Jumper Pins on Left Third of Board	2-13
2-7	Location of Jumper Pins on Middle Third of Board	2-14
2-8	Location of Jumper Pins on Right Third of Board	2-15
2-9	TM 990/203 Jumper Pin and Memory Bank Locations	2-16
2-10	Memory Configuration Switches	3-16
3-1	TM 990/203 Memory Board Block Diagram	3-2
3-2	Memory Cycle Flow Chart	3-5
3-3	Major State-Controller Components	3-9
3-4	Parity Logic	3-11
3-5	Data Multiplexing and Buffering	3-13
3-6	Memory Configuration Switches	3-16
3-7	Memory Select Scheme for the TM 990/203	3-17
3-8	Address Logic	3-19
3-9	System Timing Showing RAS and CAS Delay	3-20
3-10	Memory Write Operation Timing Diagram	3-21
3-11	Memory Read Operation Timing Diagram	3-22

LIST OF TABLES

TABLE	TITLE	PAGE
1-1	Product Matrix	1-1
1-2	Power Requirements	1-3
2-1	Backplane/P1 Pin Assignments Used by TM 990/203 Module	2-3
2-2	Switch Code Configurable Addresses	2-6
2-3	Switch Code Configurable Addresses (Extended Addressing)	2-7
2-4	Jumper Positions as Shipped	2-9
2-5	Jumper Connection Descriptions	2-9
2-6	Wait States	2-10
2-7	Wait State Jumper Connections	2-17
2-8	Comparison of Cycle Steal and Transparent Refresh Modes	2-18
2-9	Cycle Steal/Transparent Refresh Jumper Connections	2-19
3-1	Memory Address Generation	3-18

SECTION 1

INTRODUCTION

1.1 GENERAL

The Texas Instruments TM 990/203 is a memory expansion module (shown in Figure 1-1) for use with the TM 990 bus-compatible microcomputers. Its features include:

- Compatible with the TM 990 microcomputer bus and system specification.
- Designed to be used with the TM 990/510, /520 or /530 card cages.
- Up to 32K words of dynamic random-access memory.
- Issues an interrupt and lights an indicator upon a parity error.
- DMA compatibility.
- TTL compatible interface.
- Transparent or cycle-steal refresh modes.
- 16 or 20-bit address bus

The TM 990/203 is available in four versions as shown in Table 1-1. Access to the module is through the edge connector which mates to the backplane of either the TM 990/510, /520 or /530 OEM card cages. The TM 990/203 is not compatible with the TM 990/180M module which operates with an 8-bit data bus.

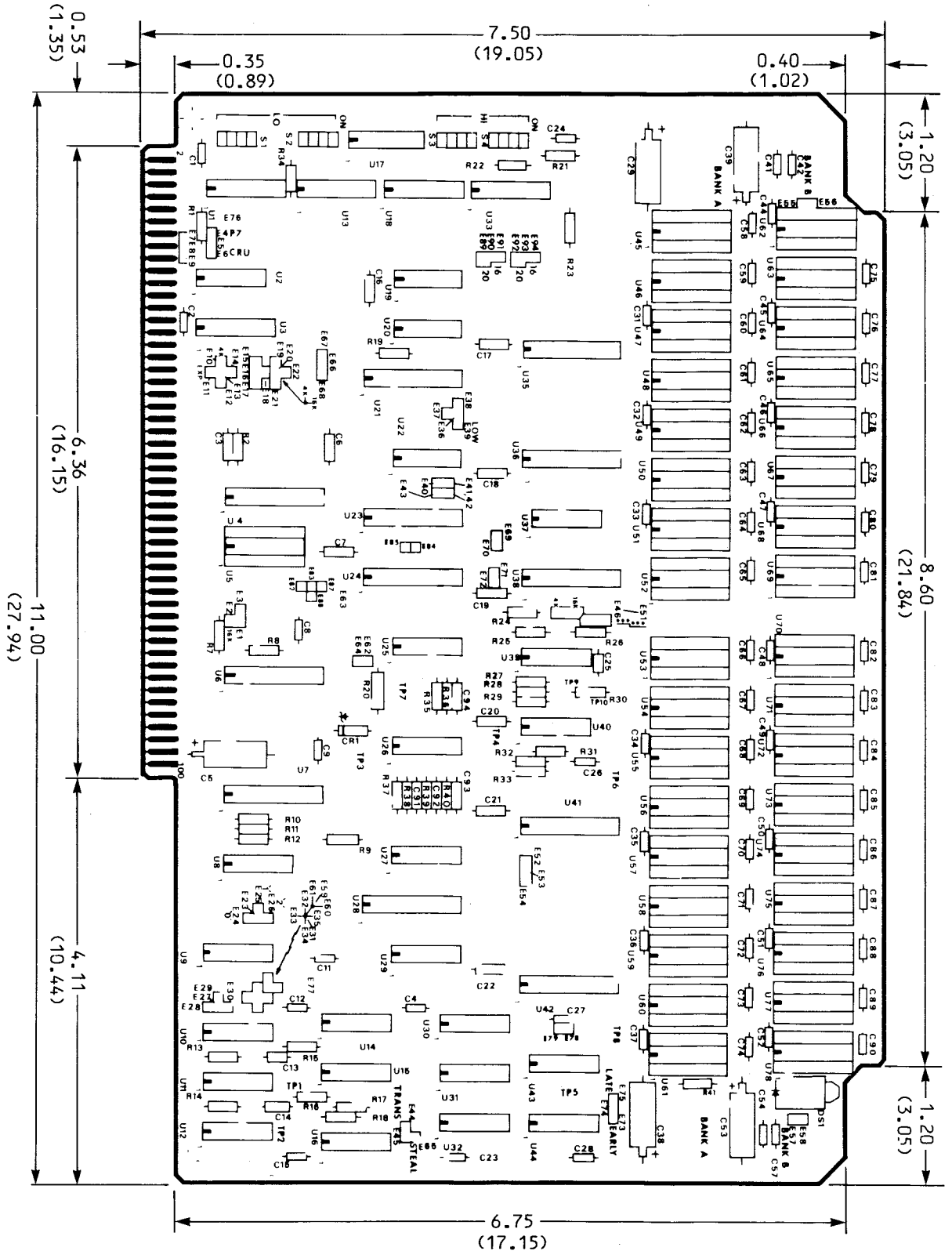
TABLE 1-1. PRODUCT MATRIX

Model	Description
TM 990/203-21	8K X 16 bits of TMS 4027 RAM with parity
TM 990/203-21A	8K X 16 bits of TMS 4108 RAM with parity
TM 990/203-22	16K X 16 bits of TMS 4116 RAM with parity
TM 990/203-23	32K X 16 bits of TMS 4116 RAM with parity

1.2 MANUAL ORGANIZATION

This manual is organized as follows:

- Section 1 provides a description of the TM 990/203, its features, dimensions, and specifications.
- Section 2 describes the correct procedure for installation, power up, and operation of the TM 990/203.
- Section 3 covers the theory of operation, including timing considerations and addressing.



NOTES: DIMENSIONS IN INCHES AND CENTIMETERS (CM IN PARENTHESES)

FIGURE 1-1 MODULE DIMENSIONS (IN INCHES)

1.3 SPECIFICATIONS

Module Dimensions: See Figure 1-2.

Operating Temperature: 0° C to 70° C.

Clock Rate: The TM 990/203 is designed to operate with clock speeds up to 6 MHz.

Devices Utilized: Either TMS 4027 dynamic RAM (4K X 1 bit each) or TMS 4108 dynamic RAM (8K X 1-bit each) in TM 990/203-21. TMS 4116 dynamic RAM (16K X 1 bit each) in TM 990/203-22 and TM 990/203-23.

Power Requirements: See Table 1-2.

TABLE 1-2. POWER REQUIREMENTS

Voltages	Currents (Cycle Steal)					
	8K		16K		32K	
	TYP	MAX	TYP	MAX	TYP	MAX
VDD +12 V +/-5 %	77 mA	1.2 A	65 mA	0.6 A	90 mA	1.2 A
VCC +5 V +/-5 %	1.9 A	3.0 A	1.85 A	2.7 A	1.9 A	3.0 A
-12 V +/-5 %	10 mA	20 mA	10 mA	15 mA	10 mA	20 mA
Voltages	**Currents (Transparent Mode)**					
	8K		16K		32K	
	TYP	MAX	TYP	MAX	TYP	MAX
VDD +12 V +/-5 %	150 mA	1.2 A	110 mA	0.6 A	190 mA	1.2 A
VCC +5 V +/-5 %	1.9 A	3.0 A	1.85 A	2.7 A	1.9 A	3.0 A
-12 V +/-5 %	10 mA	20 mA	10 mA	15 mA	10 mA	20 mA

******* NOTE *******

VBB (-5 V) to the memory devices is derived from the -12 V supply on the backplane. At no time should VBB to the devices exceed +0.3 V from ground. To ensure that this does not occur, the -12 V supply cannot exceed +0.6 V. Failure to observe this precaution will cause dissipation in excess of the maximum ratings due to internal forward bias conditions set up in the memory devices. Permanent damage to these devices will result. When a system is constructed with separate supplies, it is recommended that the -12 V supply be switched on first and off last.

After power-up, eight memory cycles must be performed to achieve proper device operation.

1.4 APPLICABLE DOCUMENTS

- TM 990/100M Microcomputer User's Guide
- TM 990/101M Microcomputer User's Guide
- TMS 9900 Microprocessor Data Manual
- TM 990 System Specification Manual

SECTION 2

INSTALLATION AND OPERATION

2.1 GENERAL

This section explains the procedure for unpacking and setting up the TM 990/203 module for operation with a TM 990/100M or TM 990/101M microcomputer.

2.2 REQUIRED EQUIPMENT

The following equipment would be required for a typical system:

- TM 990/510, /520 or /530 card cage.
- Power supply that is capable of supplying the power requirements of the TM 990/203 memory module (Table 1-2), CPU module, and other user installed equipment.
- Suitable terminal.
- TM 990 bus-compatible microcomputer.

2.3 UNPACKING

Remove the TM 990/203 module from its carton. Check the module for any abnormalities that could have occurred in shipping, and notify your supplier of any discrepancies.

2.4 INITIAL SETUP

The initial setup procedure for the TM 990/203 module involves the following:

- Module installation and hookup.
- Selection of the memory map.
- Selection of jumper options.
- Configure parity option.

2.4.1 Module Installation and Hookup

The use of either a TM 990/510, /520 or /530 card cage is recommended because it offers protection from the abuse that a loose module would receive and simplifies system hookup. The card cage provides termination resistors for the open collector signals used on the bus, provides a terminal block for ease of power supply connections, and in general allows system flexibility. All communications with the TM 990/203 will take place through the module's edge connector which mates to the backplane of the card cage.

If either the TM 990/510, /520 or /530 card cage is used, the hookup is simple. Place the microcomputer in slot 1 of the card cage and place the TM 990/203 in any of the remaining slots. This positions the memory module

between the CPU and the termination resistors on the backplane. Power supply connections can be made to the terminal block on the back of the card cage: Figure 2-1 shows the terminal block and identifies the connections.

CAUTIONS

1. If separate supplies are used, the -12 V supply should not exceed +0.6 V relative to ground. Failure to observe this precaution will cause dissipation in excess of the absolute maximum ratings due to internal forward bias conditions. After powerup, eight memory cycles must be performed to achieve proper device operation.

2. Always remove and insert modules with the power off. Do not remove or insert any module with power applied as significant damage may result.

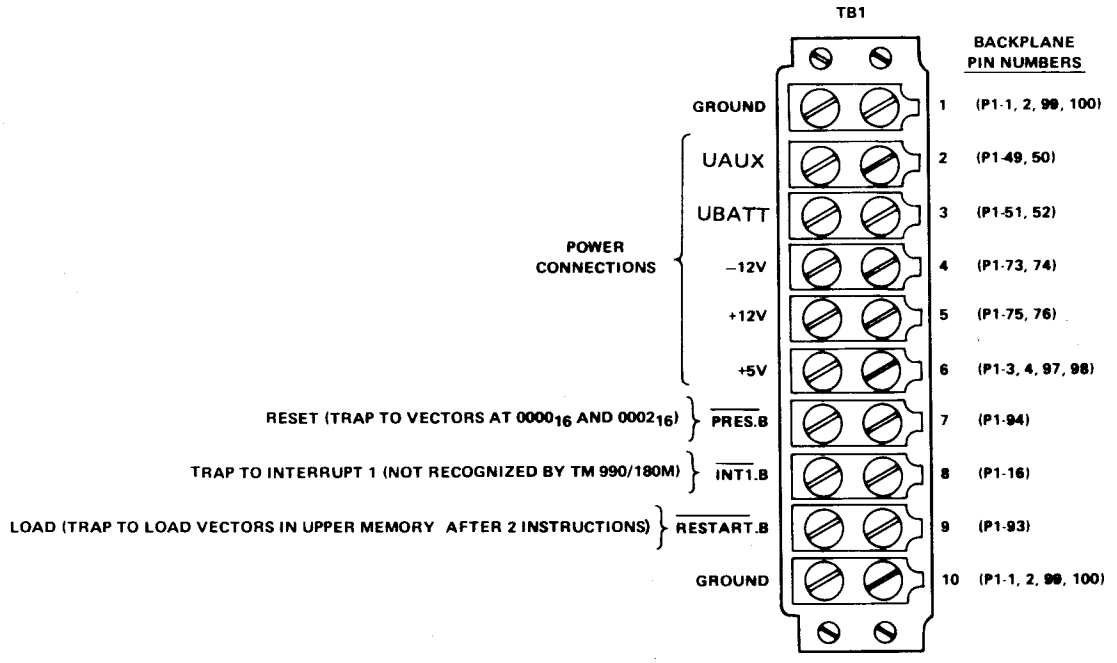


FIGURE 2-1. CARD CAGE TERMINAL BLOCK CONNECTIONS

If the memory module is to be used without a card cage, the signals normally supplied via the card cage backplane must be supplied by a suitable interface cable. The information necessary to design the interface cable is given in Table 2-1 and Figure 2-2. Table 2-1 lists the backplane/P1 pin assignments used by the TM 990/203 and Figure 2-2 is the schematic diagram of the TM 990/510 card cage backplane.

TABLE 2-1. BACKPLANE/P1 PIN ASSIGNMENTS USED BY TM 990/203 MODULE

Pin	Signal	Pin	Signal
1	GND	60	A3.B
2	GND	61	A4.B
3	+5	62	A5.B
4	+5	63	A6.B
11	INT14.B-/(P8 of TMS 9901)	64	A7.B
14	INT15.B-/(P7 of TMS 9901)	65	A8.B
21	GND	66	A9.B
22	BUSCLK.B-	67	A10.B
23	GND	68	A11.B
25	GND	69	A12.B
27	GND	70	A13.B
29	CRUIN.B	71	A14.B
30	CRUOUT.B	73	-12V
31	GND	74	-12V
33	D0.B	75	+12V
34	D1.B	76	+12V
35	D2.B	77	GND
36	D3.B	78	WE.B
37	D4.B	79	GND
38	D5.B	80	MEMEN.B
39	D6.B	81	GND
40	D7.B	82	DBIN.B
41	D8.B	83	GND
42	D9.B	84	MEMCYC.B
43	D10.B	85	GND
44	D11.B	87	CRUCLK.B
45	D12.B	89	GND
46	D13.B	90	READY.B
47	D14.B	91	GND
48	D15.B	95	GRANTOUT.B
53	XA0.B	96	GRANTIN.B
54	XA1.B	97	+5
55	XA2.B	98	+5
56	XA3.B	99	GND
57	A0.B	100	GND
58	A1.B		
59	A2.B		

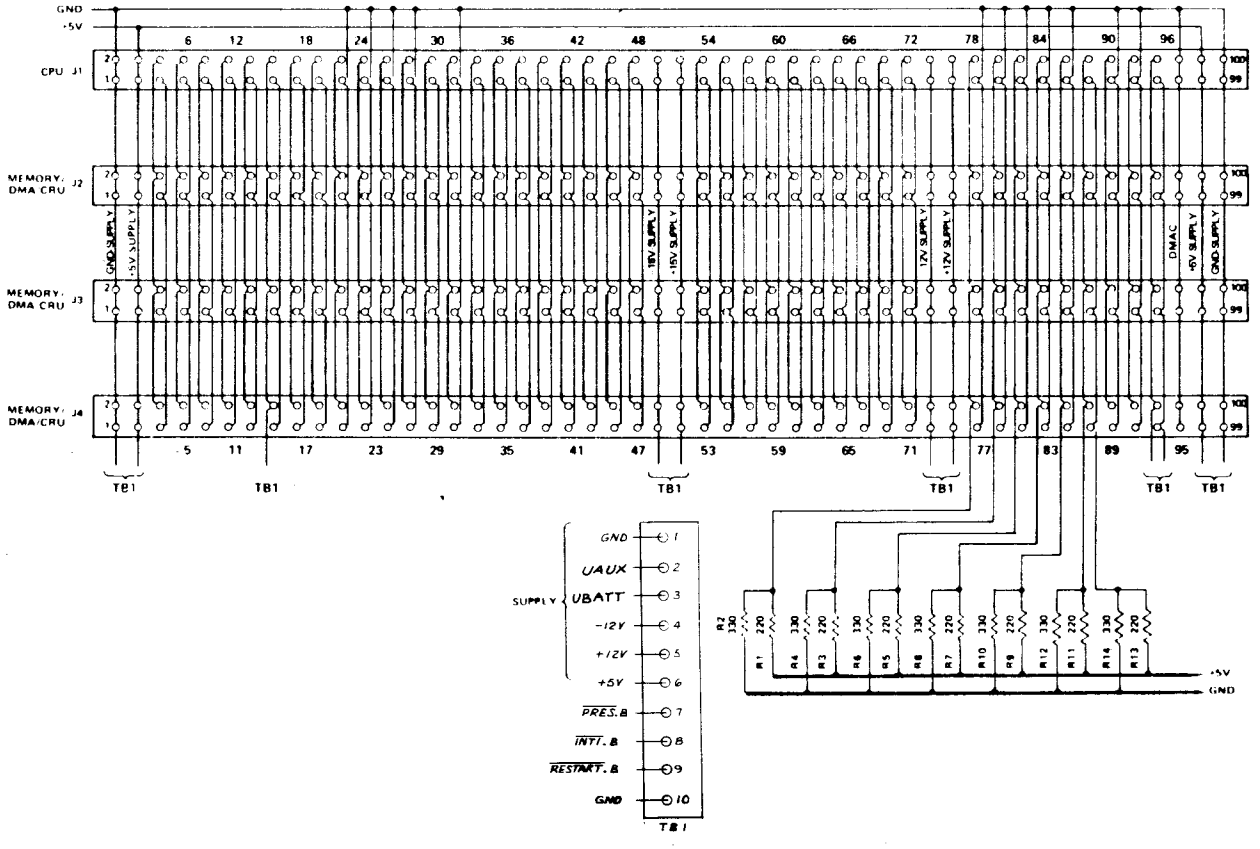


FIGURE 2-2. TM 990/510 CARD CAGE BACKPLANE SCHEMATIC

2.4.2 MEMORY MAPPING

The TM 990/203 can decode both 16 and 20-bit addresses. The TM 990/100M and TM 990/101M microcomputers generate 16-bit addresses. When the TM990/203 is used in a 16-bit address bus system, the 16/20 bit address jumpers(E89-E94) must be in the 16-bit position (E91 to E90 and E94 to E93). In this mode the 4 extended address bits (XA0 - XA3) are ignored by the /203 when decoding the memory address. If the /203 is used in a 20-bit address system, jumpers should be installed from E89 to E90 and E92 to E93. In this mode the /203 decodes the extended address bits so they must be driven by the CPU or a slave processor at all times.

The user can select the upper and lower boundaries for the TM 990/203 memory. The memory on the TM 990/203 can be set to operate on any 2K-word (4K-byte) boundary within the available address field. This upper and lower boundary selection is independent of the amount of memory actually populated on the module, so that the user is not required to use all the memory on the module (this feature is useful, for example, in reserving 4K bytes of memory for an EPROM loader).

Address boundary selection is determined by two sets of eight switches (one

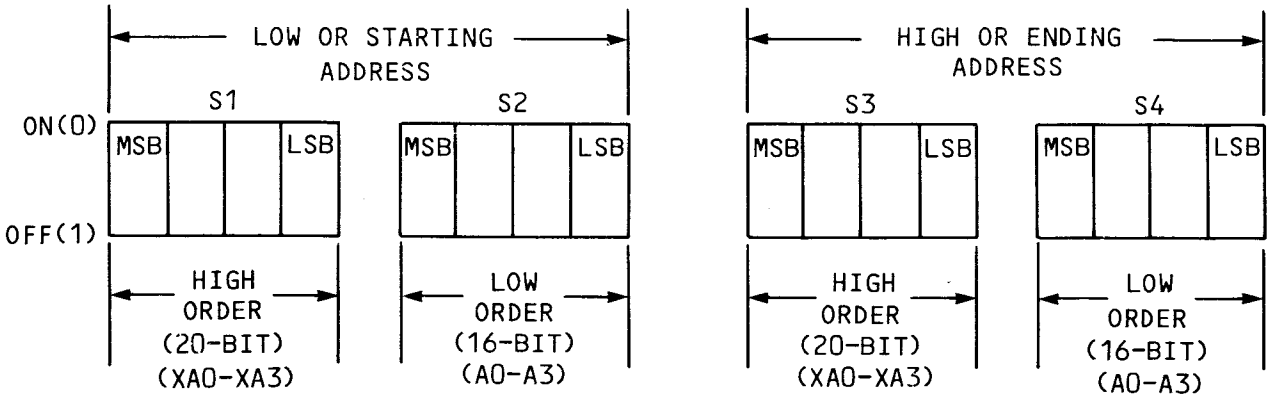


FIGURE 2-3. MEMORY MAPPING SWITCH ARRAY

set for low or starting address and one set for high or ending address). Figure 2-3 shows these switches (S1-S4).

Addresses in the general form of $VWXYZ_{16}$(20-bit address) or $WXYZ_{16}$(16-bit address can be mapped using this system. The high order switches S1 and S3 are used to set the value of 'V' and low order switches S2 and S4 are used to set the value of 'W'. 'V' may take on values from 0 to F (hexadecimal) that correspond to values 0000 to 1111 (binary) and 'W' similarly may take on values from 0 to F (hexadecimal) that correspond to values 0000 to 1111 (binary).

NOTE

If extended addressing is not used, as is the case with TM 990/100M and TM 990/101M microcomputer modules, the 16/20-bit jumper should be set in the 16-bit position. Switches S2 and S4 (Low order switches) are used to configure the memory on the TM 990/203 into the CPU module's available address space. The switch settings on S1 and S3 switches are irrelevent in this mode.

Table 2-2 lists the available switch codes and the corresponding starting and ending addresses (hexadecimal values). Switch codes 0000 (hexadecimal 0) through 1111_2(hexadecimal F) are available for memory mapping use. As an example of the use of this table, assume that switch code 0111 (hexadecimal 7) has been entered: if this code were set on starting address switch S2, a starting address of M.A. 7000_{16} would be mapped; if this code was set on ending address switch S4, an ending address of $7FFE_{16}$ would be mapped. The

user is cautioned not to set the difference between upper and lower memory addresses beyond the memory expansion capability of the TM 990/203 or the memory map will fold over or overlap.

If extended addressing is used, as is the case when a CPU module that generates a 20-bit address is used, then the 16/20-bit jumper should be set in the 20-bit position. Switches S1 through S4 will be used to configure the memory on the TM 990/203 into the CPU module's available memory space. Table 2-3 shows the available switch codes and the corresponding starting and ending addresses (hexadecimal values) for extended addressing. When two or more modules are used in a system, each can be viewed as pages in the memory map with S1 and S3 selecting page start and stop and S2 and S4 setting the bounds in the page or across the page boundary.

One additional feature involving switches S1-S4 is noteworthy: if a starting address is selected that is greater than the ending address, no memory will be selected. This feature can be used to disable the memory expansion module without removing it from the card cage.

TABLE 2-2 SWITCH CODE CONFIGURABLE ADDRESSES

Switch Code (Binary)	Starting Address Switch S2(Hex)	Ending Address Switch S4(Hex)
0000	0000	0FFE
0001	1000	1FFE
0010	2000	2FFE
0011	3000	3FFE
0100	4000	4FFE
0101	5000	5FFE
0110	6000	6FFE
0111	7000	7FFE
1000	8000	8FFE
1001	9000	9FFE
1010	A000	AFFE
1011	B000	BFFE
1100	C000	CFFE
1101	D000	DFFE
1110	E000	EFFE
1111	F000	FFFE

NOTE: ON = 0; OFF = 1; Place the 16/20 address-bit-jumper in the 16-bit position.

TABLE 2-3 SWITCH CODE CONFIGURABLE ADDRESSES (EXTENDED ADDRESSING)

Switch Code (Binary)	Starting Address Switches S1 & S2(Hex)	Ending Address Switches S3 & S4(Hex)
0000 0000	00000	00FFE
0001 0000	10000	10FFE
0010 0000	20000	20FFE
0011 0000	30000	30FFE
0100 0000	40000	40FFE
0101 0000	50000	50FFE
0110 0000	60000	60FFE
0111 0000	70000	70FFE
1000 0000	80000	80FFE
1001 0000	90000	90FFE
1010 0000	A0000	A0FFE
1011 0000	B0000	B0FFE
1100 0000	C0000	C0FFE
1101 0000	D0000	D0FFE
1110 0000	E0000	E0FFE
1111 0000	F0000	F0FFE

NOTE: ON = 0 OFF = 1; Place the 16/20 address bit jumper in the 20-bit position.

2.4.2.1 Memory Configuration Example. In order to properly configure the expansion memory into the CPU module's available address space, the CPU's memory map and the amount of expansion memory addresses must be known. In this example, a TM 990/101M CPU module will be used in conjunction with a TM 990/203-21 memory expansion module (this module provides an additional 8K words of RAM). The available memory address space for expansion extends from M.A. 1000_{16} to M. A. $EFFE_{16}$ (see Figure 2-4). To map the additional 8K words of DRAM from M. A. 1000_{16} to M. A. $4FFE_{16}$: switch S2 should be set to code 0001 (ON-ON-ON-OFF); switch S4 should be set to code 0100 (ON-OFF-ON-ON); and the 16/20 address bits jumper should be in the 16-bit position. The codes on S1 and S3 are irrelevant since XA0 - XA3 are ignored--the resulting memory map will be as shown in Figure 2-5.

NOTE

The memory mapping architecture of the TM 990 product line is explained in Appendix F.

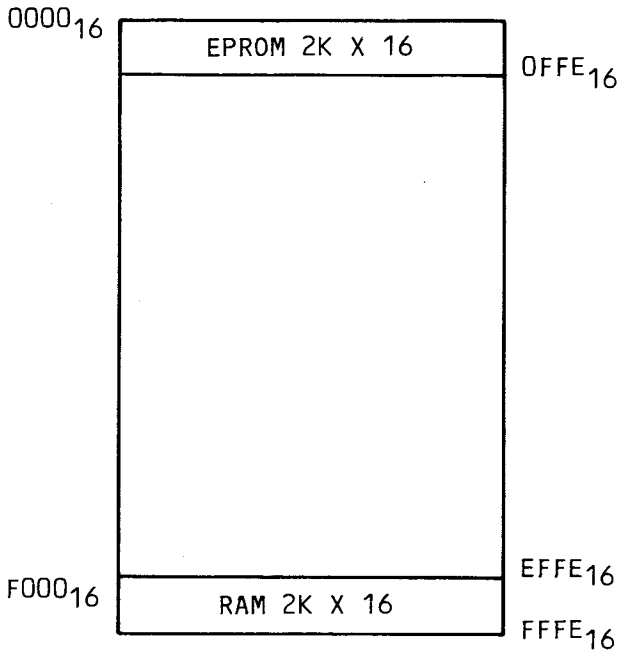


FIGURE 2-4. MEMORY MAP FOR TM 990/101M (2K X 16 EPROM, 2K X 16 RAM)

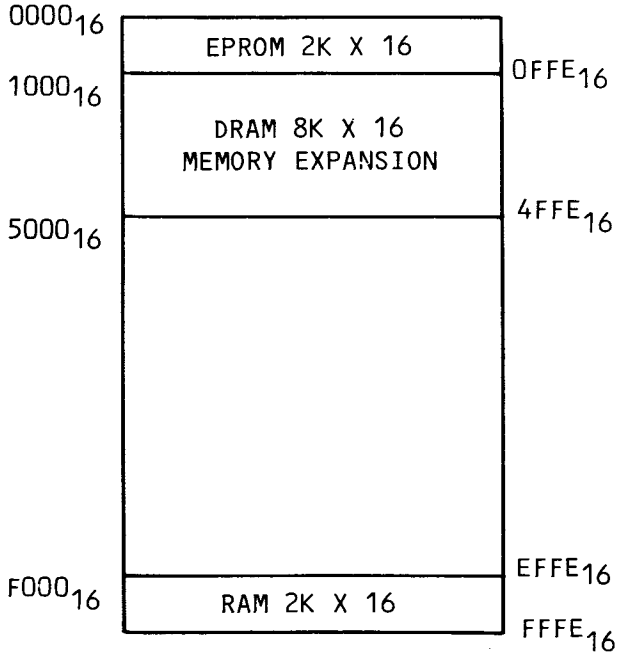


FIGURE 2-5. DESIRED SYSTEM MEMORY MAP (FOR TEXT EXAMPLE)

2.4.2.2 Memory Configuration Example: Extended Addressing. In this example of memory configuration, a hypothetical CPU module that uses extended addressing will be used in conjunction with a TM 990/203-22 memory expansion module (this module provides an additional 16K words of DRAM). Assume that the available memory space for expansion extends from M.A. 02000_{16} to M. A. $FFFFE_{16}$, and that the 16/20 address bits jumper is in the 20-bit position. In order to map the additional memory from M. A. 50000_{16} to M. A. $57FFE_{16}$: starting address switches S1 and S2 should be set to code 0101_20000_2(ON-OFF-ON-OFF ON-ON-ON-ON); and ending address switches S3 and S4 should be set to code 0101_20111_2(ON-OFF-ON-OFF ON-OFF-OFF-OFF).

2.4.3 Jumper Positions and Options

Table 2-4 provides a listing of the jumper options as shipped for the various configurations, Table 2-5 provides a brief description of the jumpers, Table 2-6 provides a description of each jumper connection, and Figure 2-6 identifies jumper pins and memory banks. Appendix A provides two additional tables that provide extensive information regarding several additional options. The jumpers that are provided on the TM 990/203 are intended to provide the following accomodations:

- Future memory expansion
- Refresh period selection
- Wait state selection
- Cycle steal or transparent refresh.
- Processor Address bus size

TABLE 2-4 JUMPER POSITIONS AS SHIPPED

TM 990/203-21	TM 990/203-21A	TM 990/203-22	TM 990/203-23
E5 to E4	E5 to E4	E5 to E4	E5 to E4
E8 to E7	E8 to E7	E8 to E7	E8 to E7
E13 to E10	No jumper from E13	No jumper from E13	E13 to E14
E16 to E15	E16 to E18	E16 to E18	E16 to E18
E20 to E19	E20 to E22	E20 to E22	E20 to E22
E23 to E24	E23 to E24	E23 to E24	E23 to E24
E27 to E28	E27 to E28	E27 to E28	E27 to E28
E31 to E35	E31 to E34	E31 to E34	E31 to E34
E45 to E65	E45 to E65	E45 to E65	E45 to E65
E47 to E46	E47 to E48	E47 to E48	E47 to E48
E73 to E74	E73 to E74	E73 to E74	E73 to E74
E82 to E83		E82 to E83	E82 to E83
E84 to E85	E87 to E83 or	E84 to E85	E84 to E85
	E88 to E83		
E91 to E90	Wirewrap wire	E91 to E90	E91 to E90
E94 to E93	E84 to E82	E94 to E93	E94 to E93
	E91 to E90		
	E94 to E93		

TABLE 2-5 JUMPER CONNECTION DESCRIPTION

Jumper Position*	Description	Schematic Sheet
E5 to E4	Bus signal INT15.B-/P7; controls parity interrupt; high/low enables/disables and clears interrupt	2
E5 to E6	CRU output (1 or 0) signal resets parity interrupt(See 2.5.2)	2
E8 to E9	Sets parity interrupt to level 1 via bus signal INT1.B-	2
E8 to E7	Sets parity interrupt to level 2 via bus signal INT2.B-	
open	Disconnects parity interrupt line from System Bus	2
E13 to E10	For 4K RAMS(TMS 4027's)	3
E13 to E14	For 8K or 16K RAMS(TMS 4108 or 4116's)	3
E13 to E11	For access to Bank A or Bank B; low/high enables Bank B/A(see 2nd par. of 2.4.3.3)	3
E13 to E12	For future expansion	3
E66 to E68	For byte-oriented microprocessor/system bus	3
E66 to E67	For future expansion	3
No jumper to E13 and E66	Bank A selected(only)	3
E16 to E15	For 4K RAMS	2
E16 to E18	For 8K or 16K RAMS	2
E16 to E17	For future expansion	2
E20 to E19	For 4K RAMS	2
E20 to E22	For 8K or 16K RAMS	2
E20 to E21	For future expansion	2

(CONTINUED)

* see Figures 2-6, 2-7 and 2-8 for jumper locations

TABLE 2-5 JUMPER CONNECTION DESCRIPTRIONS (CONTINUED)

Jumper Connection	Description	Schematic Sheet
E23 to E24 & E27 to E28	For 0 wait states	4
E23 to E25 & E27to E26	For 1 wait state	4
E23 to E26 & E27 to E30	For 2 wait states	4
E31 to E32	For 256 refresh cycles/ms	5
E31 to E33	For 128 refresh cycles/ms	5
E31 to E34	For 64 refresh cycles/ms(TMS 4116)	5
E31 to E35	For 32 refresh cycles/ms(TMS 4027)	5
E59 to E60	0 cycles/ms (test mode)	5
E59 to E61	TM 990 system: For external refresh (under control of INT15.B/P7) Other systems: For forcing refresh cycle(1 must be valid for one positive transition of BUSCLK.B-)	5
E45 to E44	Transparent refresh(with cycle steal on refresh violation)	4
E45 to E65	Cycle-steal refresh	4
E47 to E46	For 4K memory devices	3
E47 to E48	For 8K or 16K memory devices	3
E73 to E74	For processors requiring 2 clock cycles per memory cycle	4
E73 to E75	For a processor which has the potential for a memory cycle having a duration of a single clock cycle	4

TABLE 2-5 JUMPER CONNECTION DESCRIPTRIONS (CONTINUED)

Jumper Connection	Description	Schematic Sheet
E78 to E79	custom application	3
(open)	normal operation	
E83 to E82	For use when TMS 4027 or TMS 4116 memories are utilized	2
E83 to E87*	For TMS 4108-25-0 memories	
E83 to E88*	For TMS 4108-25-1 memories	
E84 to E82**	For TMS 4108 Memories (wirewrap wire)	2
E84 to E85	Jumpered when TMS 4027 or TMS 4116 are used	
E89 to E90 & E92 to E93	20-bit address bus (extended addressing)	2
E91 to E90 & E94 to E93	16-bit address bus(TM990/100,101)	2

** E83 must be in TMS 4108 position
* E84 to E82 connected via wirewrap wire

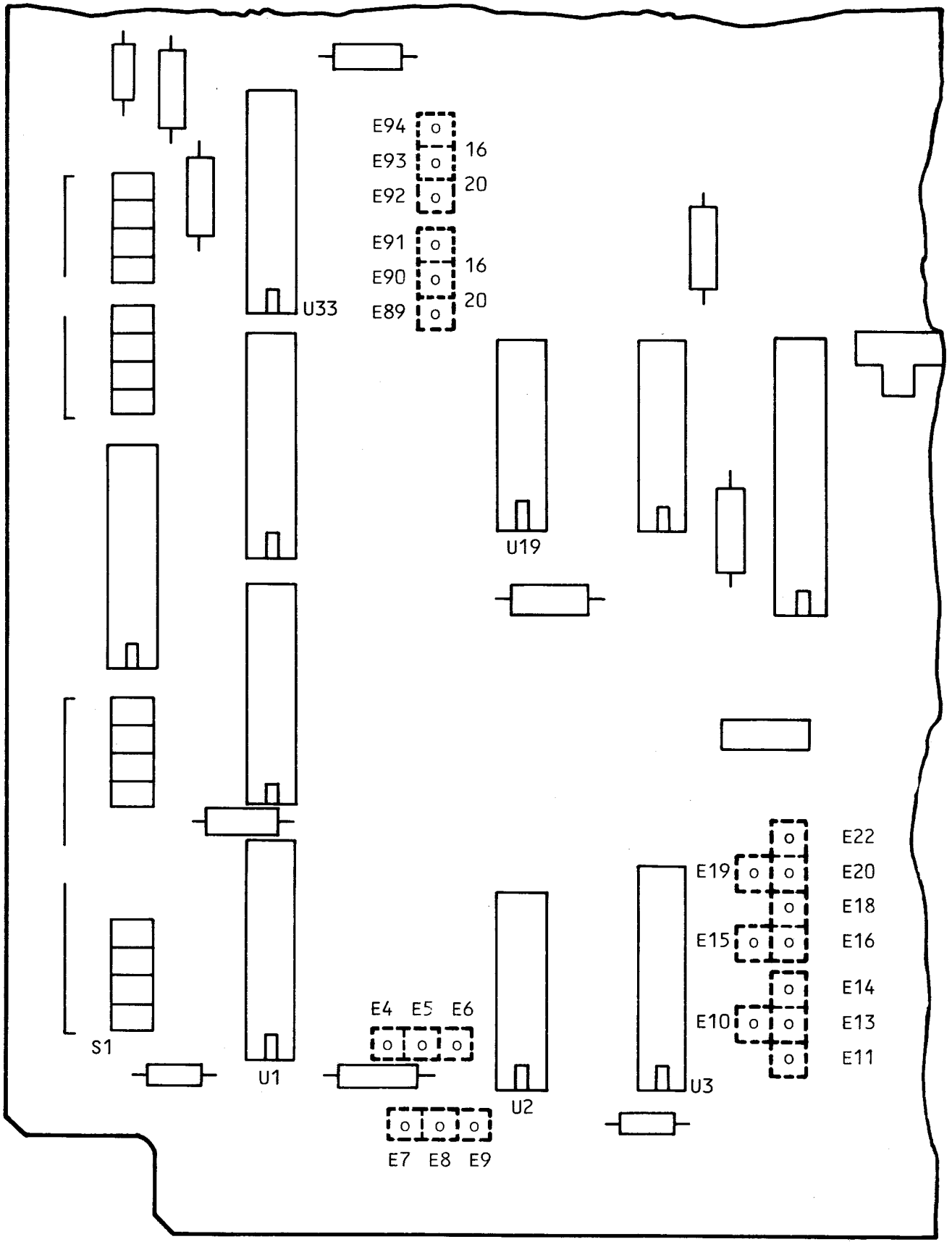


FIGURE 2-6 LOCATION OF JUMPER PINS ON LEFT THIRD OF BOARD

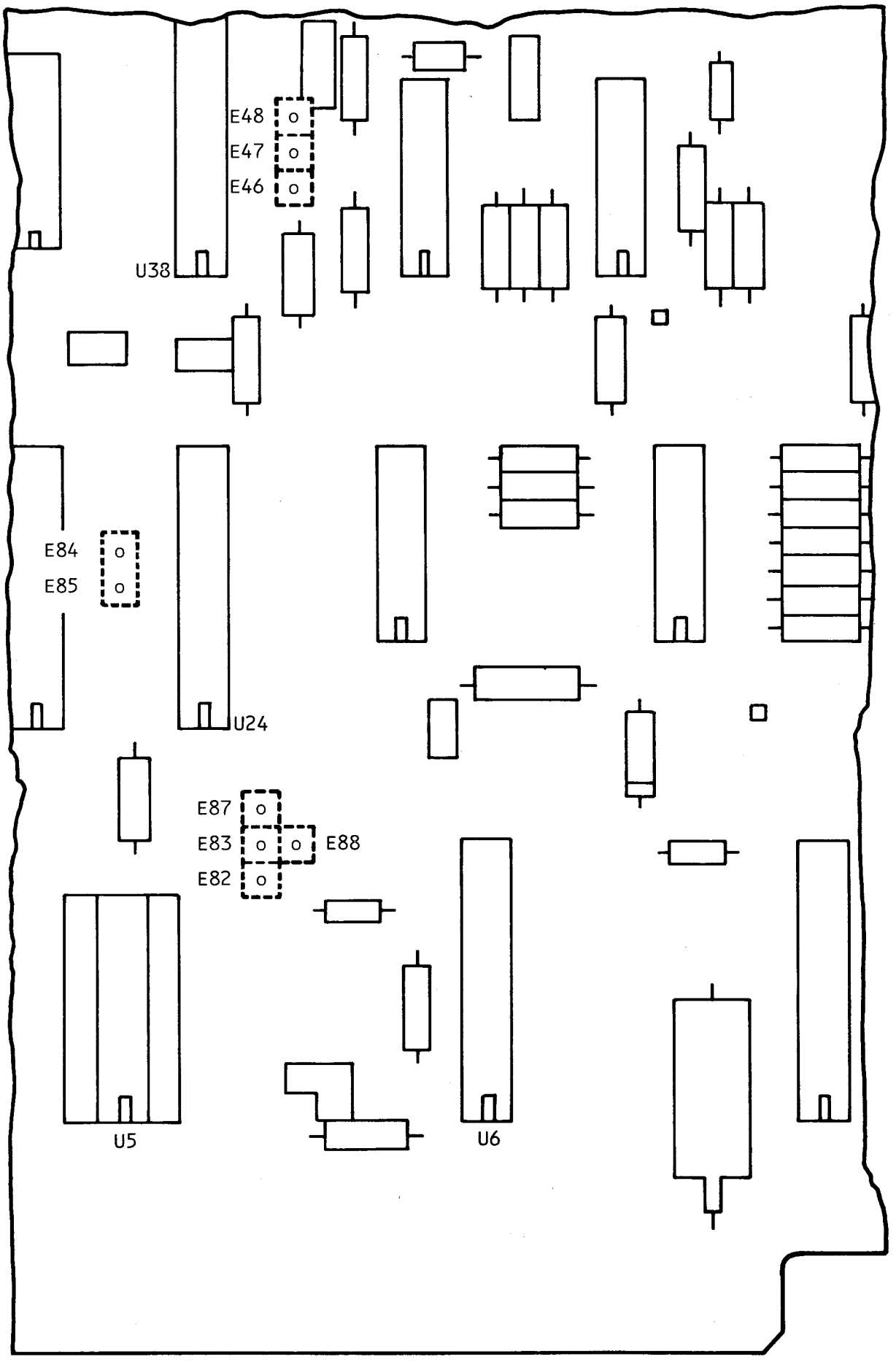


FIGURE 2-7 LOCATION OF JUMPER PINS ON MIDDLE THIRD OF BOARD

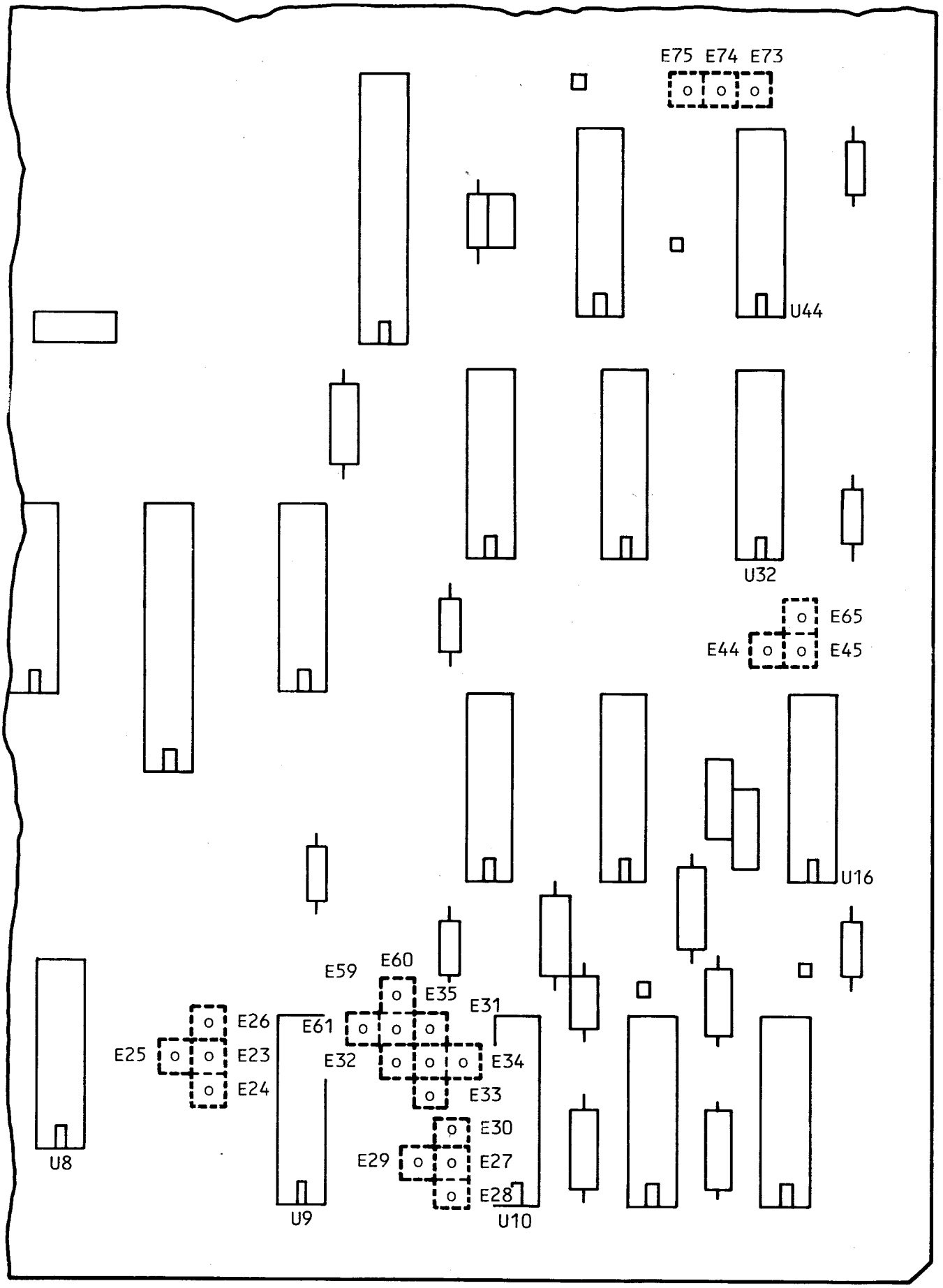


FIGURE 2-8 LOCATION OF JUMPER PINS ON RIGHT THIRD OF BOARD

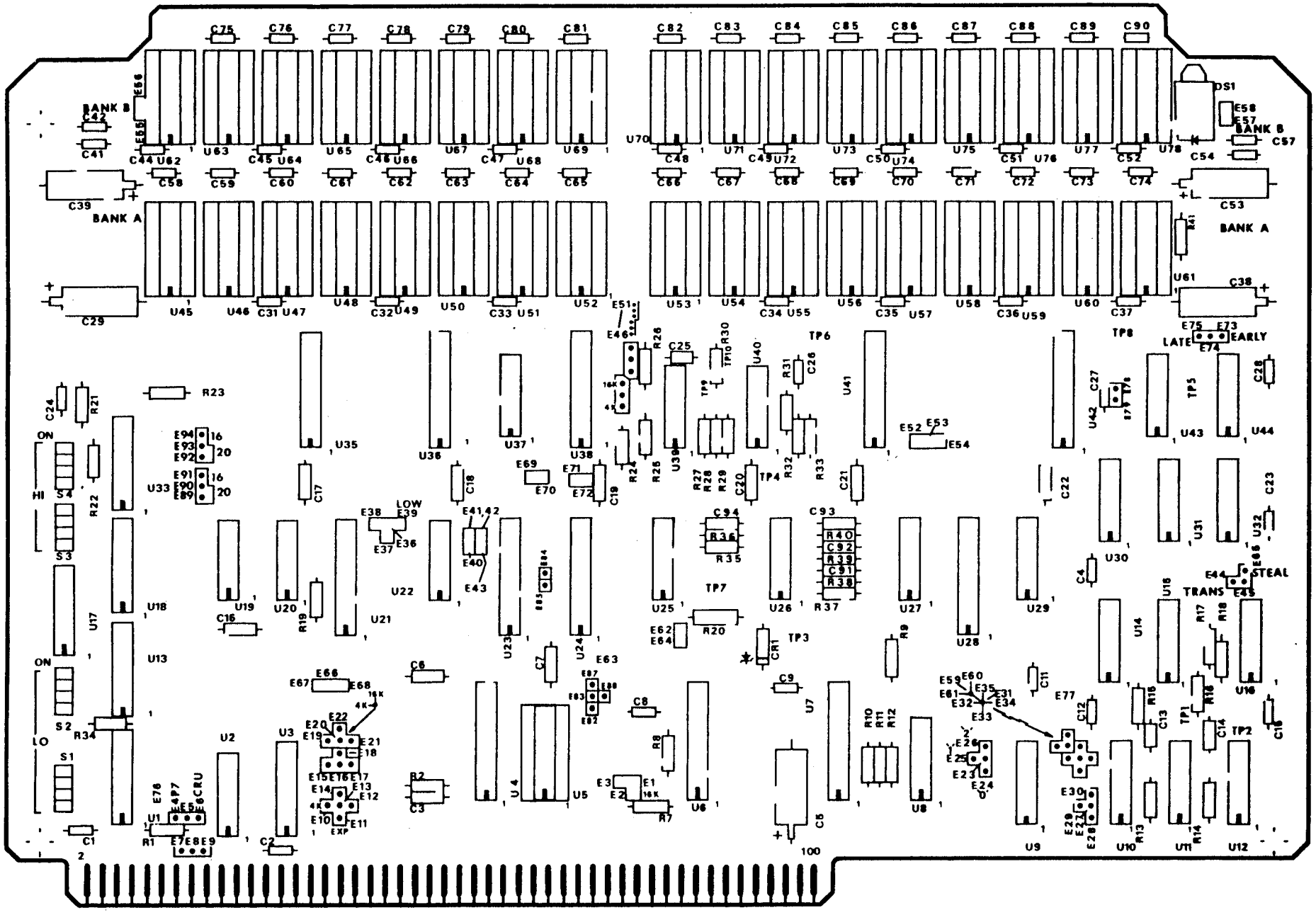


FIGURE 2-9. TM 990/203 JUMPER PIN AND MEMORY BANK LOCATIONS

2.4.3.1 Select Wait State. For processors operating at clock rates faster than 3.2 MHz and for TMS 4027 and TMS 4116 memory devices with access times other than 250 ns, consult Table 2-7 to determine the required number of wait states that are needed. As an example, for a processor operating at 4 MHz and a memory module using memory devices with a 300 ns access time, one wait state (1 WS) would be required.

TABLE 2-6. WAIT STATES

Clock Rate	Memory Access Time(WS)			
	300 ns	250 ns	200 ns	150 ns
3-3.2 MHz	0	0	0	0 WS
3.2-3.5 MHz	1	0	0	0 WS
3.5-3.9 MHz	1	1	0	0 WS
3.9-5.0 MHz	1	1	1	0 WS
5.0-6.0 MHz	2	1	1	0 WS

Table 2-8 lists the jumper connections that are required for the various wait states. It should be noted that two jumper connections are needed to properly select the required wait state. As an example, connections from E23 to E25 and E27 to E29 should be made for one wait state.

TABLE 2-7. WAIT STATE JUMPER CONNECTIONS

	Number of Required Wait States		
	0	1	2
Jumper Connections	E23 to E24 E27 to E28	E23 to E25 E27 to E29	E23 to E26 E27 to E30

2.4.3.2 Cycle Steal and Transparent Refresh. There are two dynamic refresh techniques that are used with the TM 990/203: cycle steal and transparent refresh. The choice between the two techniques essentially involves a tradeoff between power consumption and throughput. Table 2-9 lists the advantages and disadvantages of each technique.

TABLE 2-8 COMPARISON OF CYCLE STEAL AND TRANSPARENT REFRESH MODES

	Advantages	Disavdvantages
Cycle Steal:	Less power consumed	Less throughput(with 0 WS)
Transparent Refresh:	More throughput(see note)	More power consumed

CAUTION

If more than one wait state is required, the transparent refresh mode may actually interfere more with processor operation than the cycle steal refresh mode: this interference will result in less throughput.

Assuming that memories requiring no wait states are used, and maximum throughput is the main concern, the transparent refresh mode should be used.

On the other hand if minimal power consumption is the main concern, the cycle steal mode should be used. Table 2-10 shows the jumper connections that are required for cycle steal or transparent refresh operation.

TABLE 2-9 CYCLE STEAL/TRANSPARENT REFRESH JUMPER CONNECTIONS

	Cycle Steal	Transparent Refresh
Jumpers:	E45 to E65	E45 to E44

2.4.3.3 Bank Select Circuitry. Bank selection on the TM 990/203 is accomplished by jumpering (via E13) the most significant bit needed for the TM 990/203's memory size to the bank select circuitry. Thus on a 8K X 16 (two rows 4027s) module, A2 selects one of the two 4K X 16 banks; on a 16K X 16 module, the jumper is removed (since only Bank A is populated); and for a 32K X 16 (two rows 4116s) module, A0 selects Bank A or Bank B.

Another bank select jumper option available on the TM 990/203 is to select Bank A or Bank B with I/O Port 8 of the TMS 9901 (on TM 990/100M or TM 990/101 CPU modules). This method allows banks to be switched in and out of the same memory space by outputting a logic "0" or logic "1" on Port 8 via CRU. For example, assume both banks are populated with TMS 4116S (16K X 16 per bank) and it is desired that the banks be switched in and out of the same memory space. The memory enable switches should be set to enable 16K words (32K bytes) of memory space at an unused area in the memory map. Then, with a SBZ or SBO instruction at CRU address 130_{16} (in R12) each bank can be switched in or out of the memory map by software control (a logic "1" enables Bank A, a logic "0" enables Bank B). The bank that is deselected is totally inaccessable and invisible to the processor until the logic value on P8 is changed.

Note that this bank select feature dies not reflect the memory map architecture of the TM 990 product line; this architecture is explained in Appendix F.

2.4.3.4 Address Bus Jumpers. The TM990/203 gives the user the option of either a 16 or 20-bit address bus. When used with a processor which drives a 16-bit address bus(such as the TM990/101) E90-E91 and E93-E94 should be jumpered. Likewise, when a processor which drives a 20-bit address bus is chosen, E89-E90 and E92-E93 should be jumpered. Both jumpers must be changed to select the desired address bus option.

The 16/20-bit address jumpers cause the /203 to ignore the extended address lines (XA0-XA3) when in the 16-bit position, and the /203 is not affected in any way by the values on the extended address lines(XA0-XA3). When in the 20-bit position, the extended address bits are decoded and the /203 may then be used in a memory-mapped system.

2.5 OPERATION

2.5.1 Operation Without Parity

In order to operate the TM 990/203 without parity the user must remove the jumper connecting E8 to either E7 or E9. In doing this the interrupt request generated aboard the TM 990/203 is not routed to the bus and thus isolated from the system. This configuration allows the user to operate the TM 990/203 essentially as a static memory, and can use all of the TM 990 software

monitors with no modifications as long as the powerup requirements for the

memories are observed (see powerup, Section 2.5.2). In this mode, the parity error LED should be ignored since the parity bits were not initialized.

2.5.2 Operation With Parity

The TM 990/203 will interrupt the processor in the event of a parity error if operation with parity is used. In order to operate the TM 990/203 with parity the user must make two jumper connections and generate an interrupt handler software routine. The required connections are as follows:

1. Connect a jumper from E8 to E9 (interrupt level 1 enable) or from E8 to E7 (interrupt level 2 enable).

2. Connect a jumper E5 to E4 (resets the interrupt via a logic '0' outputted on the INT15-/P7 line) or from E5 to E6 (resets the interrupt via a logic '0' or a logic '1' outputted on the CRUOUT line). It should be noted that in the CRU mode, the user must supply a SN74S287 PROM programmed to enable the interrupt reset circuitry at a specific user-selectable word of CRU. For more information on the CRU method, see Appendix C.

In addition to the jumper connections, the user must generate an interrupt handler software routine which resets the parity interrupt and flags the memory location which is in error. The exact location in error may be found by the interrupt routine if a read and write operation is performed (with all additional interrupts masked) on the entire RAM memory while the software monitors the interrupt line (INT1 or INT2 which are jumper selectable) to check for errors.

If the TM 990/203 is used with the parity option, a parity error will occur when memory is first accessed unless the parity bits are first initialized. These bits can be set by writing to every location. This allows the parity generator/checker on the TM 990/203 to generate the correct parity bits and write them into the extra parity bit associated with each word. These parity bits are stored exactly the same way as the data in an extra memory device. When the data is read back, the parity bit is also read and compared with the parity bit regenerated from the data. If an error occurs, the parity flip-flop is set and an interrupt request is generated. For an example of a software powerup routine see Appendix D.

2.5.3 General

Essentially the user needs only to choose the correct memory configuration (Section 2.4.2), insert the module into the card cage, and apply power to set up the system for operation.

The operation of the TM 990/203 memory module should be transparent to the user in that no special signals are required other than those supplied through the backplane of the TM 990/510 or TM 990/520 card cage or through an equivalent interface cable.

SECTION 3

THEORY OF OPERATION

3.1 GENERAL

This section covers the theory of operation for the TM 990/203 dynamic RAM memory board. Figure 3-1 is a block diagram of the board.

3.1.1 RAM Area

RAM is populated on-board on two banks (banks A & B) of 17 RAM chips: one chip for each bit in the 16-bit data word; and an additional chip for a parity value for each word. Figure 2-6 shows the physical locations of banks A and B on the memory board. Table 1-1 lists the memory configurations available according to dash number. Present configurations shipped are as shown in Table 1-1.

3.1.2 Parity Logic

The memory board generates one bit of parity for each 16-bit word written to memory. This parity value is placed in the parity memory chip in the memory bank. The board utilizes odd parity; thus, an even number of ones in the 16-bit word results in a one written to the parity bit (and a zero for an odd number of ones). When a word is read from the memory, parity is again generated and this value is compared to the stored parity bit; an interrupt request is generated if the bits do not match. Jumpers allow this interrupt to be ignored or to be routed to INT1.B- or INT2.B- on the system bus.

3.1.3 Memory Controller

The principal duties of the memory controller include:

- Monitoring the memory system to ensure that each memory bit is refreshed properly;
- Ensuring that the processor-generated memory-access cycles and the memory-board-generated refresh cycles do not interfere with each other.

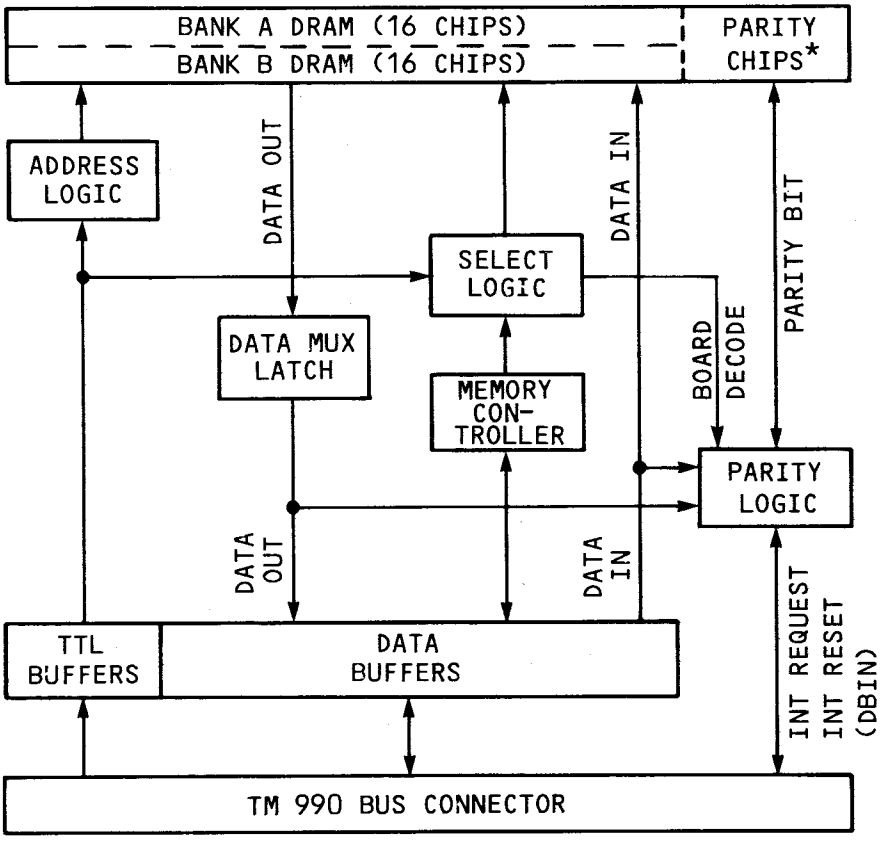
In general, the controller directs the memories and the system bus according to the inputs the controller receives from the processor or the on-board refresh circuitry.

3.1.4 Data Buffers

The system bus data lines (D0.B to D15.B) are buffered by 74LS245 octal bus transcievers on the memory board. These tri-state buffers go into the high-impedance state when the memory board is not being accessed, allowing other modules on the system bus to utilize the bus. The data buffers accept data from the system bus to the memory board and vice versa (memory board to system bus) when the memory is enabled.

3.1.5 Data Multiplexer and Latch

The data multiplexer and latch logic multiplexes the data read from memory and latches the data until it is read by the processor. Since some of the memory



*ONE PARITY CHIP FOR EACH BANK

FIGURE 3-1 TM 990/203 MEMORY BOARD BLOCK DIAGRAM

chips that can be possibly used on the TM 990/203 do not have tri-state outputs, the multiplexer selects either memory bank A or bank B output and gates this output into the latch. The latch also ensures valid data until the end of the memory cycle.

3.1.6 Wait State Logic

The TM 990/203 memory board is equipped with wait state circuitry to interface with fast processors operating up to clock speeds of 6 MHz. The number of wait states (0, 1, or 2) is jumper selectable. Table 2-7 shows the wait state needed according to memory access time and clock speed.

3.1.7 Address and Select Logic

The addressing scheme of the TM 990/203 board allows the user to locate his memory in blocks that begin on address boundaries separated by 1000_{16} (e.g., 0000_{16}, 1000_{16}, 2000_{16}, 3000_{16}, etc.). Four groups of pencil switches (four switches in each group) allow the user to select the beginning memory address and the ending memory address for systems using a 16-bit address or an extended 20-bit address depending on the 16/20-bit address jumper setting. When the 16/20-bit address bus jumper is in the 20-bit position, the beginning-address switches will be the two most-significant digits of the five-digit hexadecimal address. The ending address setting will be the two most-significant digits of the five-digit hexadecimal address plus $00FFE_{16}$. For example, if the beginning address switches are set to 03 and the ending address switches are set to 0E, the beginning address of memory is 03000_{16} and the ending address is $0EFFE_{16}$ inclusive. Also, if the beginning and ending address switches are both set to 03, the beginning address is 03000_{16} and the ending address is $03FFE_{16}$. When the address bus jumper is in the 16-bit position, the extended address line values (S1 and S3) are ignored; therefore, the settings of switches S1 and S3 are irrevalent. If the value of the beginning-address switch is larger than the ending address switch, the memory will be deselected from the board. Paragraph 3.4.4 further explains memory selection.

3.1.8 Memory Refresh

Dynamic RAM's must be refreshed periodically to avoid loss of stored data. Since the refresh process and memory access by the processor cannot occur at the same time, consideration must be given to the timing of these two functions and the possibility that the processor may have to wait while the refresh process "steals" processor time for memory refresh. By use of jumpers, the user can select one of two refresh methods:

- Transparent refresh--the refresh cycle is synchronoized to the processor memory cycle to ensure that it does not access memory during the memory access time of the processor;

- Cycle-steal refresh--the refresh circuitry accesses memory at specified time intervals; if the processor requires memory while refresh is occurring, it must wait until the refresh is finished.

3.2 MEMORY CYCLE FLOW CHART

Figure 3-2 is a flow chart detailing the flow of control for a memory cycle(Path 1 on the left side of the figure) and for a refresh cycle (Path 2 on the right).

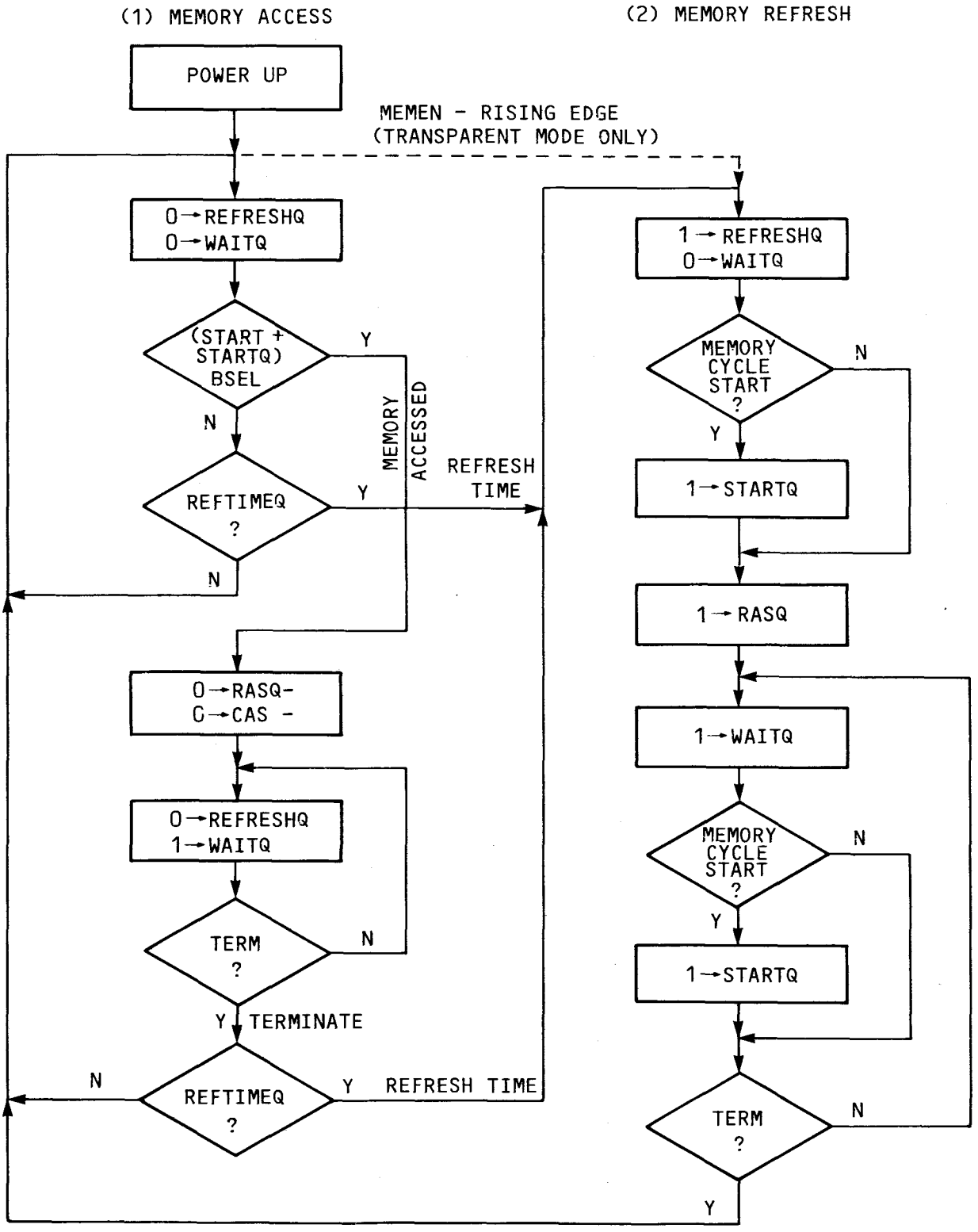


FIGURE 3-2 MEMORY CYCLE FLOW CHART

During the memory cycle path, a memory access cycle begins: if there are no wait states or refresh cycles in progress; and if there is (or has been) an attempted memory access and a board select. If all these conditions are met, a row address strobe (RAS) and then a column address strobe (CAS) are generated that strobe the row and column addresses to the memory chip. The memory controller then sets the wait state flip-flop and waits for a TERM pulse(terminate memory cycle), which is controlled by the number of wait states selected(by jumper) by the user. After a TERM pulse is received, the controller either returns to the beginning of the memory access cycle loop, or if REFTIMER is active (high), the controller branches to the start of the refresh cycle loop.

Transparent refresh is initiated by the rising edge of every MEMEN- (memory enable) and is represented by the dotted line in Figure 3-9. Thus, at the end of every memory access (onboard and offboard memory), the TM 990/203 initiates a refresh cycle. For more information on refresh, see subsection 3.3.

A refresh cycle consists of first setting the REFRESHQ flip-flop(0 to WAITQ) which signals the start of a refresh cycle. The row address strobe (RAS) then strobes in the row address(column addresses are not needed in a refresh) to both memory banks. Then the refesh flip-flop and the wait flip-flop are both set(1 to REFRESHQ, 1 to WAITQ) until a terminate (TERM) ends the refresh cycle. If at any time during a refresh cycle the processor attempts to access the memory board(1 to STARTQ), the memory controller halts the processor(brings READY low) until the refresh cyucle is completed, and then allows the memory cycle to proceed.

3.3 REFRESH

The RAM data cells are organized into a matrix of rows and columns with on-chip gating to select the addressed bit. Refresh of the RAM cell matrix is accomplished by performing a memory cycle on each of the 64 (4K RAM) or 128 (16K RAM) row addresses every 2 ms or less. Performing a memory cycle at any cell on a row refreshes all cells in that row. Note that only row refresh is necessary; therefore, columnm addressing is not used for refresh.

There are several dynamic memory refresh techniques which can be used for a TMS 9900 system. If the system periodically accesses at least one cell of each row every 2 ms, then no additional referesh circuitry is required. A CRT controller which refreshes the display periodically is an example of such a system.

Refresh control logic must be included, however, in most systems since the system cannot otherwise ensure that all rows are refreshed every 2 ms. Several refresh methods exist that can be used on TMS 9900 systems:

- block(not used on TM 990/203)
- cycle stealing (jumper selectable on TM 990/203)
- transparent (jumper selectable on TM 990/203)

Only two of the above (previous page) refresh modes are available on the TM 990/203: cycle stealing and transparent refresh; chosing the mode is jumper selectable at E45. An explanation of the block refresh mode is provided in the following paragraph for information only since it is not available on the board.

3.3.1 Block Refresh

The block refresh mode halts the CPU every 2 ms and then refreshes all of the rows individually. This halt period lasts 64 memory cycles for a 4K RAM and 128 memory cycles for a 16K RAM. Some TMS 9900 systems cannot use this technique because of the possible slow response to priority interrupts or because of the effect of the delay during critical software timing loops or I/O routines. It is for this reason that block refresh is not used on the TM 990/203 memory board.

3.3.2 Cycle Stealing Refresh

The cycle stealing mode is selected on the TM 990/203 by jumpering E45 to E65. This mode "steals" a cycle from the system periodically to refresh one row. If the processor wishes to access memory during refresh, it must wait until refresh is complete; thus, the refresh process "steals" memory access time from the processor.

The refresh interval is determined by the maximum refresh time and the number of rows to be refreshed. The 4K dynamic RAM's have 64 rows to be refreshed every 2 ms; thus, the maximum possible time "stolen" from the processor will be 31.2 micro-seconds every 2 ms. Likewise, the 16K RAM's have 128 rows to be refreshed; thus, the maximum possible time stolen(assuming zero wait states will be 62.4 micro-seconds every 2 ms.

During the refresh cycle, the CPU can attempt to access the dynamic memory since the CPU is not synchronized to the refresh controller. If the CPU attempts to access memory during any clock cycle of the refresh memory cycle, the refresh controller places the CPU in a wait state during this interval.

The specific row address to be refreshed is provided by a counter which keeps track of the next row to be refreshed. The counter is incremented after each refresh cycle, setting the address for the next refresh cycle to allow the rows to be refreshed sequentially.

3.3.3 Transparent Refresh

The transparent refresh mode is selected on the TM 990/203 board by jumpering E45 to E44. This mode differs from the cycle stealing refresh mode in that refresh occurs only when the processor is not accessing memory; thus, cycles do not have to be "stolen." This is possible by synchronizing the refresh cycle to the CPU memory cycle. The rising edge of MEMEN- from the processor marks the end of a processor memory cycle and the start of at least one non-memory cycle by the processor. This is an indication to the TM 990/203 that the next two clock cycles are available for refresh. Thus, a memory refresh cycle can begin at the rise of MEMEN- without interfering with the processor's memory access. That is, if there are no wait states, the memory cycle is two clock cycles long. Using the worst-case situation of the processor executing continuous divides, sufficient non-processor memory cycles exist to guarantee complete memory refresh within 2 ms.

While the transparent refresh mode eliminates refresh-related system performance degradation (for zero wait states), the system power consumption can be higher since the RAM's are refreshed more often than required. As many as one-half of the CPU machine cycles can generate refresh cycles, resulting in multiple refresh cycles for each row during the required 2 ms refresh interval.

Another disadvantage of transparent refresh with the TMS 9900 occurs when transparent refresh is used with memory wait states; with each memory access or refresh taking three or more clock cycles. Using an example with one wait state, at every rising edge of MEMEN (end of processor memory cycle), the memory controller initiates a refresh cycle which then consumes three clock cycles to complete the refresh. If the processor tries to again access memory within three clock cycles of the previous memory cycle, it must wait for the refresh cycle to complete (wait for READY to be brought high by the memory controller), thus decreasing memory efficiency. Therefore, if one or more wait states are necessary, the transparent refresh mode may actually interfere more with processor operation than the cycle steal refresh mode.

3.3.4 Direct Memory Access (DMA)

DMA is possible on the TM 990/203 board independent of the type of refresh used. Since the transparent mode reverts to cycle steal upon refresh violation, DMA transfers and other hold conditions may occur without destroying data. The following are requirements for DMA devices used with the TM 990/203 board:

a. The DMA device must use the system clock.
b. The DMA device must use the READY/NOT READY handshake methods for data transfer as outlined in the TMS 9900 Microprocessor Data Manual, and the TM 990 System Specification
c. Memory timing must be synchronized to the microcomputer board.

The transparent refresh mode on the TM 990/203 is not strictly transparent in that any time a refresh violation occurs, the memory board will steal enough memory cycles to ensure that the memory remains refreshed. This allows DMA

transfers and other HOLD conditions to occur without destroying data on the TM 990/203 board.

3.4 CIRCUIT DESCRIPTIONS

3.4.1 Controller Section (Schematic Sheet 4)

Figure 3-3 shows the major components of the state controller section of the TM 990/203 memory board. Every occurrance of MEMEN (from the system bus via P1-80) causes a memory cycle to start by enabling the NAND gate at U30. If a board select (BSEL) is not present, the memory cycle ends since BSEL low disables U29 and U14. However, if BSEL is present and no refresh cycle or wait state delay are in progress (U32-4,5,6 and U32-8,9,10), a normal memory access cycle begins. The output of RASD- from U14-8 starts the row address strobe and column address strobe sequence, and sets the Q- output of U31 (WAITQ-). After the jumper-selected number of wait states have occurred, the terminate flip-flop (U9) resets the controller to ready it for another refresh cycle.

If a refresh cycle is already in progress (REFRESHQ high) when the processor (issuing MEMEN-) tries to access onboard memory, the attempt is sensed at U29, pins 12 and 13; this enables signal STARTJ which is clocked into the STARTQ- flip-flop at U31. If BSEL at U8-9 and REFRESHQ1 at U8-10 are enabled, the READY.B line to the processor goes low, causing the processor to enter a wait state. When the refresh is completed, STARTQ- is gated through U30-1 and RASD- becomes active which starts a normal memory access cycle by the processor.

A refresh cycle is initiated either by (1) the rising edge of MEMEN- which means a memory cycle is over (transparent refresh) or (2) the timing out of the refresh time clock (cycle steal refresh).

In the transparent mode the rising edge of MEMEN- clocks MEMENQ- active at U16 (schematic sheet 5) which (through jumper E44-E45) presets the REFRESHQ flip-flop (U15 on Figure 3-3) to start a refresh cycle. Since the TMS 9900 processor guarantees that there is at least two unused clock cycles after the rising edge of MEMEN-, the transparent mode does not have to halt the processor to complete the refresh cycle (assuming zero wait states). If the processor is in a hold state (MEMEN- high), transparent refresh reverts to cycle stealing upon a refresh time violation.

The cycle steal refresh mode initiates a refresh cycle when the jumper-selected (E31) amount of time has occurred at the refresh timer U10 (schematic sheet 5). When the refresh timer times out, REFTIMER becomes active to flip-flop U15 (Figure 3-3) where REFTIMEQ is clocked in by the system clock. REFTIMEQ then waits (U8-4,5,6 or U14-2,3) until the current memory cycle is completed (if one in progress) before clocking the refresh request into the REFRESHQ flip-flop through U14-3,2 (or U14-4,5) and U32. Once REFRESHQ becomes active, it disables (through U29) any onboard memory accesses until the refresh cycle is completed by pulling the READY.B line low to halt the processor.

Once a refresh cycle is in progress, it behaves exactly the same as a memory access except that there are no column address strobes (CAS) since only rows need to be addressed during a refresh. The row addresses are generated by U37 which increments the row number after each refresh. (Further explanation of row address generation is covered in paragraph 3.4.4 which references Figure 3-7.)

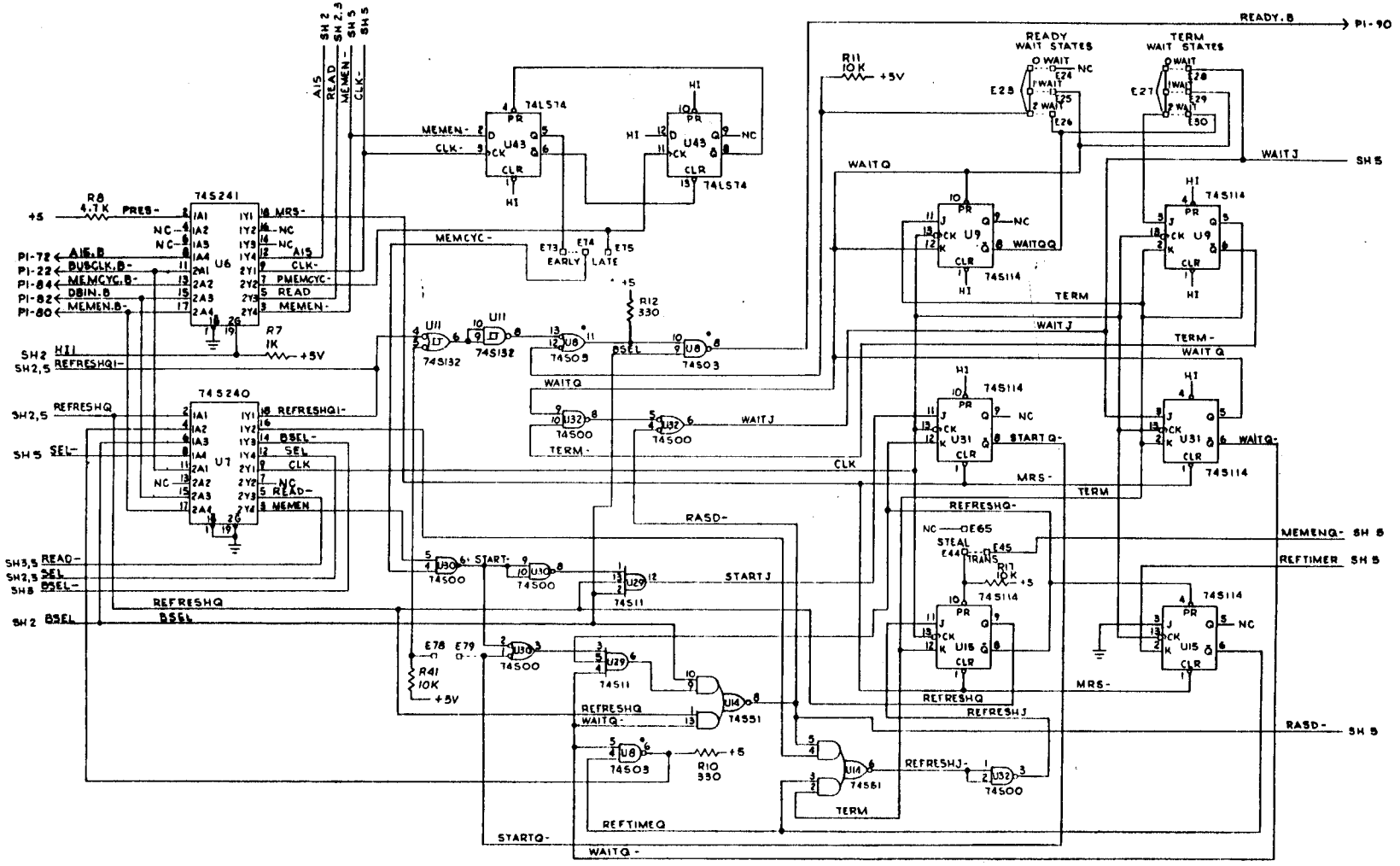


FIGURE 3-3. MAJOR STATE-CONTROLLER COMPONENTS

3.4.2 Parity Logic (Schematic Sheet 3)

The parity logic is shown in Figure 3-4. One bit of parity is generated by the parity generators/checkers (U20 & U27) during every memory write operation to the memory board. The parity bit is stored in a memory chip on the memory board. Odd parity is used; thus, a one is generated if an even number of one bits are found, and a zero generated if an odd number is found. When the same word is read from memory, parity is again generated and the parity bits compared with the bit stored during the write operation. If a parity error is found by the parity generators (U27 and U20), the board LED is lit and an interrupt request is generated to either INT.1- or INT.2- (jumper selectable at E7,E8, E9 with E7-E8 for INT1.B- or E7-E6 for INT2.B-) at the TMS 9901 via the system bus. By not inserting this jumper, the parity interrupt is disabled; however, the board parity error LED will be lit and remains on until reset through I/O port P7 at the microcomputer's TMS 9901 or through the CRU.

When a parity error is found during a memory access read operation, the error signal (PARITYERR) is gated through U29 and clocked into the parity error flip-flop (U3) by RASQ causing the parity error LED on the memory board to be lit; this also activates the interrupt signal. The interrupt service routine may clear the LED by a low signal sent to CLR of U3 via one of two jumper-selectable paths: through either (1) INT15-/P7 of the microcomputer board TMS 9901 or (2) via a CRU write instruction with address lines A4 to A10 containing the CRU hardware base address for the input to a user-supplied and -programmed 74S287 PROM at U5 (top of Figure 3-4). A low on the MSB of the four-bit PROM output will extinguish the LED; thus the specified address input to the PROM is programmed to output a $0XXX_2$ pattern on DO4 to DO1, and the other bits of the PROM are programmed high (or a one in the other DO4s output since DO1 to DO3 are not connected). The PROM remains enabled by both select lines tied low. LED disabling paths are selected by jumpering E5-E4 for disabling via INT15-/P7 (a zero at P7 of the TMS 9901) or E5-E6 for LED disabling through CRU addressing.

The CRU method of checking for parity error and for clearing the latched parity error LED allows parity error handling on several memory expansion boards within one system. It also frees system bus line INT15- (TMS 9901 P7). Such a parity error scheme requires having different CRU addresses for each board's PROM U5 (this PROM is supplied by the user). Note on the schematic that only address lines A4 to A10 go to this PROM; thus, only bits 4 to 10 of register 12 are sensed for the hardware CRU base address. When a parity error is detected via an interrupt, the boards can be polled by a TB instruction to the CRU addresses of each boards' PROM U5. A TB to the board issuing the interrupt will sense a one on the CRUIN line; the other boards will result in a zero sensed on CRUIN. Jump instructions (JEQ or JNE) then can be used to decide if a board did or did not cause the interrupt (JEQ will execute if a one is on CRUIN). To clear the error indication, writing (via either an SBO or SBZ instruction) to the PROM-U5 CRU address will cause a CRUCLK pulse, and the zero (from U5 DO4) at gate U2 will go to the CLR- pin of J-K flip-flop U3, causing Q- to go high, disabling the parity-error LED.

NOTE

The user should be aware that 16 bits of CRU space is required for each unique PROM address. Only address lines A3 to A10 are decoded by the 74S287 PROM at U5 (instead of A3 to A14). This means that the address on A11 to A14 does not affect the selection of the 4 bits output by the PROM. To program this PROM, see Appendix C.

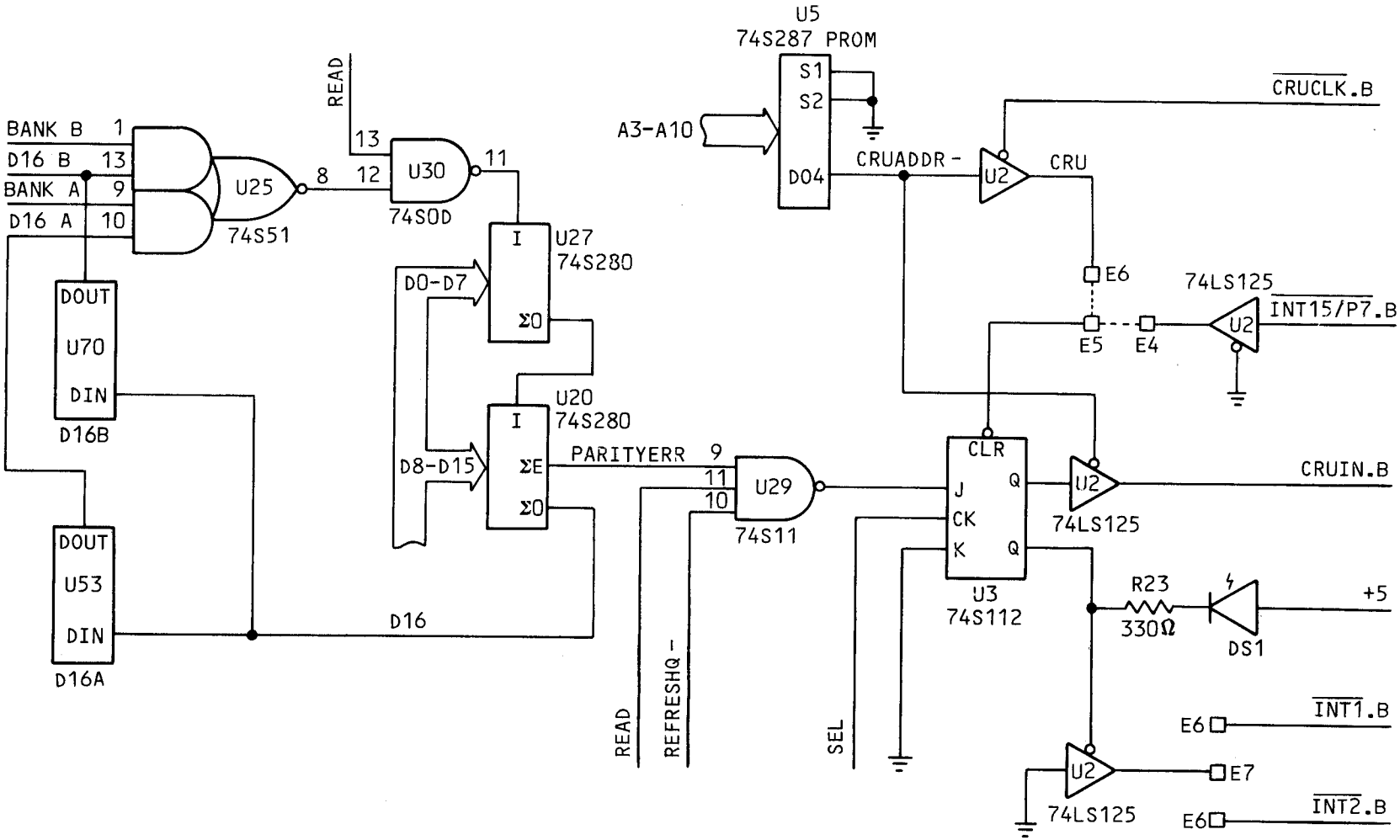


FIGURE 3-4. PARITY LOGIC

The following programming example is for a hypothetical system which contains two TM 990/203 modules. After a parity error interrupt, each of the two modules is polled through the CRU with the two PROM enabling addresses located at software base addresses (R12 contents) 0200_{16} and 0210_{16}. Once the interrupt service routine (ISR) has been called, it first must check to find which module issued the parity interrupt by monitoring the CRUIN line at the CRU address of each module. One of the final steps in the ISR is to reset the interrupt indicator of the module in error via the same CRU address that enables the U5 PROM. Any CRU output (CRUCLK-active) will reset the indicator.

```
INTRPT      (START OF INTERRUPT SERVICE ROUTINE)
       LI   R12,>0200     ADDRESS OF PROM U5 ON MODULE 1
       TB   0             INTERRUPT OCCUR ON MODULE 1?
       JEQ  RESET         YES, CORRECT, RESET THE PARITY ERROR
       LI   R12,>0210     ADDRESS OF PROM U5 ON MODULE 2
       TB   0             INTERRUPT OCCUR ON MODULE 2?
       JNE  END           IF NO INTERRUPTS, EXIT ROUTINE
RESET  SBO  0             RESET INTERRUPT
        .                 FIND LOCATION WITH ERROR
        .                 AND CORRECT PARITY BIT OR DATA
END    RTWP               RETURN
```

3.4.3 Data Multiplexing and Buffering

Figure 3-5 is a diagram of the data multiplexing and buffering section of the TM 990/203. Since the TMS 4027 memory does not have tri-state data outputs, it is necessary to use data multiplexers to select the correct memory bank.

Two signal pairs: READ and BANK A SEL or READ and BANK B SEL are ANDed through gate U19 to enable the outputs of the correct latches, while the outputs of the disabled latches are at high impedance. The data then is buffered through U21 and U28 to the system bus. Data being written to memory is gated directly to the memory chips from the buffers single only the bank which receives the correct RAS and CAS signals will accept data.

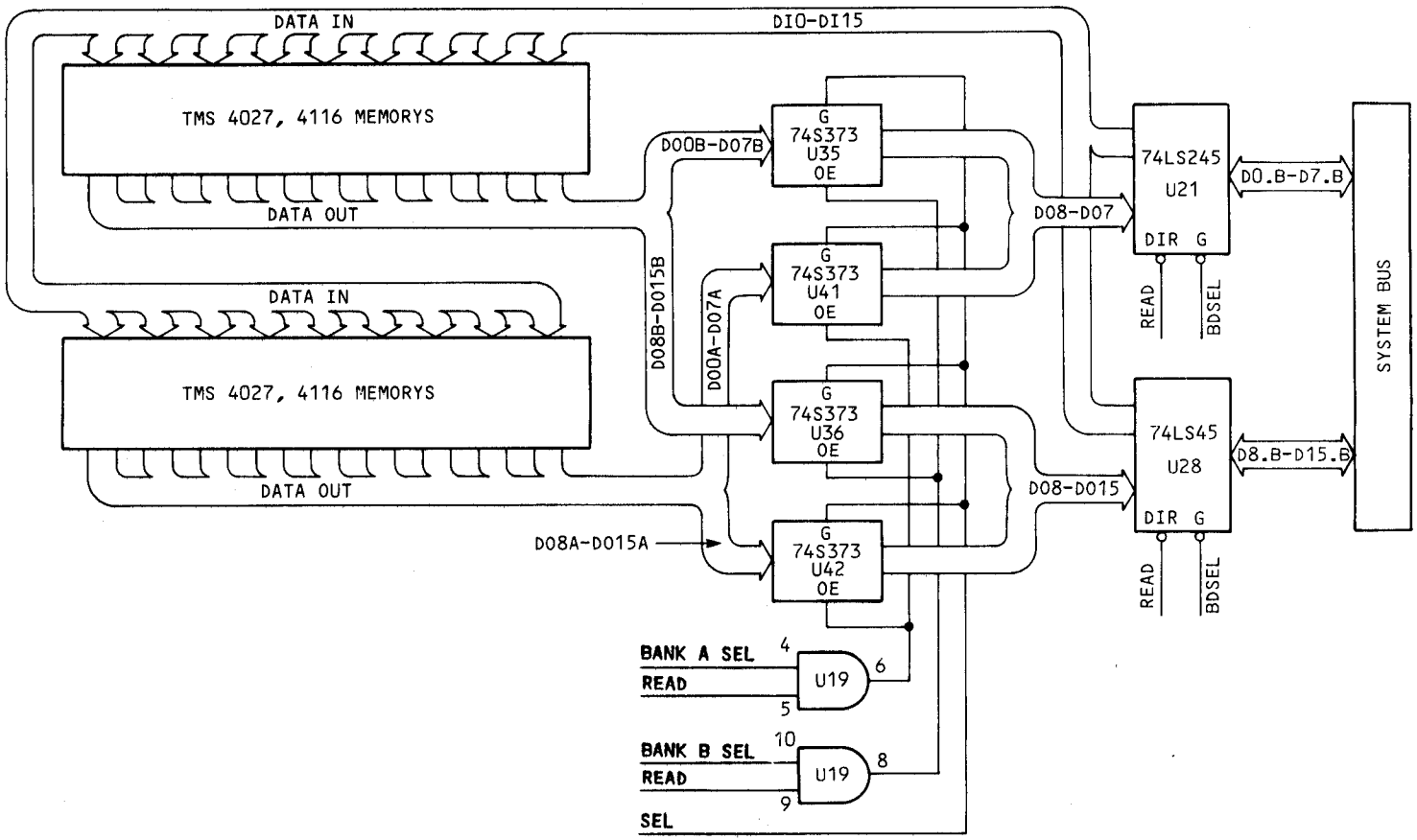


FIGURE 3-5. DATA MULTIPLEXING AND BUFFERING

3.4.4 Addressing and Select Logic (Schematic Sheet 4)

TM 990/203 memory boards can be part of a memory system containing up to one megabyte of memory depending upon the system processor. With 16-bit-address microcomputers (e.g., TM 990/100 or TM 990/101), only address lines A0 to A15 are used, and the /203 16/20-bit address jumpers should be in the 16-bit positions. When the /203 is used with a 20-bit address bus(i.e. address jumpers in the 20-bit position), the additional four most-significant bits are labelled XA0 to XA3. These four extra address bits become the most significant hexadecimal digit of the five-digit hexadecimal address. When using 16-bit-address processors, the address jumper should be in the 16-bit position and these four extended address lines are ignored; for systems with a 20-bit address, S1 and S3 will be set to the most significant binary values.

Memory-map selection on the TM990/203 is provided by four DIP 4 pole double throw DIP switches(Off=1 and On=0) which are used to designate the upper and lower memory address bounds of the populated memory area on the board. Switches S1 and S3 select the upper and lower bounds of the extended addresses (XA0-XA3). Switches S2 and S4 select the upper and lower bounds of the most significant "nibble" of the 16-bit address. The three least significant digits are by default FFE_{16} for the upper memory bound; and 000_{16} for the lower memory bound. The 16/20-bit address bus jumper causes the /203 to either ignore the extended address lines(E90 to E91 and E93 to E94) in the 16-bit position; or to decode the upper and lower bounds of the extended address lines(E89 to E90 and E92 to E93) in the 20-bit position.

For example, if the following switch settings are made for a 16-bit address system(address bus jumpers in the 16-bit position):

- S1 - don't care
- S2 to 4_{16} (0100_2 or ON-OFF-ON-ON)
- S3 - don't care
- S4 to D_{16} (1101_2 or OFF-OFF-ON-OFF)

then memory would be enabled from 4000_{16} to $DFFE_{16}$ (inclusive of these two addresses). Therefore, the use of the S1-S4 switch settings allows memory bounds to be set on any 4K or 2K word boundary. If the lower bound is set greater than the upper bound, the entire memory will be disabled.

In another example, if the following settings are made for a 16-bit address system:

- S1 and S3 - don't care
- S2 and S4 set to 0_{16} (0000_2 or all ON)

Memory would be enabled from 0000_{16} to $0FFE_{16}$.

Board select (BSEL) is determined by comparing the eight-most significant incoming address lines XA3.B to A3.B with the values set at switches S1 through S4. Figure 3-6 shows the memory select scheme of the TM 990/203. Switch S1 and S2 settings are magnitude compared (via U1 and U13) to the incoming address line value to determine if the memory address is equal to or higher than the low bound value at S1 and S2 (thus a valid incoming low address); if so, a low is sent to U25 where a similar value from comparitors U33 and U18 shows if the incoming address is equal to or lower than the upper bound set at S3 and S4. The 16/20 bit address bus jumpers allow the /203 to ignore or sense the extended address lines XA0 - XA3. In the 16-bit positions, the magnitude comparitors at U1 and U18 are effectively removed from the circuit, and the extended address bits are ignored. The switch settings on S1 and S3 are irrelevant in this case, and the switch settings on S3 and S4 are compared with the adress lines A0-A3 to determine BDSEL. In the 20-bit position the /203 also compares the switch settings on S1 and S2 with the address lines XA0-XA3 when determining BDSEL. If the address is within the upper and lower bounds, BDSEL becomes active (high) at U25-6.

Refer to Figure 3-6 to determine the Memory Bounds switch settings.

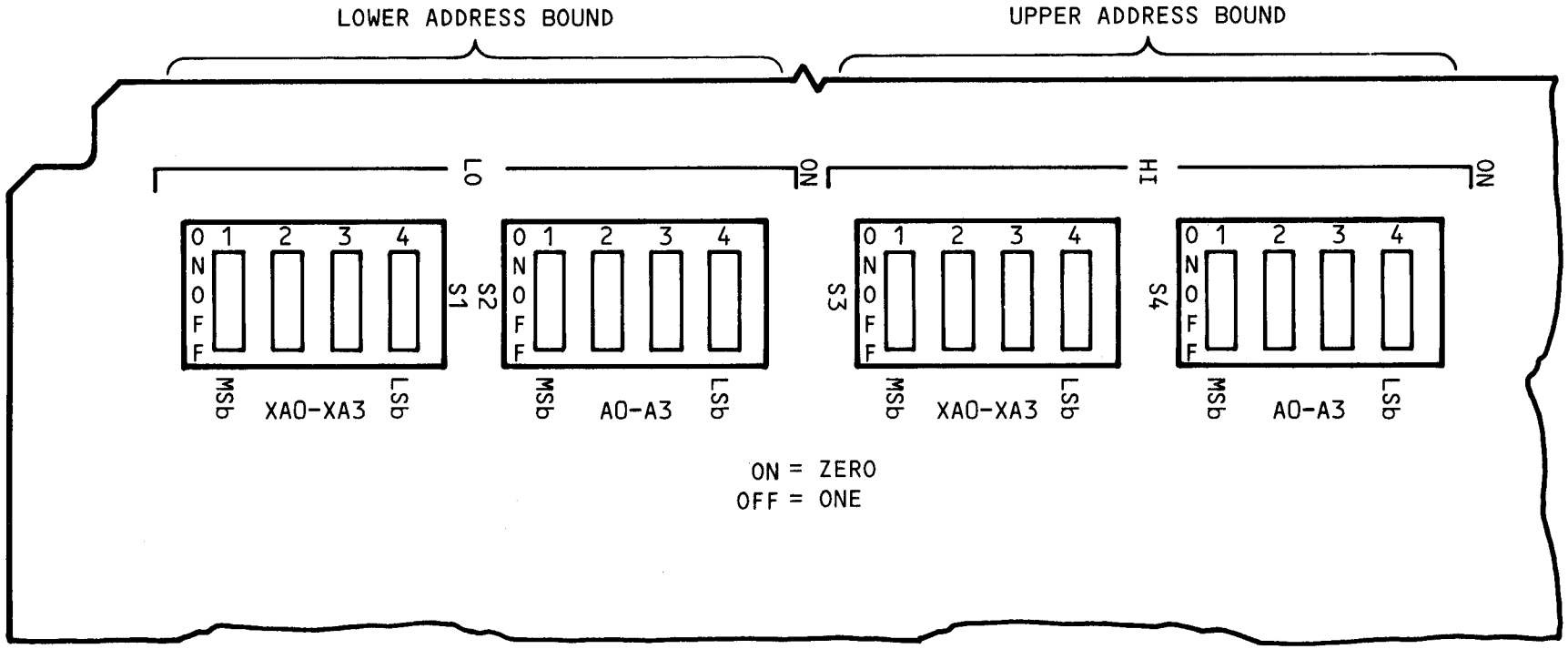


FIGURE 3-6. MEMORY CONFIGURATION SWITCHES

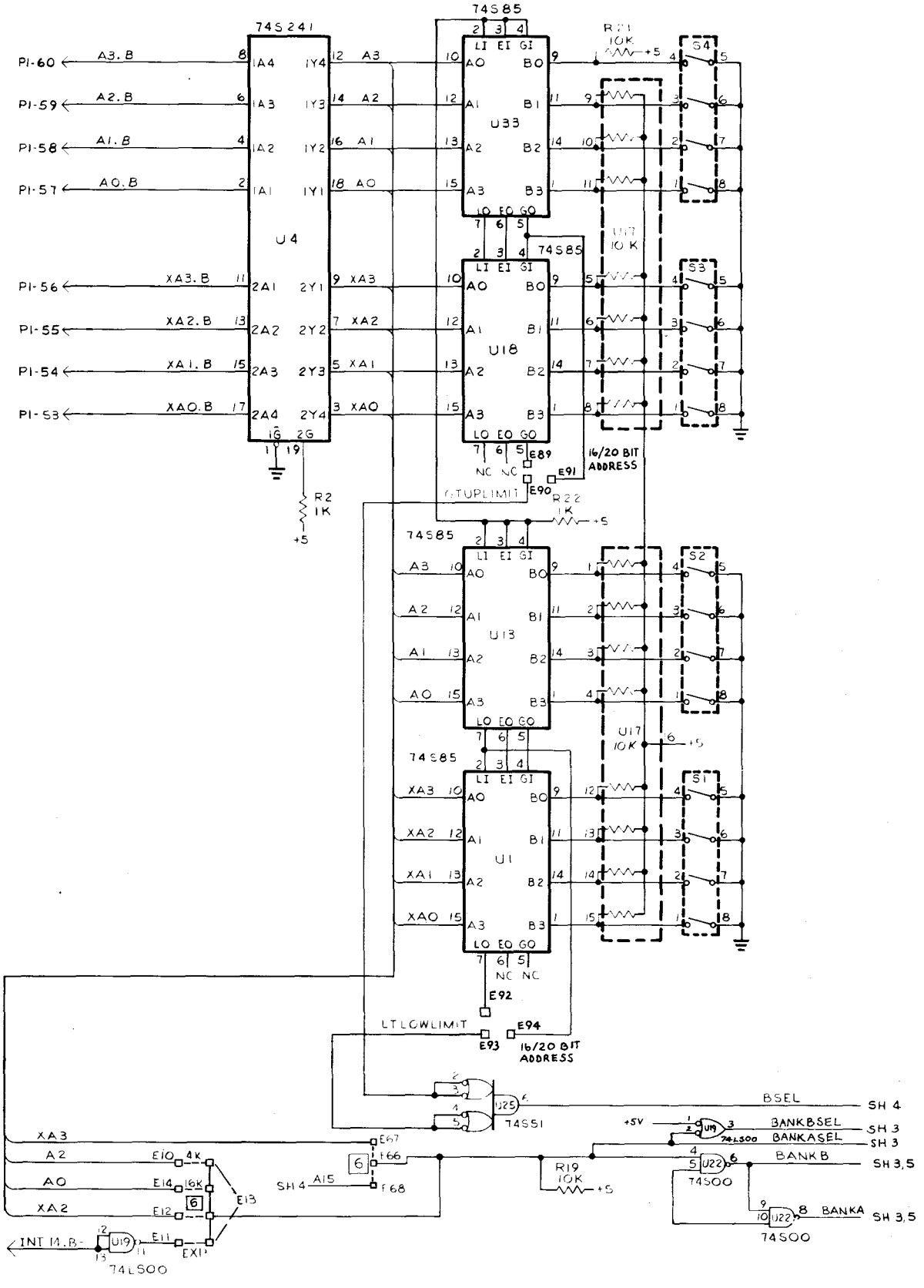


FIGURE 3-7 MEMORY SELECT SCHEME FOR THE TM 990/203

Figure 3-7 illustrates the addressing logic aboard the TM 990/203. U37 is a counter that generates the refresh row addresses, and U38 is the buffer that drives the address to the memory devices during a refresh cycle. The address is incremented at the end of the refresh cycle, ready for the next refresh. The counter ensures that all row addresses are accessed.

U24 and U23 are buffers used to multiplex the memory addresses from the processor to the RAM. During processor memory cycles, U24 enables the row address, and U23 enables the column addresses.

The row and column addresses for the memory chips are set up by two 74LS244 octal buffers (U23 and U24) which strobe the row and column address to the memory chips. Table 3-1 lists which processor address lines comprise the row and the column address inputs for the two different memory chips that can be used on the TM 990/203. Figure 3.7(a) depicts the row counter and buffering logic for the memory refresh addresses; Figure 3-7(b) depicts the row and column generation for memory access. Devices U23 and U24 select first the row and then the column address, selected first by RASADDR- then CASADDR-.

TABLE 3-1. MEMORY ADDRESS GENERATION

Memory Address Signal		Memory Size	
		4 K (TMS 4027)	16 K (TMS 4116)
Row Address	MA0	A6	A6
	MA1	A5	A5
	MA2	A4	A4
	MA3	A3	A3
	MA4	A8	A7
	MA5	A7	A2
	MA6	-	A1
	MA7	-	-
Column Address	MA0	A14	A14
	MA1	A13	A13
	MA2	A12	A12
	MA3	A11	A11
	MA4	A10	A10
	MA5	A9	A9
	MA6	-	A8
	MA7	-	-
Bank Select	Bank B	A2	A0
	Bank A	A2	A0

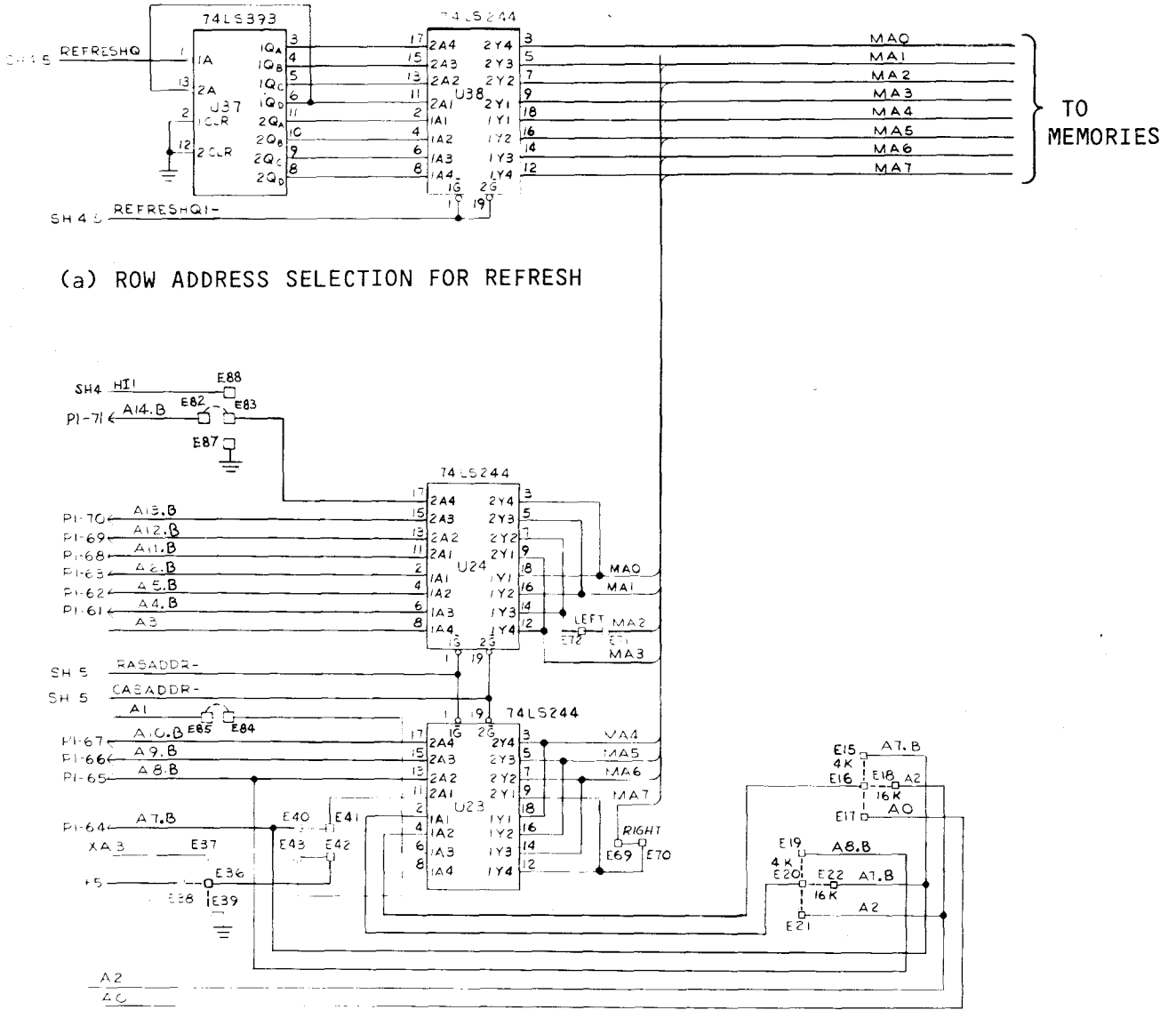
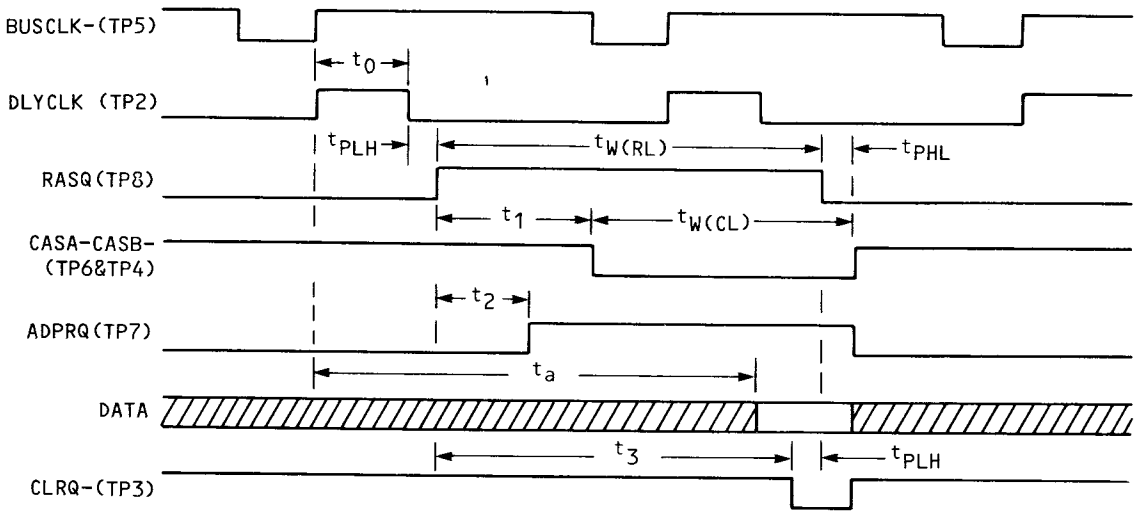


FIGURE 3-8. ADDRESS LOGIC

3.4.5 Read/Write Timing

Figure 3-8 shows the various timer delays and details maximum and minimum timing tolerances on the TM 990/203 board.

Figures 3-9 and 3-10 show the basic read and write operations within the specified temperature range on the board. Figure 3-9 shows a read operation with (1) no wait state (left side of the figure) and (2) one wait state (right side of the figure). Figure 3-10 contains timing for read operations with similar wait states shown.



Times in Nanoseconds (ns)

	MIN	TYP	MAX
t_0	126	140	154
t_1	70	80	90
t_2	40	50	60
t_3	318	333	350
$t_{W(RL)}$	255	287	319
$t_{W(CL)}$	165	212	258
t_a	376	395	418

t_{PHL} & t_{PLH} 3 to 13.5 nanosecond propogation delays

FIGURE 3-9. SYSTEM TIMING SHOWING RAS AND CAS DELAY

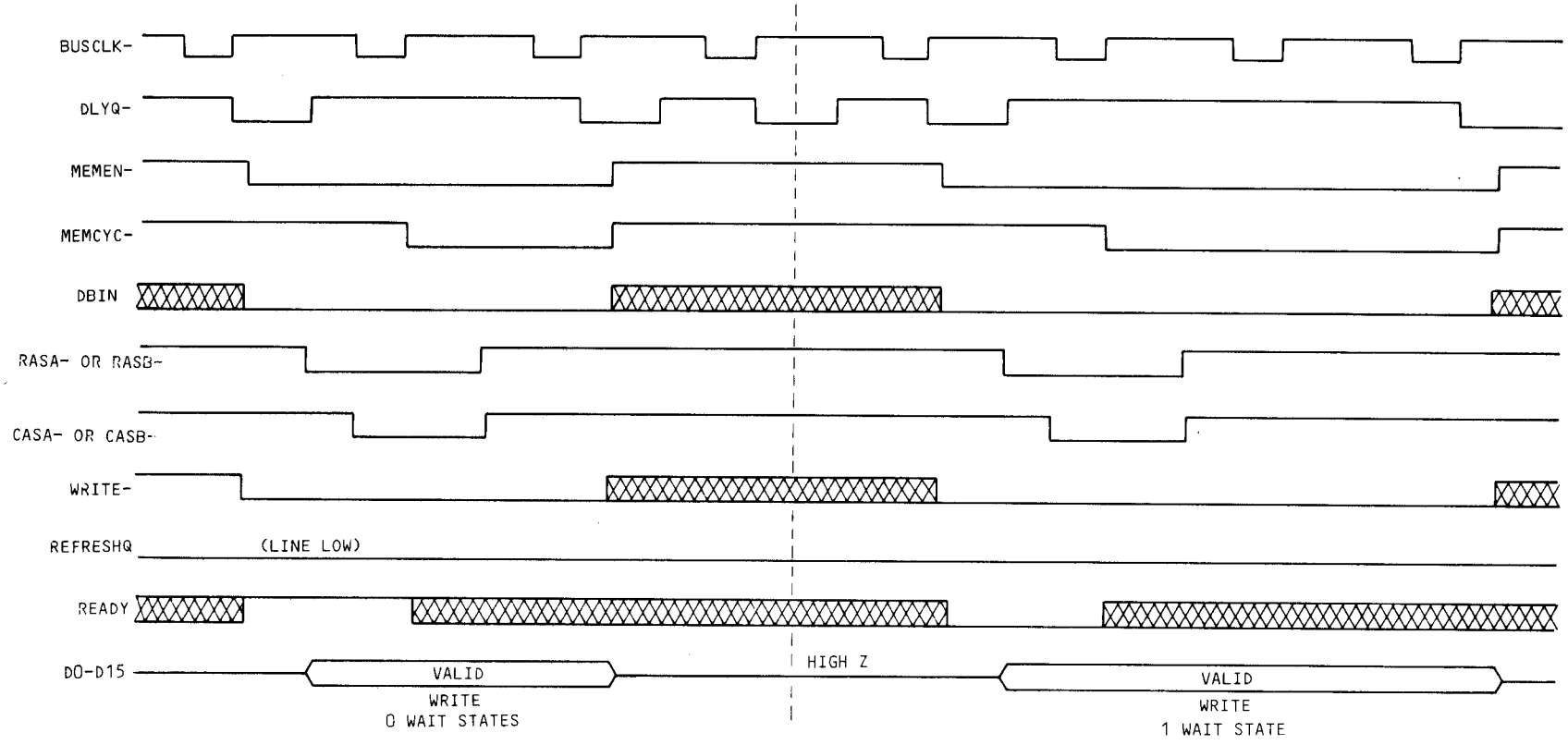


FIGURE 3-10 MEMORY WRITE OPERATION TIMING DIAGRAM

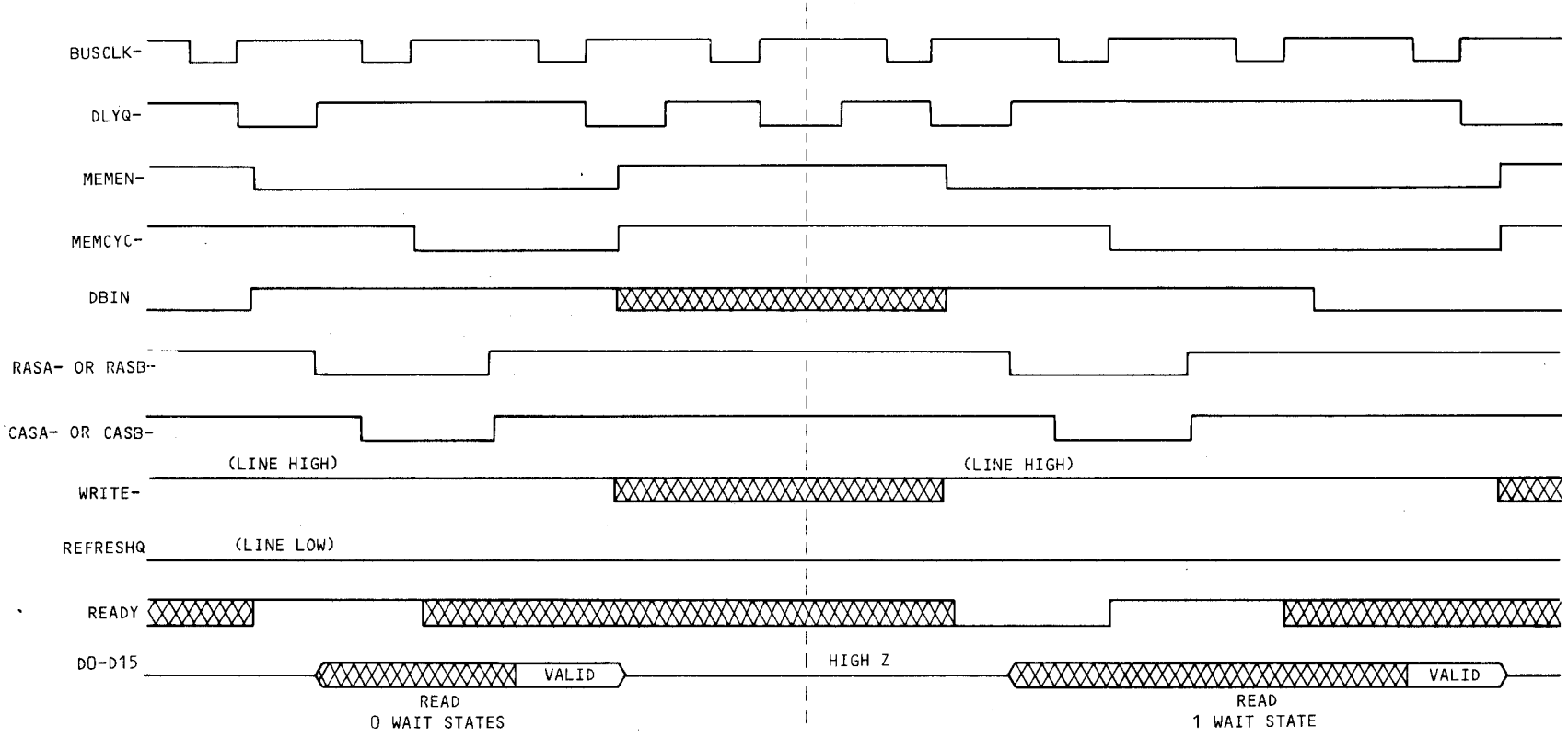


FIGURE 3-11. MEMORY READ OPERATION TIMING DIAGRAM

APPENDIX A

TM 990/203 JUMPER DESCRIPTIONS

The TM 990/203 was designed to fill the memory requirements of current TM 990 systems as well as to provide the flexibility to accommodate future system enhancements.

The jumpers are described here in order to provide an overview of the module's versatility. This is not intended as a guide to modify the module but an aid for the more experienced design engineer so that debugging and reconfiguration tasks can be simplfied.

Table A-1 provides a listing of the variations that are available by using Jumpers E1 through E75. Table A-2 provides a listing showing each possible jumper position and a description of the variation/configuration that is made possible by this connection. In some cases, the etch connecting certain pins must be cut - the user is informed when this is the case. All pins are called out by the schematic diagram sheet where they are located.

CAUTION

Cutting etch or soldering jumper wires to a TM 990 module will void the factory warranty.

TABLE A-1 JUMPER DESCRIPTIONS

PINS	DESCRIPTION
E1 to E2 or E3	Configures power to the memory chips being used.
E5 to E4 or E6	Determines which signal is used to reset the parity interrupt generated on the TM 990/203.
E8 to E7 or E9	Determines which interrupt level (INT1 or INT2)is to be used for a parity interrupt.
E13 to E10, E11, E12, or E14 E66 to E67 or E68	Controls which bank of memory is selected when more than one bank is populated. If only one bank is populated, it must be bank A. The jumper from E13 must be removed to select bank A. *
E16 to E15, E17, or E18 E20 to E19, E21, or E22	Configure the address required for various RAM sizes to meet the address requirements for multiplexed addressing.
E23 to E24, E25, or E26 E27 to E28, E29, or E30	Configure wait states for systems using the "READY" signal for memory timing control. These jumpers must be set in the same position for proper timing
E31 to E32, E33, E34, or E35 E59 to E60 or E61	Determine the number of refresh cycles performed during a 1 millisecond period.*
E36 to E37, E38, or E39	Select for portion of future RAM that is good.
E40, E41, E42, E43	For future expansion
E45 to E44 or E65	Selects the type of refresh to be accomplished.
E47 to E46 or E48	Configures the chip selects for RAM.
E50 to E49 or E51	Configures pin 9 of RAM devices
E53 to E52 or E54	Configures parity for byte or word modes
E55 to E56	Used in conjunction with jumpers E49-E51. Capacitors are added to filter pin 9 (+ 5 V Vcc

(CONTINUED)

* If the E13 jumper group is used, the E66 jumper group should have no jumpered connections and vice versa. Likewise, if the E31 jumper group is used, the E51 jumper group should have no jumpered connections and vice versa.

TABLE A-1 JUMPER DESCRIPTIONS (CONCLUDED)

PINS	DESCRIPTION
E57 to E58	Jumper when connected capacitors (E55 to E56) are added to the circuit.
E59,E60, E61	See E31-E35.
E62 to E63 or E64	Selects write control for the memories
E65	See E44-E45
E66,E67, E68	See E10-E14
E69, E70, E71,E72.	Configures the MSB of the address entered to the physical MSB of the device in order to partition the portion of RAM that is good in the case of a partially good RAM.
E74 to E73 or E75	Memory cycle signal select
E78 to E79 or open	Custom/Normal operation (Normal operation = open)
E83 to E82,E87,E88	For:TMS 4027/4116;TMS4108-25-0;or TMS4108-25-1 memories
E82 or E85 to E84	For TMS4108 memories (wire wrap wire), /4027 or /4116 memories
E90 to E91 or E89 E93 to E94 or E92	16-bit address bus (TM 990/100,/101) or 20-bit address bus system

TABLE A-2. JUMPER CONECTION DESCRIPTIONS

Jumper Connection	Description	Schematic Sheet	Cut Etch?
E1 to E2	Supplies +12 volts to pin 8 of memory devices TMS 4027 or TMS 4116.	1	Y
E1 to E3	Jumpered to supply +5 volts to pin 8 of future memory devices	1	N
E5 to E4	In this position the parity interrupt is reset by backplane signal INT15.B-/P7. A low output on this line will clear and disable the parity interrupt. A high on this line will enable the parity interrupt.	2	N
E5 to E6	A parity interrupt is reset by executing a CRU output (zero or one) to a particular location. The particular location is programmed into a SN74S287N PROM by the user. This allows the user to free INT15.B- for other purposes. It also allows the user to configure his parity reset in the CRU address map. Up to 16 different addresses can be programmed simultaneously	2	N
E8 to E9	Configures the TM 990/203 parity interrupt to level 1 via backplane signal INT1.B-.	2	N
E8 to E7	Configures the parity interrupt to level 2 via INT2.B- on the backplane.	2	N
No Jumper	Interrupts disabled.		
E13 to E10	Jumper when 4K DRAMs are used.	3	N
E13 to E14	Jumper when 16 K RAMs are used.	3	N
E13 to E11	Jumper when memory expansion is used. This feature allows a swap of Bank A and Bank B memory. A CRI output to bit 8 of the TMS 9901 on the processor module swaps access to Bank A or Bank B. In this way the user can expand his memory by swapping memory Banks. A low output to bit 8 enables Bank B and a high output enables Bank A. The memory bounds switches should not be set for more memory than there is in one bank.	3	N
No Jumper	Select Bank A memories only	3	N
E13 to E12	For future expansion.	3	N

(CONTINUED)

TABLE A-2. JUMPER CONECTION DESCRIPTIONS(CONTINUED)

Jumper Connection	Description	Schematic Sheet	Cut Etch?
E66 to E68	Should be jumpered in byte mode. Both Banks A and B should be populated in this case.*	3	N
E66 to E67	For future expansion	3	N
E16 to E15	Jumper for 4 K RAMs.	2	N
E16 to E18	Jumper for 8K or 16 K RAMs.	2	N
E16 to E17	For future expansion.	2	N
E20 to E19	Jumper for 4 K RAMs.	2	N
E20 to E22	Jumper for 8K or 16 K RAMs.	2	N
E20 to E21	For future expansion.	2	N
E23 to E24	0 memory access wait states.	4	N
E23 to E25	1 memory access wait states.	4	N
E23 to E26	2 memory access wait states.	4	N
E27 to E28	0 memory access wait states.	4	N
E27 to E29	1 memory access wait states.	4	N
E27 to E30	2 memory access wait states.	4	N
E31 to E32	256 cycles/ms	5	N
E31 to E33	128 cycles/ms	5	N
E31 to E34	64 cycles/ms	5	N
E31 to E35	32 cycles/ms	5	N
E59 to E60	0 cycles per 1 ms (test position).	5	N
E59 to E61	In TM 990 system this pin is ground and allows external refreshes (under control from P1-91 on backplane of card cage). A one on this pin forces a refresh cycle. The one must be valid for one positive transition of BUSCLK.B-.	5	N
E36 to E37	Jumper for totally good memory devices.	2	Y
E36 to E38	Most significant row or column has to be high for good operation.	2	Y
E36 to E39	Most significant row or column has to be tied low for selection of portion of RAM that is good.	2	Y

(CONTINUED)

* If the E13 jumper group is used, the E66 jumper group should have no jumpered connections and vice versa. Likewise, if the E31 jumper group is used, the E51 jumper group should have no jumpered connections and vice versa.

TABLE A-2. JUMPER CONECTION DESCRIPTIONS (CONCLUDED)

Jumper Connection	Description	Schematic Sheet	Cut Etch?
E40 to E41	Standard configuration.	2	Y
E40 to E43	Partition RAM memory.	2	Y
E42 to E43	Standard configuration.	2	Y
E42 to E41	Partition RAM memory.	2	Y
E45 to E44	Transparent refresh with cycle steal upon refresh violation.	4	N
E45 to E65	Cycle steal refresh.	4	N
E47 to E46	Jumper for use of 4K memory devices.	3	N
E47 to E48	Jumper for use of any other.	3	N
E50 to E49	Jumper for those RAM devices requiring +5 volts on pin 9 (4K and 16K).	3	Y
E50 to E51	For future expansion.	3	Y
E53 to E52	Jumper for parity on 16 bits or entire word.	3	Y
E53 to E54	Jumper for parity on 8 bits, bits D0-D7.	3	Y
E55 to E56	Used in conjunction with jumpers E49-E51. In cases where pin 9 of RAM is Vcc (+5V) capacitors are added to filter this supply.	3	Y
E57 to E58	When connected capacitors are in the circuit.	3	Y
E62 to E63	Standard configuration.	5	Y
E64 to E65	Allows off-board control.	5	Y
E69 to E70	Standard configuration.	2	Y
E71 to E72	Standard configuration.	2	Y
E69 to E72	Partition RAM memory.	2	Y
E70 to E71	Partition RAM memory.	2	Y
E74 to E73	For systems having early MEMCYC- signal.	4	Y
E74 to E75	For systems utilizing standard MEMCYC- signal (signal comes during 2nd memory clock cycle).	4	Y
E78 to E79	Custom application	3	N
open	Normal application		
E83 to E82	For use when TMS 4027 or TMS 4116 memories are present	3	N
E83 to E87*	For TMS 4108-25-0 memories		
E83 to E88*	For TMS 4108-25-1 memories		
E84 to E82**	For TMS 4108 Memories(Jumper wire)	2	N
E89 to E90 & E92 to E93	20-bit address bus (extended addressing)	2	N
E91 to E90 & E94 to E93	16-bit address bus (TMS990/100,101)	2	N

** E83 must be in TMS 4108 position
* E84 to E82 Jumper must also be in place

APPENDIX B

SCHEMATICS

NOTES: UNLESS OTHERWISE SPECIFIED:

1. CAPACITANCE VALUES ARE IN MICROFARADS
2. RESISTANCE VALUES ARE IN OHMS
3. ALL RESISTOR ARE .25 W, 5%
4. E59, E60, & E61 ARE USED FOR TEST ONLY
5. RESISTOR VALUES TO BE SELECTED AT UNIT TEST
6. FOR FUTURE EXPANSION
7. USER OPTION
8. -0001 ASSY USES TMS 4027-25. POPULATED AT U45 THRU U78.
   -0002 ASSY USES TMS 4116 POPULATED AT U45 THRU U61.
   -0003 & -0006 ASSY USES TMS4116 POPULATED AT U45 THRU U78.
   -0004 ASSY (MP9703-1) USES TMS 4116-15 POPULATED AT U45 THRU U61
   -0005 ASSY (MP9703-2) USES TMS4116-15 POPULATED AT U45 THRU U78
   -0007 ASSY USES TMS4108-25, POPULATED AT U45 THRU U61
9. FOR TMS 4108-25-0, JUMPER E83, E87
   FOR TMS 4108-25-1, JUMPER E83, E88
10. TM990/203 ASSEMBLIES ARE REPRESENTED ON SHEETS 1 THRU 5 OF THIS DIAGRAM
    MP9703 ASSEMBLIES ARE REPRESENTED ON SHEETS 1 & 2A THRU 5A OF THIS DIAGRAM
    MP9705 ASSEMBLY IS REPRESENTED ON SHEETS 1 & 2B THRU 5B OF THIS DIAGRAM
11. WIRE WRAP WIRE CONNECTION, NO JUMPER PLUG ON E85

JUMPER CONFIGURATION

MP9705	MP9703-1	MP9703-2	/203-21	/203-21A
E5, E6	E5, E6	E5, E6	E4, E5	E4, E5
E7, E8	E7, E8	E7, E8	E7, E8	E7, E8
E13, E14	———	E13, E14	E10, E13	E10, E13
E16, E18	E16, E18	E16, E18	E15, E16	E16, E18
E20, E22	E20, E22	E20, E22	E19, E20	E20, E22
E23, E24	E23, E24	E23, E24	E23, E24	E23, E24
E27, E28	E27, E28	E27, E28	E27, E28	E27, E28
E59, E61	E59, E61	E59, E61	E31, E35	E31, E35
E45, E65	E45, E65	E45, E65	E45, E65	E45, E65
E47, E48	E47, E48	E47, E48	E46, E47	E47, E48
E73, E74	E73, E74	E73, E74	E73, E74	E73-E74
E78, E79	E78, E79	E78, E79	E82, E83	[9]
E82, E83	E82, E83	E82, E83	E84, E85	E84, E82 [11]
E84, E85	E84, E85	E84, E85	E90, E91	E90, E91
E89, E90	E89, E90	E89, E90	E93, E94	E93, E94
E92, E93	E92, E93	E92, E93	———	———

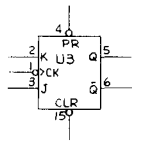
JUMPER CONFIGURATION

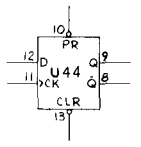
/203-22	/203-23
E4, E5	E4, E5
E7, E8	E7, E8
———	E13, E14
E16, E18	E16, E18
E20, E22	E20, E22
E23, E24	E23, E24
E27, E28	E27, E28
E31, E34	E31, E34
E45, E65	E45, E65
E47, E48	E47, E48
E73, E74	E73, E74
E82, E83	E82, E83
E84, E85	E84, E85
E90, E91	E90, E91
E93, E94	E93, E94

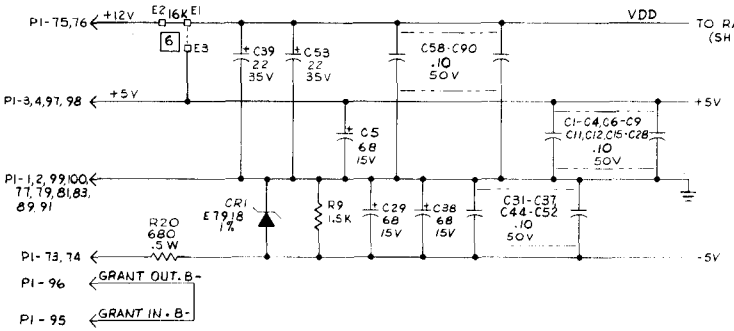
REFERENCE DESIGNATORS	
USED	NOT USED
C1 - C94	C55, C10, C30, C40, C43 C56
CR1	
DS1	
P1	
R1 - R41	R3 - R6
S1 - S4	
U1 - U18	U34
TP1 - TP10	
E1 - E94	E80, E81, E86

DEVICE TYPE	PIN NO. +5V	PIN NO. GND
74S00	14	7
74S03	14	7
74LS00	14	7
74S09	14	7
74S11	14	7
74S51	14	7
74S74	14	7
74LS74	14	7
74S85	14	7
74S112	16	8
74S114	14	7
74LS125	14	7
74S132	14	7
74S240	20	10
74S241	20	10
74LS244	20	10
74LS245	20	10
74S280	14	7
74S373	20	10
74LS393	14	7
SOCKET (XU5)	16	8

SPARES







REV STATUS	REV	E	E	E	E	E	E	E	E	E	E	E	E
OF SHEETS	SH	1	2	3	4	5	2A	3A	4A	5A	2B	3B	4B	5B

SEQ NO	IDENT	F SPEC	NO	ADDITIONAL	NOTES
	PROCESS		CLASSIFICATION		

PROCESSES — FOR CORRELATION TO PRINTING SPECIFICATIONS, SEE TI DRAWING 1048767

1600012	8111
1600011	8117
NEXT ASSY	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED
- DIMENSIONS ARE IN INCHES
- TOLERANCES ANGLES ± 1°
  3 PLACE DECIMALS ± .010
  2 PLACE DECIMALS ± .02
- INTERPRET DRAWING PER MIL-D-1000
- REMOVE ALL BURRS AND SHARP EDGES
- CONCENTRICITY MACHINED DIAMETERS .010 TIR
- DIMENSIONAL UNITS APPLY BEFORE PROTEXES
- PARENTHETICAL INFO FOR REF ONLY

HOLE TOLERANCE

PARTS LIST

ITEM NO	QTY	PART OR IDENTIFYING NUMBER	NOMENCLATURE OR DESCRIPTION	PROCUREMENT SPECIFICATION	NOTES

DWN 11/16/78
ENGR 2/9/79
CHK 3/6/79
QA 3/6/79
MFG 3/7/79

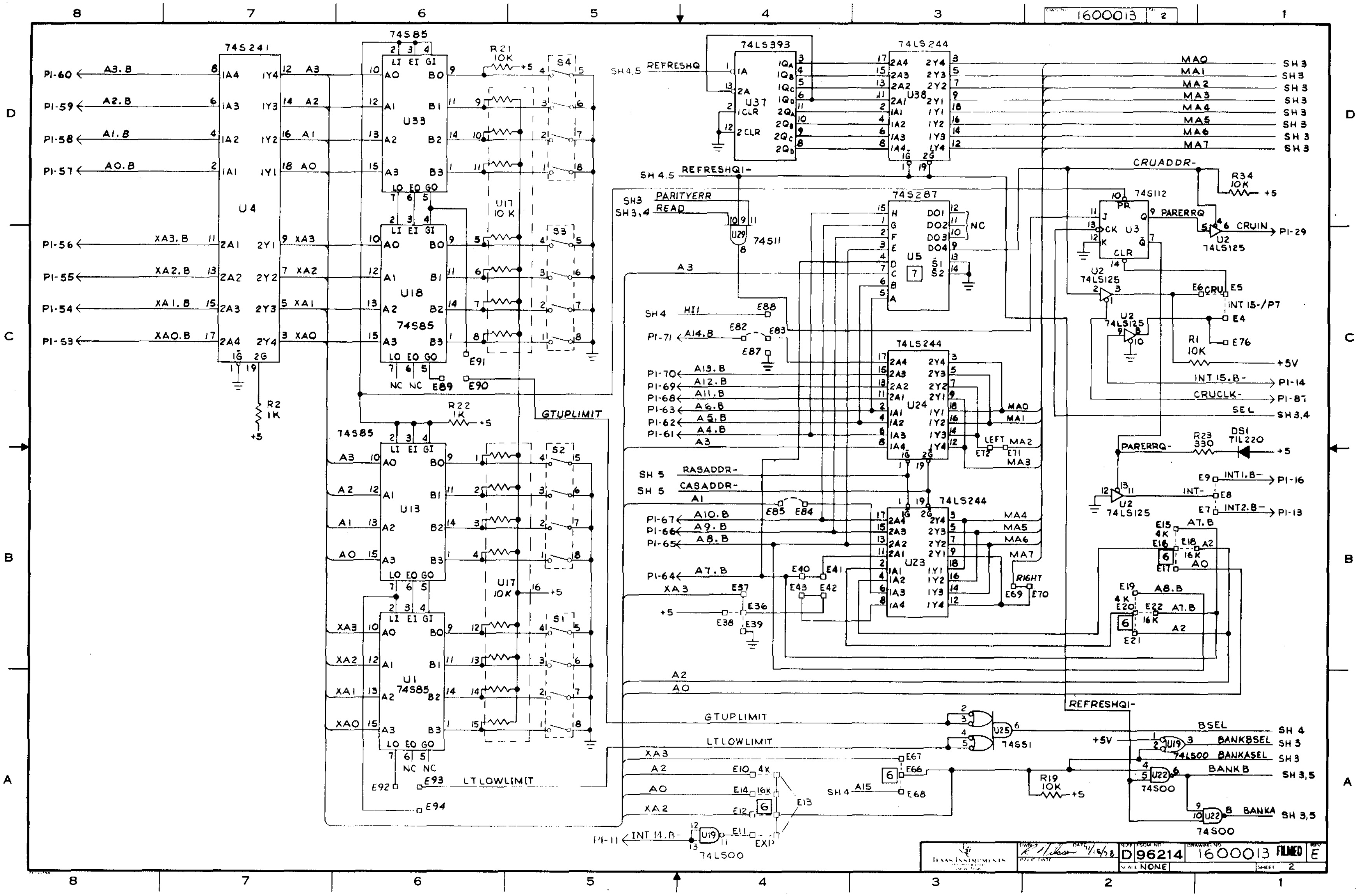
TEXAS INSTRUMENTS INCORPORATED
DALLAS, TEXAS

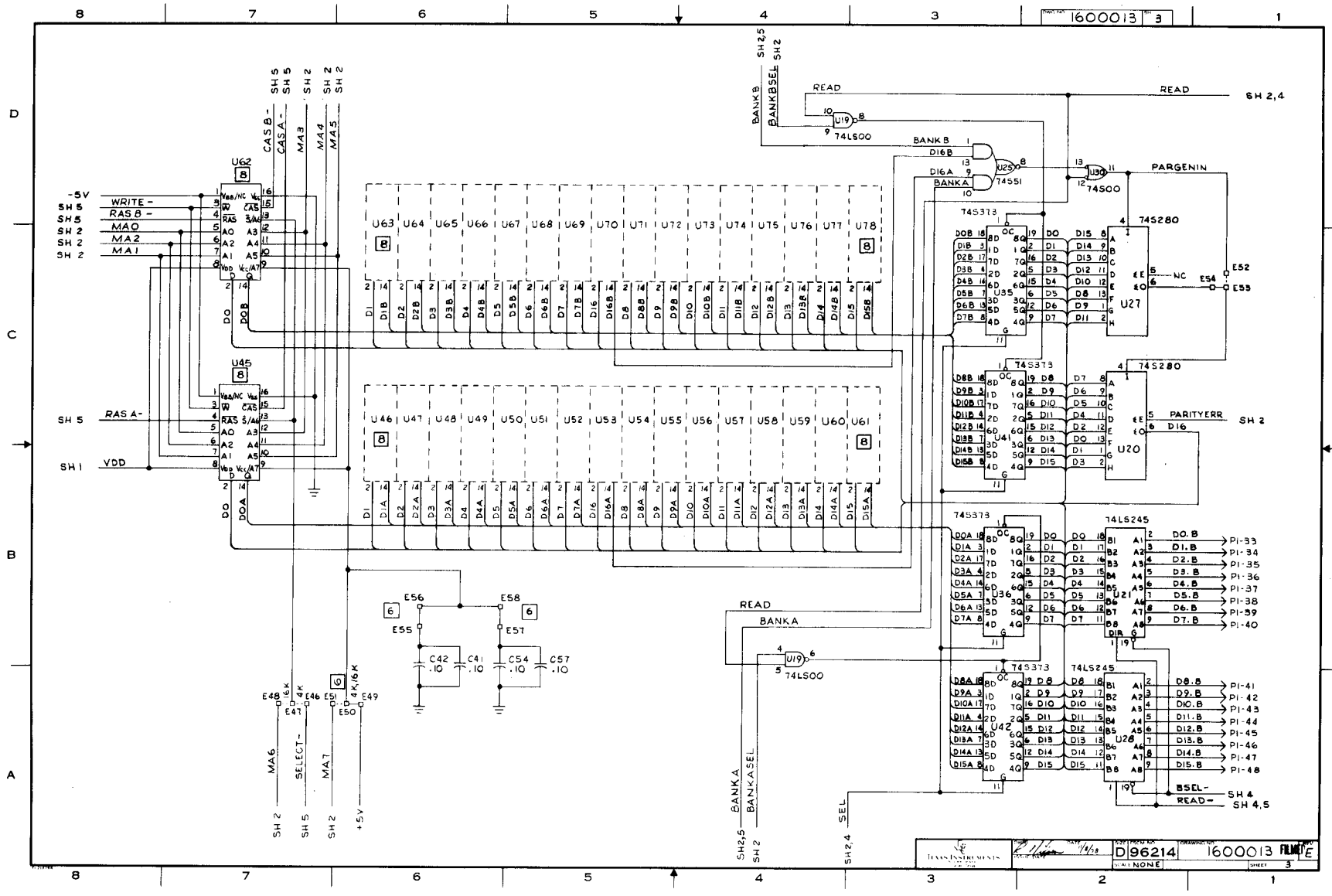
DIAGRAM, LOGIC, TM 990/203

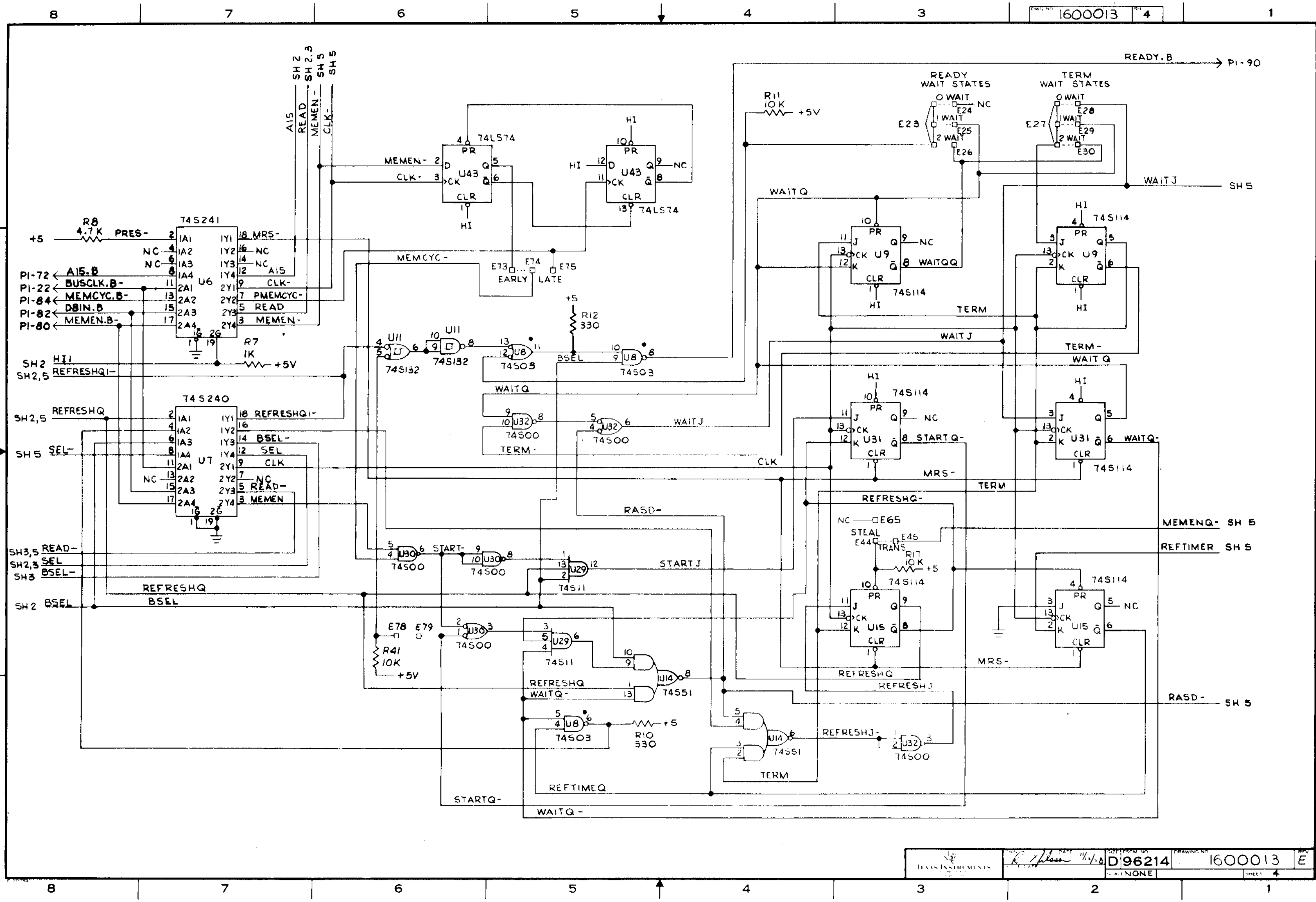
NEXT ASSY D 96214 | DRAWING NO 1600013 | FILMED

SCALE NONE | SHEET 1 OF 13

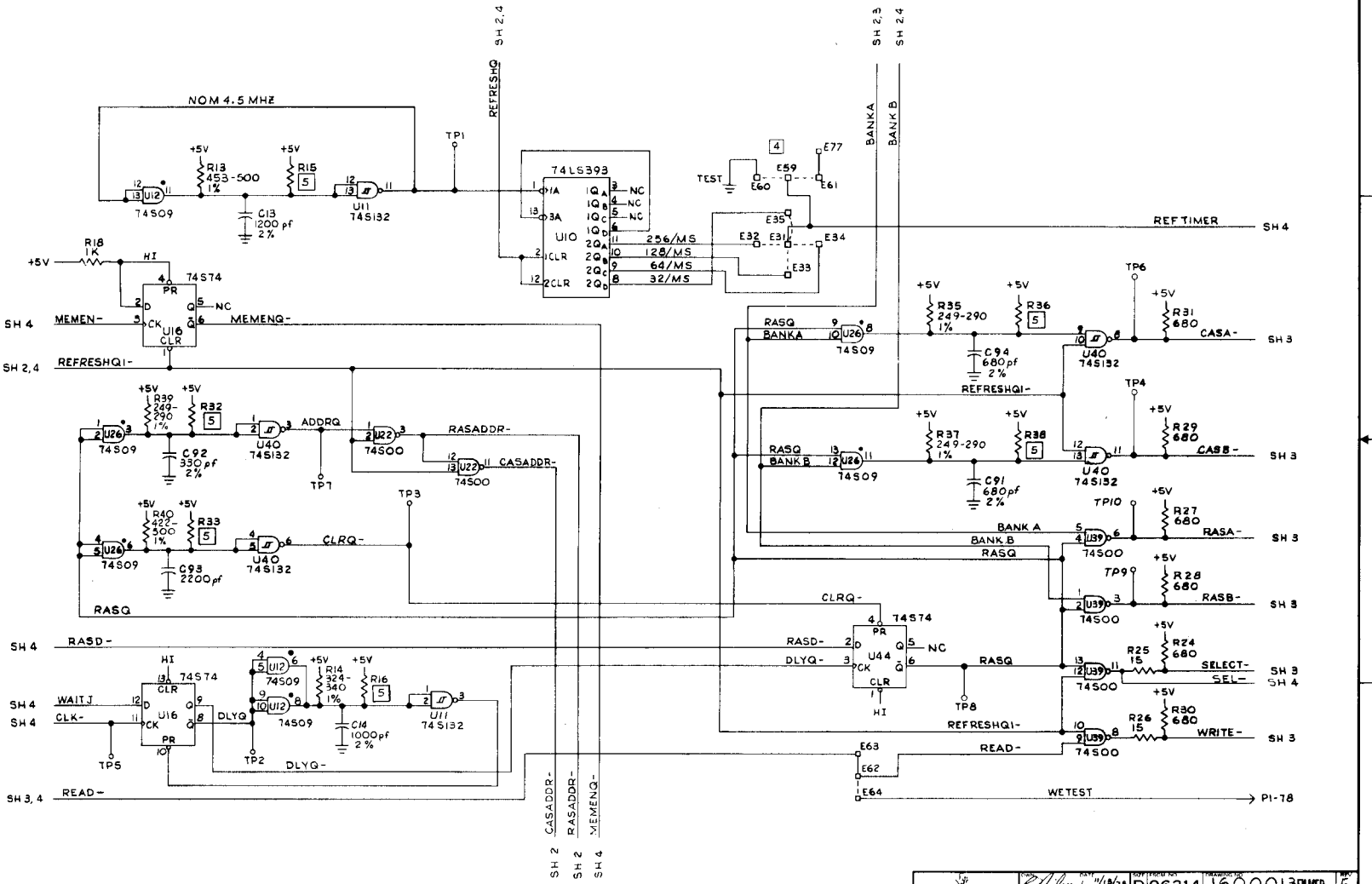
REVISIONS			
REV	DESCRIPTION	DATE	APPROVED
A	CN45153 (INFORMAL) 1) UPDATE TO AGREE WITH REV C PWB 1600010	8/27/79	
B	CN447324	8/7/79	
C	CN454360 1) ADDED MP9703 DOCUMENTATION, SHEETS 2A THRU 5A	1/24/80	
D	CN454388 ADDED MP9705 DOCUMENTATION, SHEETS 2B THRU 5B	8/3/80	
E	CN471748		


1600013
74S241
U4
74S85
U33
U18
U13
U1
U17 10K
R21 10K
R22 1K
R2 1K
S1
S2
S3
S4
GTUPLIMIT
LTLOWLIMIT
74LS393
U37
74LS244
U38
REFRESHQ
REFRESHQI-
PARITYERR
READ
U29 74S11
74S287
U5
U24
U23
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7
RASADDR-
CASADDR-
CRUADDR-
R34 10K
74S112
U3
PARERRQ
CRUIN
U2 74LS125
INT 15-/P7
R1 10K
INT15.B-
CRUCLK-
SEL
PARERRQ-
R23 330
DS1 TIL220
INT1.B-
INT2.B-
LEFT
RIGHT
U25 74S51
BSEL
BANKBSEL
BANKASEL
BANKB
BANKA
U19 74LS00
U22 74S00
R19 10K
INT14.B-
EXP
TEXAS INSTRUMENTS
D 96214 1600013 FILMED E
SCALE NONE
SHEET 2


1600013 3
U62
U45
U63 U64 U65 U66 U67 U68 U69 U70 U71 U72 U73 U74 U75 U76 U77 U78
U46 U47 U48 U49 U50 U51 U52 U53 U54 U55 U56 U57 U58 U59 U60 U61
-5V
WRITE -
RAS B -
MA0
MA2
MA1
RAS A -
VDD
CASB -
CASA -
MA3
MA4
MA5
SH 5
SH 2
READ
BANKB
BANKA
BANKSEL
SH2,5
SH 2,4
PARGENIN
74S373
74S280
74LS245
74LS00
74S51
U35
U41
U36
U42
U27
U20
U21
U28
U19
U25
U30
PARITYERR
D16
NC
E52
E54
E53
E56
E58
E55
E57
C42 .10
C41 .10
C54 .10
C57 .10
E48 16K
E47 4K
E46
E51
E50
E49 4K/16K
MA6
SELECT -
MA7
+5V
SEL
BSEL -
READ -
SH 4
SH 4,5
PI-33
PI-34
PI-35
PI-36
PI-37
PI-38
PI-39
PI-40
PI-41
PI-42
PI-43
PI-44
PI-45
PI-46
PI-47
PI-48
D96214
1600013 FILM
NONE
SHEET 3



1600013 4
READY.B
P1-90
READY WAIT STATES
TERM WAIT STATES
0 WAIT
1 WAIT
2 WAIT
E23
E24
E25
E26
E27
E28
E29
E30
R11 10K +5V
WAITJ SH5
SH 2 SH 2,3 SH 5 SH 5
A15 READ MEMEN- CLK-
74LS74
U43
MEMEN-
CLK-
HI
R8 4.7K PRES-
+5
74S241
U6
74S240
U7
P1-72 A15.B
P1-22 BUSCLK.B-
P1-84 MEMCYC.B-
P1-82 DBIN.B
P1-80 MEMEN.B-
MRS-
PMEMCYC-
MEMCYC-
E73 E74 E75
EARLY LATE
R7 1K +5V
SH2 HI1
SH2,5 REFRESHQ1-
SH2,5 REFRESHQ
SH5 SEL-
U11 74S132
U8 74S03
R12 330
BSEL
WAITQ
WAITQQ
74S114
U9
U31
U15
TERM
TERM-
WAIT J
WAIT Q
STARTQ-
74S00
U32
CLK
MRS-
RASD-
REFRESHQ-
MEMENQ- SH 5
REFTIMER SH 5
E65
STEAL
TRANS
E44
E45
R17 10K +5
SH3,5 READ-
SH2,3 SEL
SH3 BSEL-
SH2 BSEL
U30
START-
STARTJ
U29 74S11
E78 E79
R41 10K +5V
U14 74S51
REFRESHJ
REFRESHJ-
RASD- SH 5
R10 330
REFTIMEQ
WAITQ-
D 96214
1600013
E
NONE
4



TEXAS INSTRUMENTS
11/18/76
D 96214 1600013 FILMED E
SCALE NONE
SHEET 5

APPENDIX C

CRU PARITY READ/RESET USER OPTION

C.1 INTRODUCTION

This appendix contains detailed information on selecting the CRU map and coding the 74LS287 PROM (in U5) for using the CRU to read and reset the parity interrupts on the TM 990/203. In a system using more than one TM 990/203 board, each parity interrupt PROM can be given a separate CRU address; thus a unique address can be used to handle parity errors for each board.

A logic 1 found at this address indicates a parity error occured; a zero indicates no error occured. The error can be reset by writing to the address (any CRU write instruction---SBO, SBZ, or STCR).

C.2 CRU ADDRESS NOMENCLATURE

Refer to Figure C-1 for the following definitions:

- CRU Hardware Base Address: Contents of bits 3 to 14 of register 12
- CRU Bit Address: CRU Hardware Base Address plus a signed displacement contained in the CRU instruction
- CRU Software Base Address: Contents of bits 0 to 15 of register 12.

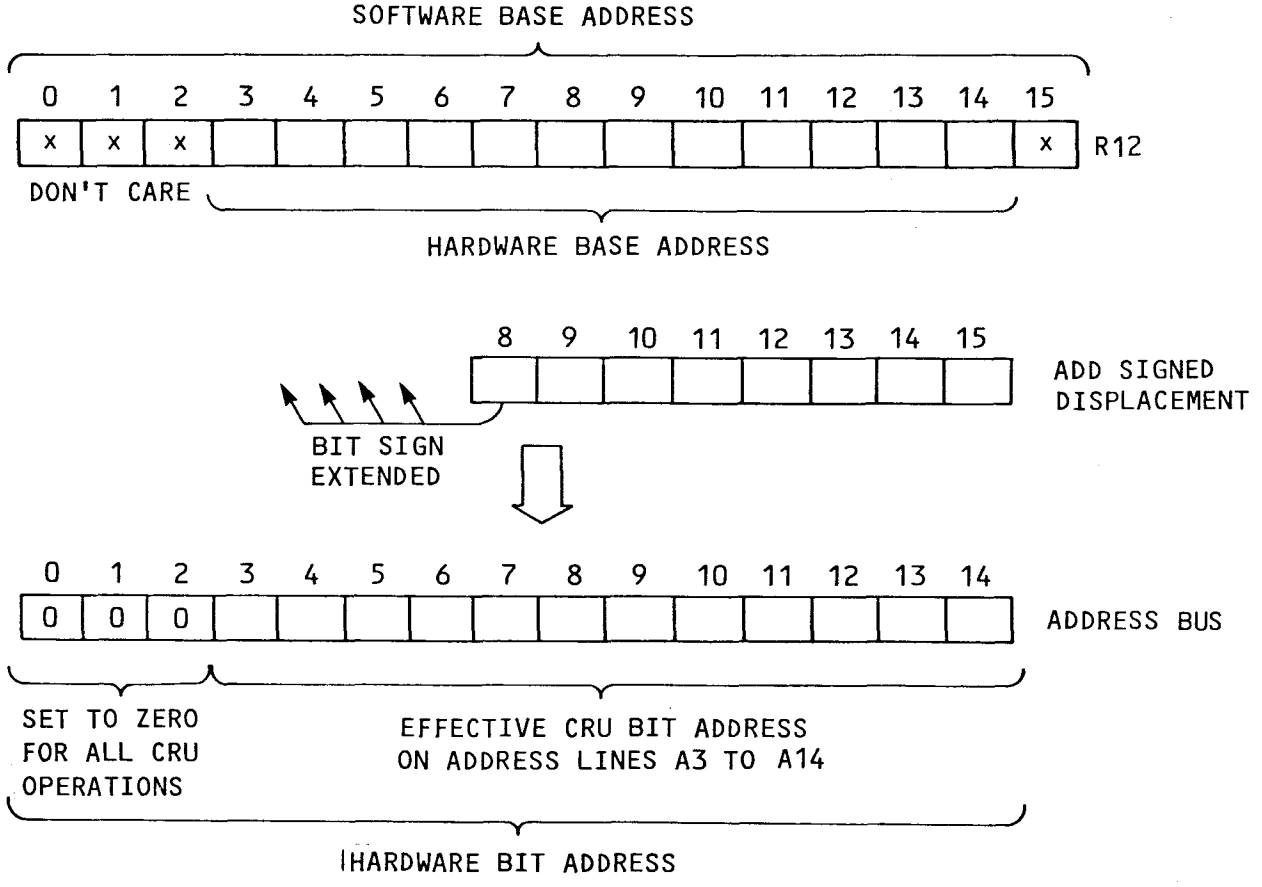


FIGURE C-1. CRU ADDRESS NOMENCLATURE

C.3 CRU MAP SELECTION AND PROGRAMMING CONSIDERATIONS

The following considerations must be taken into account when setting up the CRU map and programming the PROM:

- The TM 990/100 and /101 microcomputer boards reserve the first 256 bits (sixteen 16-bit words) of the total 4096-bit CRU space for I/O interrupt lines and on-board expansion. Therefore, CRU hardware baseaddresses $0000\text{-}00FF_{16}$ z(contained on address lines A3-A14 or software base address $0000\text{-}01FE_{16}$) are reserved, and CRU hardware base addresses $0100_{16}\text{-}0FFF_{16}$ are available for expansion.

- Sixteen contiguous bits of CRU space are enabled by the PROM "nibble" (4-bits) which is programmed to a logic 0. CRU Address lines A11-A14 (least significant) are not decoded by the PROM and therefore are "don't cares" that do not affect the selecting of the PROM nibbles. The 16 CRU bit addresses enabled can be computed by concatenating any of the poissible 16 hex logic values ($0 - F_{16}$) of address lines A11-A14 to the end of the hex logic value on A3-A10. For example, if A3-A10 containewd address $7C_{16}$ then CRU bit addresses $7C0_{16}$ to $7CF_{16}$ would all enable the same PROM nibble located at CRU bit address $7C0_{16}$ (see Figure C-2).

- Only one nibble per PROM should be programmed to a logic 0 (all others to a 1), and in multiboard memory systems, each of the nibbles programmed to logic 0 on each board should have its own unique CRU address. This insures that a CRU software base address (one 16-bit word) will enable only one PROM on one board. In this way, a CRU address from the microcomputer will enable only one PROM; thus, preventing the outputs to CRUIN from different memory boards from overlapping and conflicting with one another.

- The specific "nibble" in the 74S287 PROM must be programmed to a logic 0 output for a selected CRU address input on address lines A#-A10. However, to simplify PC board layout, address lines A3 - A10 do not correspond directly to address input lines H - A on the 74LS287 PROM (see Figure C-2).

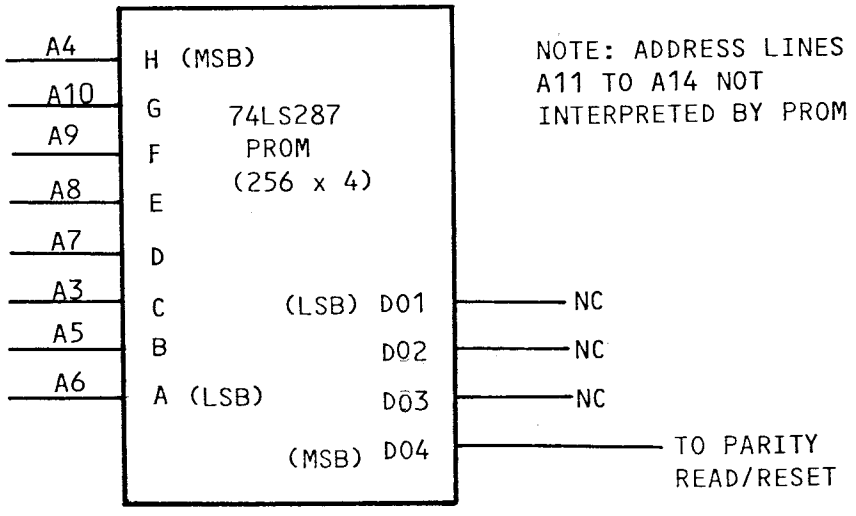


FIGURE C-2 PROM U5 (74LS287) ADDRESS AND DATA PINS

C.4 DETERMINE PROM ADDRESS TO BE CODED WITH ZEROES

Use the following steps to code the PROM correctly:

1) Determine the desired hardware base address (bits 3 - 14 of R12) of the PROM. (Note that only bits 3-10 are used). Address bits A11-A14 are not decoded and are "don't cares" (see Figure C-2).

2) Insert the binary value of the hardware base address in the 12 blanks in the top of Figure C-3 (Note that bits 11-14 are "don't cares", and bits 0-3 and bit 15 are not used)

3) Transfer the binary value of bits 3 to 10 in the top of Figure C-3 to the correspondingly numbered boxes in the bottom of the Figure. The resulting 8-bit vbalue in the bottom boxes will be the hex PROM address (input pins A-H) of the "nibble" to be programmed to a logic 0.

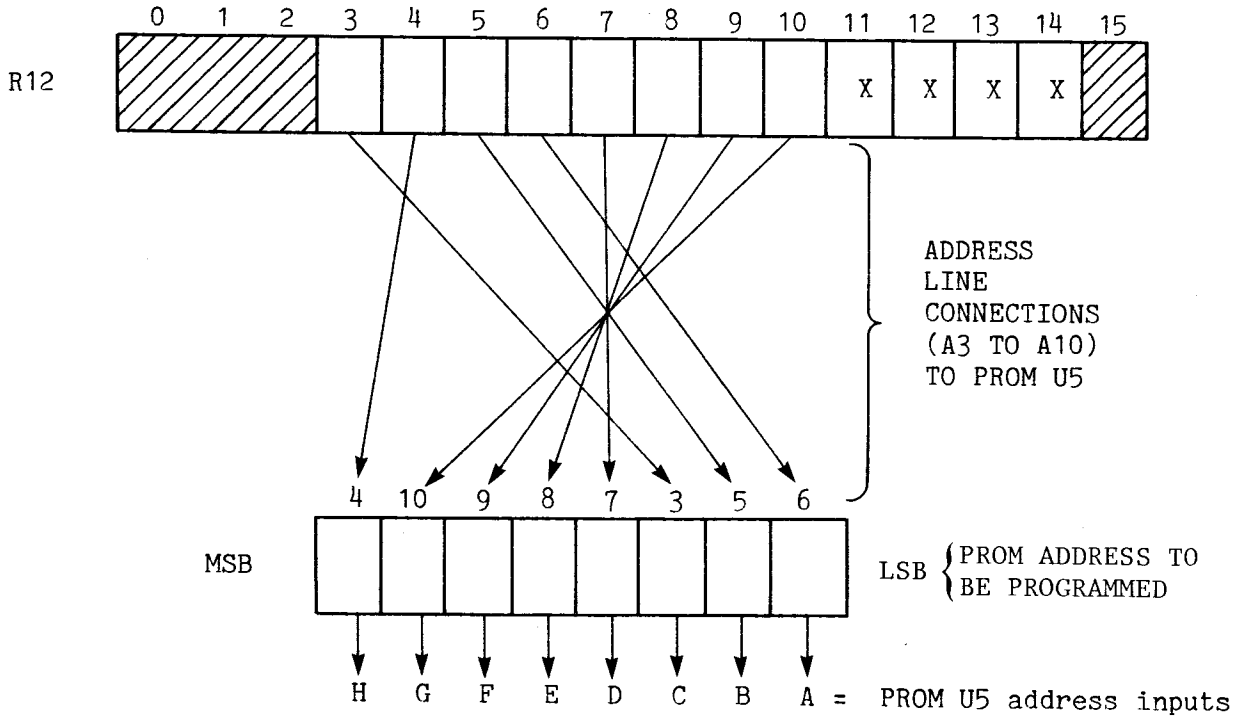


FIGURE C-3 INTERPRETING R12 CONTENTS INTO ADDRESS OF PROM "NIBBLE"

C.5 PROGRAMMING EXAMPLES

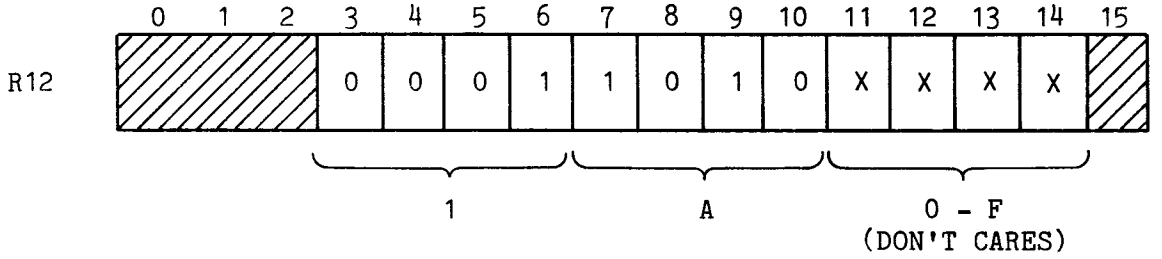
This subsection illustrates the development of a PROM address from the CRU hardware base address.

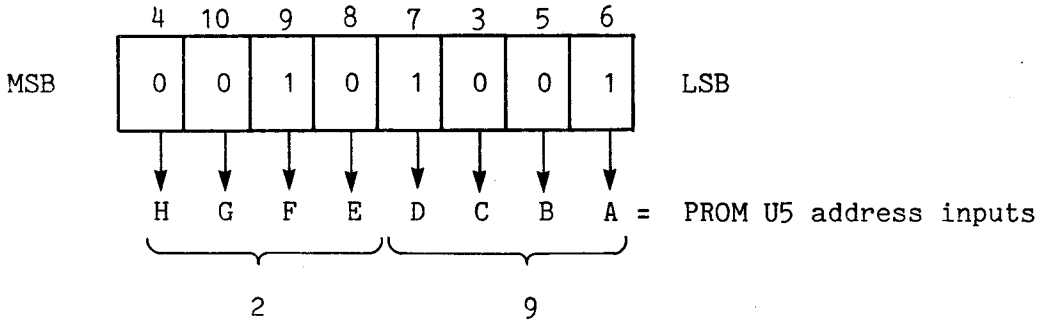
C.5.1 Example 1

In this example, the desired hardware base address is $01A0_{16}$.

1) CRU Hardware Base Address = $01A0_{16}$
   CRU bits enabled = $01A0_{16}$ - $01AF_{16}$

2) Interpret "nibble" address in PROM





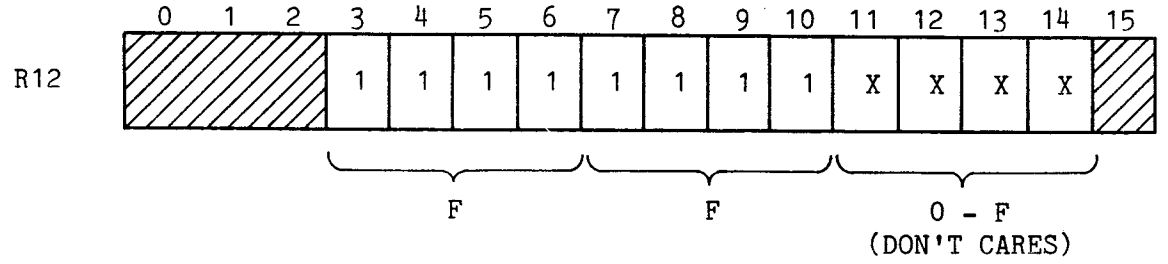
Therefore, the PROM "nibble" at address 29_{16} should be programmed to all zeroes, and the remaining "nibbles" programmed to all ones (note that the "nibble" can be enabled by using CRU hardware base addresses $1A0_{16}$-$1AF_{16}$, software base addresses 0340_{16} - $035E_{16}$, because only bits 3-10 of R12 are used to select the PROM address.

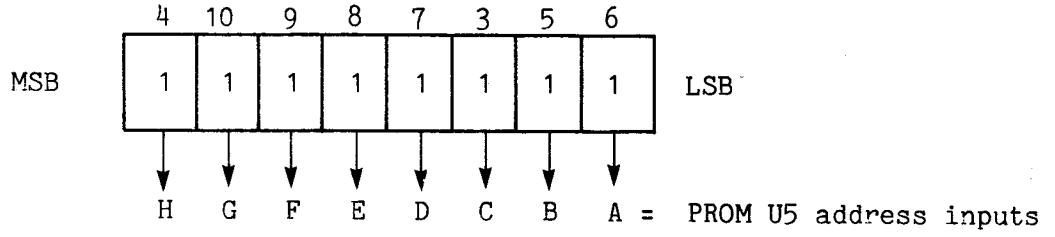
C.5.2 Example 2

In this example, the desired CRU software base address is $1FFE_{16}$.

1) CRU Software Base Address = $1FFE_{16}$
   CRU Hardware Base Address = $0FFF_{16}$
   CRU bits enabled = $0FF0_{16}$

2) Interpret "nibble" address in PROM.





Note that because address lines A3-A10 were all logic 1, the bit pattern to the PROM was the same as the contents of bits 3-10 of register 12.

C.5.3 PROM Contents Example

Figure C-4 is a listing of the 256 four-bit "nibbles" in a PROM that has been programmed to enable the parity error read/reset circuitry when $FF_{16}$is applied to the PROM address inputs (as shown in the second example in C.5.2 above). Each line labeled ADD00 to ADDF0 represents 16four-bit "nibbles", whose contents is reflected by the 16hexadecimal characters to the right of these lables. The first line contains the values for PROM input address 00 to $0F_{16}$, the second line for input address 10_{16} to $1F_{16}$, is at PROM input address FF_{16}.

```
        TITL 'ROM203              2/2/79'
        IDT  'ROM203'
ADD00   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADD10   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADD20   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADD30   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADD40   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADD50   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADD60   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADD70   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADD80   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADD90   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADDA0   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADDB0   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADDC0   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADDD0   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADDE0   DATA >FFFF,>FFFF,>FFFF,>FFFF
ADDF0   DATA >FFFF,>FFFF,>FFFF,>FFF0
        END
```

FIGURE C-4 EXAMPLE LISTING OF PROM CONTENTS

APPENDIX D

UTILIZING THE PARITY OPTION

D.1 GENERAL

This section describes the requirements for using the parity option on the TM 990/203. An example program is also included.

D.2 PARITY OPTION SOFTWARE REQUIREMENTS

The software requirements for utilizing the parity option of the TM 990/203 are as follows:

1. Initialize routine - initializes all the parity bits upon powerup or parity interrupt rrequest.

2. Interrupt setup routine - routine that enables the appropriate interrupt and possibly loads interrupt vectors.

3. Interrupt handler routine - routine written to possibly recover from a parity failure and diagnose or analyze a parity problem. It may call the parity initialize routine in order to correct the parity bit that failed.

D.3 PARITY INITIALIZE ROUTINE

The parity initialize routine should be written as a subroutine so that it is available upon request and also can be called during a powerup sequence. The Interrupt Handler may use this routine to correct "soft" errors. A very simple but effective method of initializing the parity bits can be accomplished by using the code given in the SETPAR routine (page 005 of the listing in this appendix). In this routine the content of each memory location is read and then written back. This method is non-destructive to the contents of memory and can be called at any time with no consequence to the user. In short the initialize routine should write to every enabled location in the dynamic memory. The ability to reset memory without disturbing the contents is a useful feature.

D.4 INTERRUPT SETUP ROUTINE

The setup routine as shown in the listing is somewhat more complex than seemingly need be because it is designed to operate under the TIBUG monitor for the TM 990/101. The reason for the added complexity is that the hardware is fixed. In essence, the interrupt vectors required for interrupt processing are "fixed" in EPROM. The program example reads the interrupt vector program counter value from EPROM. It inserts a BLWP (branch and load workspace pointer) command to branch to the location of the interrupt handler. The processor reads the vector upon receipt of an interrupt request, branches to the program counter location and then encounters the branch inserted at this location. This method is possible because the programmed values for the interrupt vectors in 101 TIBUG point to user RAM and thus can be loaded with the BLWP command followed by the branch address. If possible the interrupt vectors should be programmed with the correct program counter and workspace pointer. In that case the program sequence that loads the branch commands can be omitted: this precludes the use of TIBUG for the TM 990/101.

Another more essential part of the software interrupt routine is to load the TMS 9901 interrupt mask to enable the selected parity interrupt for the TM 990/203 interrupt request. This is necessary because the interrupt from the TM 990/203 is processed by the TMS 9901 on the processor module.

D.5 INTERRUPT HANDLER

An interrupt handler serves two purposes. The first is to indicate that something was in error. The second is to allow the user to recover from a parity error and isolate the problem or return the control to the processor.

Two examples of interrupt handlers are given in the listing. One prints a snapshot of the program environment at the time of the interrupt request. This is a printing of the workspace pointer, program counter and status as well as the contents of the workspace registers. The other examines the interrupt input while reading consecutive locations in memory. This routine maps the memory by indicating the location(s) with parity problems, if the problem is repeatable. It may not catch a location with a "soft" error.

Diagnostic routines can be substituted in place of these routines in order to locate failing devices.

D.6 HARDWARE

The hardware needed for the example software given is a TM 990/101M microcomputer and a TM 990/203 expansion memory module. TIBUG resides at locations 0000_{16} to 0FFE16 in EPROM on the TM 990/101M, and and the RAM area on the TM 990/101M is mapped into the memory area from FE0016 to FFFE16.

The memory on the TM 990/203 can occupy any of the memory space not occupied by the program and the TM 990/101 memory. The program should reside in EPROM.

D.7 BOARD INITIALIZATION LISTING

The listing of an example powerup routine that can be used to initialize the TM 990/203 is given on the following pages.

TM990/203 DYNAMIC RAM SET UP ROUTINE

```
0002                    IDT  'PWRUP203'
0003              *
0004              *          THIS PROGRAM IS JUST ONE EXAMPLE OF A POWER-UP
0005              *     ROUTINE THAT COULD BE USED TO INITIALIZE THE TM990/203
0006              *     DYNAMIC MEMORY BOARD.
0007              *          IN GENERAL TERMS THE FLOW OF THE PROGRAM IS AS
0008              *     FOLLOWS:
0009              *          1. INITIALIZE THE PARITY BITS IN MEMORY
0010              *          2. SET THE TMS9901 INTERRUPT MASK TO ACCEPT
0011              *             THE PARITY INTERRUPT.
0012              *          3. INSERT BRANCHES TO THE PARITY INTERRUPT
0013              *             HANDLERS AT THE PARITY INTERRUPT
0014              *             PROGRAM COUNTER LOCATION.
0015              *          4. ENABLE THE TMS9900 INTERRUPTS
0016              *          5. EXIT
0017              *
0018              *          TWO EXAMPLE INTERRUPT HANDLERS ARE GIVEN.
0019              *
0020              *
0021              * 1. *
0022              *          TASK NUMBER ONE IS TO INITIALIZE ALL THE PARITY
0023              *     BITS IN MEMORY. THIS IS ACCOMPLISHED BY READING FROM
0024              *     EACH LOCATION AND THEN WRITTING THE SAME INFORMATION
0025              *     BACK. IN THIS WAY ALL THE PARITY BITS ARE SET AND
0026              *     NONE OF THE INFORMATION IN MEMORY IS DESTROYED. THIS
0027              *     ROUTINE IS VERY USEFUL IF EXECUTED IN A WORKING
0028              *     ENVIRONMENT WHERE THE DATA IN MEMORY CAN NOT BE
0029              *     DESTROYED.
0030              *
0031              *
0032              * 2. *
0033              *          THE SECOND TASK IS TO WRITE THE 9901 INTERRUPT
0034              *     MASK WITH A ONE FOR THE LEVEL OF INTERRUPT ASSIGNED
0035              *     TO THE PARITY INTERRUPT. NOTE HERE THAT ONLY THE
0036              *     BITS USED ARE WRITTEN TO, NONE OF THE OTHER INTERRUPT
0037              *     MASK BITS ARE CHANGED.
0038              *
0039              *
0040              * 3. *
0041              *          THE THIRD PART OF THIS EXAMPLE INSERTS BRANCHES
0042              *     AT THE INTERRUPT VECTOR PROGRAM COUNTER LOCATIONS.
0043              *     WHEN A PARITY INTERRUPT OCCURRS THE PROCESSOR PICKS UP
0044              *     THE PROGRAM COUNTER AND WORKSPACE POINTER FROM THE
0045              *     INTERRUPT VECTOR AS USUAL. UPON ARRIVAL AT THE NEW
0046              *     PROGRAM COUNTER LOCATION THE PROCESSOR FINDS A BRANCH
0047              *     TO THE INTERRUPT HANDLER ROUTINE, ASSUMING THAT NOTHING
0048              *     HAS TAMPERED WITH THE CONTENTS OF THIS LOCATION SINCE
0049              *     THE POWERUP ROUTINE WAS EXECUTED.
0050              *          FOLLOWING THIS BRANCH TO THE INTERRUPT HANDLER
0051              *     THERE MUST BE A RETURN INSTRUCTION INSERTED. THIS IS
0052              *     INSERTED TO ENABLE THE PROGRAM TO RETURN TO THE
0053              *     PROGRAM THAT WAS EXECUTING AT THE TIME OF THE INTERRUPT
0054              *     REQUEST.
0055              *          THE ASSUMPTIONS THAT ARE MADE ARE AS FOLLOWS.
```

```
0056            *   A. THE USER IS EXECUTING TIBUG 401-3.
0057            *                      THIS TIBUG HAS THE INTERRUPT VECTORS
0058            *                      PROGRAMMED SO THAT THE PC AND WS ARE
0059            *                      BOTH IN RAM. THUS THE BRANCH TO  AN
0060            *                      INTERRUPT HANDLER CAN BE INSERTED.
0061            *       TIBUG 401-3 CAN BE USED BUT ONLY INT. 2
0062            *       VECTOR IS PROGRAMMED TO BE USEFUL.
0063            *   B. WORKSPACES ARE DEFINED BY THE USER NOT TO CONFLICT
0064            *                      WITH THOSE USED BY THE HANDLERS OR
0065            *                      THE INTERRUPT VECTOR ALLOCATIONS.
0066            *   C. THIS ROUTINE IS CALLED VIA TIBUG'S 'E' COMMAND
0067            *                      AFTER THE PC IS SET UP USING THE
0068            *                      'R' COMMAND. EXECUTION STARTS AT THE
0069            *                      FIRST LOCATION.
0070            *
0071            *
0072            * 4. *
0073            *         THE FOURTH TASK IS SIMPLY A LOAD INTERRUPT MASK
0074            *    IMMEDIATE (LIMI) STATEMENT OF THE PROPER LEVEL IN
0075            *    ORDER THAT THE TMS9900 WILL ACCEPT THE INTERRUPT.
0076            *
0077            *
0078            *===============================================================
0079            *
0080            *       WORKSPACE ALLOCATIONS
0081            *
0082     FF00   TSTWS  EQU  >FF00              WORKSPACE FOR POWER UP ROUTINE
0083     FF20   INTWS  EQU  >FF20              WORKSPACE FOR INTERRUPTS
0084            *
0085            *
0086            *===============================================================
0087            *
0088            *       XOP  DEFINITIONS
0089            *
0090                   DXOP HEXO,10
0091                   DXOP PRNT,14
0092            *
0093            ****************************************************************
0094            *  MISCELEANOUS EQUATES
0095            *
0096     0100   BA9901 EQU  >0100              9901 CRU BASE ADDRESS
0097     0080   TIBUG  EQU  >0080              TIBUG ENTRY POINT
0098     0380   RTWPC  EQU  >0380              RTWP COMMAND
0099     0420   BLWPC  EQU  >0420              BLWP COMMAND
0100            *
0101            *
0102            ****************************************************************
0103            ****************************************************************
0104            *    POWER UP MEMORY INITIALIZE ROUTINE                        *
0105            ****************************************************************
0106            ****************************************************************
0107            *
0108            *           THIS ROUTINE WRITES TO EVERY LOCATION TWICE
0109            *       IN ORDER TO SET ALL THE PARITY BITS.
```

```
0110                *       THE REASON FOR WRITING TO THE MEMORY TWICE IS TO
0111                *       INITIALIZE THE MEMORY CHIPS. A MINIMUM OF EIGHT
0112                *       CYCLES IS REQUIRED BEFORE THE MEMORIES ARE OPERABLE.
0113                *
0114                *
0115                POWER
0116 0000 02E0              LWPI TSTWS              LOAD INITIALIZE WORKSPACE
     0002 FF00
0117 0004 2FA0              PRNT @PWRUP             PRINT BANNER TO INDICATE
     0006 023E'
0118                *                               WHAT IS HAPPENING.
0119 0008 06A0              BL   @SETPAR            WRITE TO ALL LOCATIONS
     000A 004E'
0120 000C 06A0              BL   @SETPAR            TWICE
     000E 004E'
0121                *                               TO CLEAR ALL PARITY BITS
0122                *
0123                ****************************************************************
0124                *       THIS SECTION ENABLES THE PARITY INTERRUPT BY
0125                *       LOADING THE 9901 INTERRUPT MASK. ONLY ONE INTERRUPT
0126                *       INPUT NEED BE ENABLED DEPENDING ON THE STATE OF THE
0127                *       JUMPER ON THE TM990/203 MEMORY BOARD THAT SELECTS
0128                *       THE PRIORITY OF THE PARITY INTERRUPT.
0129                *
0130 0010 020C              LI   R12,BA9901         R12 = 9901 BASE ADD.
     0012 0100
0131 0014 1E00              SBZ  0                  INT MODE
0132 0016 1D01              SBO  1                  INT 1 MASK
0133                *                               ***OMIT PREVIOUS LINE IF INT.
0134                *                                  LEVEL 2 IS TO BE USED.
0135                *                               JUMPER E8 TO E7
0136                *
0137 0018 1D02              SBO  2                  INT 2 MASK
0138                *                               ***OMIT PREVIOUS LINE IF INT.
0139                *                                  LEVEL 1 IS TO BE USED TO
0140                *                                  HANDLE PARITY INT. REQUESTS
0141                *                               JUMPER E8 TO E9
0142                *
0143                ********************************************************
0144                * NEXT
0145                *           THE PARITY INTERRUPT FLIP FLOP IS RESET
0146                *       AND A BRANCH TO THE PARITY INTERRUPT HANDLER IS
0147                *       INSERTED AT THE INT 1 AND INT 2 PROGRAM
0148                *       COUNTER LOCATIONS.
0149 001A 1E17              SBZ  23                 CLEAR INT. REQUEST IF ACTIVE
0150 001C 1D17              SBO  23                 RE-ENABLE PARITY INTERRUPT
0151                *                               BIT 23 IS P7 OF 9901 USED
0152                *                               TO CLEAR PARITY INTERRUPTS
0153                *                               ON THE TM990/203.
0154                *
0155                *           THE FOLLOWING INSERTS OF BRANCHES TO INTERRUPT
0156                *       HANDLER ROUTINES CAN BE OMITTED IF THE INTERRUPT
0157                *       VECTORS CAN BE CHANGED TO ACCOMMODATE THE INTERRUPT.
0158                *           THIS HAS BEEN ADDED TO WORK WITH 401-3 TIBUG.
```

```
0159                *       THE INTERRUPT VECTOR FOR INT 1 IN 401-3 TIBUG IS
0160                *       WS=FF5A PC=FF7A
0161                *       THE INTERRUPT VECTOR FOR INT 2 IN 401-3 TIBUG IS
0162                *       WS=FF4E PC=FF6E
0163                *
0164 001E 0201              LI    R1,>6             VECTOR FOR INT1
     0020 0006
0165 0022 C051              MOV   *R1,R1            GRAB VECTOR PC VALUE
0166 0024 0202              LI    R2,BLWPC          REG2 = BLWP COMMAND
     0026 0420
0167 0028 CC42              MOV   R2,*R1+           SET UP BRANCH COMMAND
0168 002A 0203              LI    R3,INT1           PICK UP INT HANDLER ENTRY ADD.
     002C 0058'
0169 002E CC43              MOV   R3,*R1+           SET UP CODE ENTRY (INT PC )
0170 0030 0204              LI    R4,RTWPC          THIS IS A RTWP COMMAND
     0032 0380
0171 0034 C444              MOV   R4,*R1            INSERT FOR RETURN
0172 0036 0201              LI    R1,>A             VECTOR PC FOR INT2
     0038 000A
0173 003A C051              MOV   *R1,R1            GRAB VECTOR PC VALUE
0174 003C CC42              MOV   R2,*R1+           SET UP BRANCH COMMAND
0175 003E 0203              LI    R3,INT2           PICK UP INT HANDLER ENTRY ADD.
     0040 0102'
0176 0042 CC43              MOV   R3,*R1+           SET UP CODE ENTRY (INT PC)
0177 0044 C444              MOV   R4,*R1            INSERT RTWP INSTR.
```

```
0179                ************************************************************
0180 0046 0300          LIMI 15                     UNMASK INTERRUPTS
     0048 000F
0181 004A 0460          B    @TEST1
     004C 018C'
0182                *
0183                ****************************************
0184                * THIS ROUTINE IS CALLED VIA 'BL'
0185                * NOTE THAT IT DOES NOT CHANGE THE CONTENTS
0186                *      OF MEMORY.
0187                ****************************************
0188                SETPAR
0189 004E 04C3          CLR  R3                     SET START ADDRESS
0190                SETP
0191 0050 C4D3          MOV  *R3,*R3                READ THEN WRITE AT SAME LOC.
0192 0052 05C3          INCT R3                     INC ADDRESS AND CHECK IF = 0
0193 0054 16FD          JNE  SETP                   IF NOT DONE CONTINUE
0194 0056 045B          RT                          OTHERWISE RETURN
0195                *
0196                ************************************************************
0197                * PARITY INTERRUPT HANDLER  (INT1)
0198                ************************************************************
0199                *
0200                *           THIS ROUTINE PRINTS A SNAPSHOT OF
0201                *      THE PROGRAM UNDER EXECUTION AT THE TIME OF
0202                *      A PARITY INTERRUPT REQUEST (INT1)
0203                *           THIS PROGRAM FETCHES REGISTERS FOR TWO
0204                *      BRANCHES BACK. ONE FOR THE INTERRUPT AND ONE
0205                *      FOR THE BRANCH INSERTED AT THE INT. VECTOR
0206                *      PROGRAM COUNTER.
0207                *
0208                INT1
0209 0058 FF20          DATA INTWS,INT1PC           W.S. AND P.C.
     005A 005C'
0210                INT1PC
0211 005C 0300          LIMI 0                      DISABLE INTERRUPTS
     005E 0000
0212 0060 020C          LI   R12,>0120              SET CRU BASE TO 9901
     0062 0120
0213 0064 1E07          SBZ  7                      CLEAR AND DISABLE PAR. INT.
0214 0066 2FA0          PRNT @INTMSG                PRINT 'PAR. INT. HAS OCCURRED'
     0068 00D2'
0215 006A C20D          MOV  R13,R8                 PICK UP BRANCH WORKSPACE
0216 006C C068          MOV  @26(8),R1              PICK UP PROGRAM W.P.
     006E 001A
0217 0070 2FA0          PRNT @WS                    PRINT HEADER FOR WS
     0072 00EC'
0218 0074 2E81          HEXO R1                     OUTPUT IT
0219 0076 2FA0          PRNT @PC                    PRINT HEADER FOR PC
     0078 00F2'
0220 007A C068          MOV  @28(8),R1              GET PROGRAM P.C.
     007C 001C
0221 007E 2E81          HEXO R1                     OUTPUT IT
0222 0080 2FA0          PRNT @ST                    PRINT HEADER FOR STATUS
```

```
     0082 00F8'
0223 0084 C068          MOV  @30(8),R1          GET PROGRAM ST.
     0086 001E
0224 0088 2E81          HEXO R1                 OUTPUT IT
0225 008A 2FA0          PRNT @CRLF              CLEAR PRINT LINE
     008C 0189'
0226 008E 2FA0          PRNT @CRLF              SPACE LINE
     0090 0189'
0227 0092 C228          MOV  @26(R8),R8         PICK UP PROG.REGS.
     0094 001A
0228 0096 0203          LI   R3,>5230           ASCII 'R0'
     0098 5230
0229 009A 0204          LI   R4,>3D00           ASCII '='
     009C 3D00
0230 009E 04C2          CLR  R2                 CLEAR FLAG
0231              WSOUT
0232 00A0 0200          LI   R0,>FFF8           COUNTER FOR 8 CHAR'S
     00A2 FFF8
0233              WSOUT1
0234 00A4 C078          MOV  *R8+,R1            GET OLD REG. CONTENTS
0235 00A6 2FA0          PRNT @SPACE             OUTPUT SPACES
     00A8 00FE'
0236 00AA 2F83          PRNT R3                 OUTPUT REG. NAME
0237 00AC 2E81          HEXO R1                 OUTPUT REG. CONTENTS
0238 00AE 0583          INC  R3                 UPDATE NAME
0239 00B0 0283          CI   R3,>523A           IS NAME INVALID ?
     00B2 523A
0240 00B4 1602          JNE  FIXRET             IF NOT, BYPASS FIX
0241 00B6 0203          LI   R3,>5241           CHANGE 'R:' TO 'RA'
     00B8 5241
0242              FIXRET
0243 00BA 0580          INC  R0                 INCREMENT REG. COUNT
0244              *                             EIGHT ON THE LINE ?
0245 00BC 16F3          JNE  WSOUT1             IF NOT, CONTINUE
0246 00BE 2FA0          PRNT @CRLF              OTHERWISE, CLEAR PRINT LINE
     00C0 0189'
0247 00C2 C082          MOV  R2,R2              ALL DONE ?
0248 00C4 1602          JNE  ENDPAR             IF SO, RETURN
0249 00C6 0702          SETO R2                 SET LOOP FLAG
0250 00C8 10EB          JMP  WSOUT              AND FINISH
0251              ENDPAR
0252 00CA 06A0          BL   @SETPAR
     00CC 004E'
0253 00CE 1D07          SBO  7                  RE-ENABLE PAR. INT.
0254 00D0 0380          RTWP                    RETURN TO ON GOING PROG.
```

```
0256                *****************************************************************
0257                **************   INTERRUPT HANDLER MESSAGES   ***************
0258                *****************************************************************
0259                INTMSG
0260 00D2   0D          BYTE >D,>A
     00D3   0A
0261 00D4   20          TEXT ' PARITY INT. OCCURRED'
0262 00E9   0D          BYTE >D,>A,>0
     00EA   0A
     00EB   00
0263                WS
0264 00EC   20          TEXT '  WS='
0265 00F1   00          BYTE 00
0266                PC
0267 00F2   20          TEXT '  PC='
0268 00F7   00          BYTE 00
0269                ST
0270 00F8   20          TEXT '  ST='
0271 00FD   00          BYTE 00
0272                SPACE
0273 00FE   20          TEXT '  '
0274 0100   00          BYTE 00
0275                    EVEN
```

```
0277                    *
0278                    ************************************************************
0279                    * PARITY INTERRUPT HANDLER  (INT2)                         *
0280                    ************************************************************
0281                    *
0282                    *           THIS ROUTINE READS THROUGH MEMORY TO FIND
0283                    *       THE LOCATION OF PARITY ERRORS. THE ADDRESS
0284                    *       OF EACH LOCATION FOUND WITH A PARITY PROBLEM
0285                    *       IS PRINTED.
0286                    *           TO HALT THE EXECUTION OF THIS PROGRAM
0287                    *       TYPE ANY CHARACTER.
0288                    *           ALTHOUGH THIS PROGRAM IS NON-DESTRUCTIVE
0289                    *       TO THE CONTENTS OF MEMORY THE ROUTINE MUST NOT
0290                    *       RUN THRU THE WORKSPACE REGISTERS. UPON COMPLETION
0291                    *       EXECUTION IS SENT BACK TO THE ROUTINE UNDER
0292                    *       WAY AT THE TIME OF THE INTERRUPT.
0293                    *
0294                    INT2
0295 0102 FF20                  DATA INTWS,INT2PC
     0104 0106'
0296                    INT2PC
0297 0106 0300                  LIMI 0                  MASK ALL INTERRUPTS
     0108 0000
0298 010A 2FA0                  PRNT @INTMSG            OUTPUT INTERRUPT MESSAGE
     010C 00D2'
0299 010E 2FA0                  PRNT @INT2CK            PRINT INT2 MSG.
     0110 0170'
0300 0112 04C1                  CLR  R1                 CLEAR LOOP COUNT
0301 0114 04C4                  CLR  R4                 CLEAR FOUND FLAG
0302 0116 020C                  LI   R12,>080           SET CRU BASE TO 9902
     0118 0080
0303 011A 1E12                  SBZ  18                 CLEAR 9902 RCV. BUF. FULL FLAG
0304 011C 0A1C                  SLA  R12,1              SET UP FOR 9901 BASE
0305                    AGAIN1
0306 011E 0202                  LI   R2,>FFFE           SET START ADDRESS - 2
     0120 FFFE
0307 0122 0203                  LI   R3,>FE00           SET END ADDRESS
     0124 FE00
0308                    INT2GO
0309 0126 0200                  LI   R0,>FFF6           SET #/LINE COUNT (10)
     0128 FFF6
0310 012A 2FA0                  PRNT @CRLF              GO TO NEXT LINE
     012C 0189'
0311                    MIX
0312                    ************************************************************
0313                    * CHECK FOR A CHARACTER TO TERMINATE INT. HANDLER
0314                    *       PREMATURELY
0315                    *
0316                    CHARCK
0317 012E 1FD5                  TB   >AA->100/2         9902 RECIEVE BUFFER FULL
0318                    *                               >AA IS BIT 21 AT THE 9902 BASE
0319                    *                               ADD. OF >80. THIS IS THE RCV.
0320                    *                               BUFFER FULL FLAG.
0321 0130 1318                  JEQ  EINT2              IF SO, END ROUTINE
```

```
0322                  ************************************************************
0323 0132 1E17              SBZ  23                  RESET AND DIASBLE INT.
0324 0134 1D17              SBO  23                  RE-ENABLE INT.
0325                  MIX1
0326 0136 05C2              INCT R2                  INCREMENT ADDRESS
0327 0138 80C2              C    R2,R3               DONE YET ?
0328 013A 130C              JEQ  EOINT2              IF SO, GO TO END OF INT2
0329 013C C052              MOV  *R2,R1              READ LOCATION
0330 013E 1000              NOP                      ***WAIT***WAIT***
0331 0140 C481              MOV  R1,*R2              REWRITE PARITY BIT
0332 0142 1F02              TB   2                   READ INT INPUT
0333 0144 13F8              JEQ  MIX1                IF NOT, KEEP GOING !
0334 0146 2FA0              PRNT @SPACE
     0148 00FE'
0335 014A 0704              SETO R4                  SET FOUND FLAG
0336 014C 2E82              HEXO R2                  OUTPUT ERROR ADDRESS
0337 014E 0580              INC  R0                  INCREMENT #/LINE COUNT
0338 0150 13EA              JEQ  INT2GO
0339 0152 10ED              JMP  MIX                 MIX IT UP
0340                  EOINT2
0341 0154 C104              MOV  R4,R4               FOUND ANY ?
0342 0156 1605              JNE  EINT2               IF SO EXIT
0343 0158 C041              MOV  R1,R1               2 LOOPS DONE?
0344 015A 1601              JNE  EOI2                IF SO, END TEST
0345 015C 10E0              JMP  AGAIN1
0346                  EOI2
0347 015E 2FA0              PRNT @NONE               IF NOT PRINT SO
     0160 017D'
0348                  EINT2
0349 0162 2FA0              PRNT @CRLF               CLEAR PRINT LINE
     0164 0189'
0350 0166 06A0              BL   @SETPAR             RESET PARITY BITS
     0168 004E'
0351 016A 1E17              SBZ  23                  RESET PARITY INTERRUPT
0352 016C 1D17              SBO  23                  RE-ENABLE PAR. INT.
0353 016E 0380              RTWP
```

```
0355                    *       MESSAGES FOR INT2
0356                    *
0357                    INT2CK
0358 0170   45                  TEXT 'ERRORS AT:'
0359 017A   0D                  BYTE >D,>A,>0
     017B   0A
     017C   00
0360                    NONE
0361 017D   0D                  BYTE >D,>A
     017E   0A
0362 017F   4E                  TEXT 'NONE FOUND'
0363 0189   0D          CRLF    BYTE >D,>A,>0
     018A   0A
     018B   00
0364                            EVEN
0365                    *
0366                    ************************************************************
0367                    *       TEST 1      LOCATES VALID RAM MEMORY                *
0368                    ************************************************************
0369                    *
0370                    * TEST 1 DEFINES START AND STOP ADDRESSES
0371                    *            AND PERFORMS A BASIC READ/WRITE TEST
0372                    *
0373                    *       PROGRAM DESCRIPTION
0374                    *            THIS TEST WRITES TO THE FIRST LOCATION AND
0375                    *            THEN READS IT BACK.  IF IT IS CORRECT THEN
0376                    *            THE INVERSE DATA IS WRITTEN TO THE LOCATION.
0377                    *            IF IT IS CORRECT THEN THIS IS THE START ADD-
0378                    *            RESS.  IF IT IS NOT CORRECT THEN THE NEXT
0379                    *            LOCATION IS CHECKED ECT. UP TO >F000.
0380                    *               ALL CONTIGUOUS BLOCKS OF MEMORY ARE
0381                    *            RECORDED BETWEEN >2000 AND >F000.
0382                    *
0383                    *            THIS CAN BE USED TO CHECK THE SETTING OF THE
0384                    *       ADDRESS SWITCHES ON THE TM990-203 BOARD.
0385                    *
0386                    *
0387                    TEST1
0388 018C 0204          LI    R4,>F000          R4 HAS SEARCH END ADD.
     018E F000
0389 0190 0701          SETO  R1                DISTURB DATA = >FFFF
0390 0192 04C2          CLR   R2                DATA = 0
0391 0194 0203          LI    R3,>FFFE          R3 HAS SEARCH START ADD.
     0196 FFFE
0392                    *                       MINUS 2
0393 0198 04C6          CLR   R6                CLR FIRST FLAG
0394 019A 04C8          CLR   R8                CLEAR BLOCK FOUND FLAG
0395                    FIND
0396 019C 05C3          INCT  R3                INCREMENT ADDRESS
0397 019E C1D3          MOV   *R3,R7            SAVE DATA FOR RESTORE
0398 01A0 CCC2          MOV   R2,*R3+           MOVE DATA TO LOCATION
0399 01A2 C013          MOV   *R3,R0            SAVE DATA HERE ALSO
0400 01A4 C4C1          MOV   R1,*R3            DISTURB DATA BUS
0401 01A6 0643          DECT  R3                RESTORE ADDRESS OF LOCATION
```

TM990/203 DYNAMIC RAM SET UP ROUTINE

```
0402 01A8 C153          MOV  *R3,R5          READ DATA BACK
0403 01AA 8085          C    R5,R2           IS DATA VALID ?
0404 01AC 1610          JNE  ENDCK1          IF NOT CK IF LAST LOC.
0405 01AE CCC1          MOV  R1,*R3+         MOV INV DATA TO LOC.
0406 01B0 C4C2          MOV  R2,*R3          DISTURB DATA BUS
0407 01B2 0643          DECT R3              RESTORE LOC. ADDRESS
0408 01B4 C153          MOV  *R3,R5          READ DATA BACK
0409 01B6 8045          C    R5,R1           IS DATA VALID ?
0410 01B8 160A          JNE  ENDCK1          IF NOT CHECK FOR END ADD.
0411 01BA CCC7          MOV  R7,*R3+         RESTORE DATA HERE
0412 01BC C4C0          MOV  R0,*R3          AND HERE
0413 01BE 0643          DECT R3              AND RE-ADJUST R3
0414 01C0 8103          C    R3,R4           UPPER LIMIT REACHED ?
0415 01C2 1405          JHE  ENDCK1          IF SO, QUIT
0416 01C4 C186          MOV R6,R6            OTHERWISE,SEE IF START ADD.
0417              *                          HAS BEEN FOUND
0418 01C6 16EA          JNE  FIND            IF SO, GO ON
0419 01C8 0706          SETO R6              OTHERWISE,SET START ADDRESS
0420 01CA C243          MOV  R3,R9           SET START ADDRESS
0421 01CC 10E7          JMP  FIND            AND GO FIND END
0422              ENDCK1
0423 01CE C8C0          MOV  R0,@2(R3)       RESTORE DATA
     01D0 0002
0424 01D2 C186          MOV  R6,R6           CHECK IF START
0425              *                          ADDRESS FOUND YET
0426 01D4 1607          JNE  ENDADD          IF SO R3 HAS END ADD.
0427 01D6 8103          C    R3,R4           DONE YET ?
0428 01D8 1AE1          JL   FIND            IF NOT, KEEP GOING !
0429 01DA C208          MOV  R8,R8           HAS ANY MEMORY BEEN FOUND ?
0430 01DC 1612          JNE  EOT1            IF SO CK END OF TEST 1
0431 01DE 2FA0          PRNT @MSG16          IF NOT PRINT NONE FOUND
     01E0 021A'
0432 01E2 100F          JMP  EOT1            GO TO EOT1
0433              ENDADD
0434 01E4 C283          MOV  R3,R10          SET END ADDRESS
0435 01E6 C208          MOV  R8,R8           IF NOT FIRST BLOCK, DON'T
0436 01E8 1602          JNE  PASS            PRINT ALL OF MESSAGE
0437 01EA 2FA0          PRNT @MBOUND         PRINT 'MEMORY FOUND'
     01EC 0206'
0438              PASS
0439 01EE 0708          SETO R8              SET BLOCK FOUND
0440 01F0 2E89          HEXO R9              PRINT 'START ADD'
0441 01F2 2FA0          PRNT @TO             PRINT 'TO'
     01F4 0215'
0442 01F6 2E8A          HEXO R10             PRINT 'END ADDRESS'
0443 01F8 2FA0          PRNT @CRLF           PRINT A CAR.RET. AND LINE FEED
     01FA 0189'
0444 01FC 04C6          CLR  R6              CLEAR START ADDRESS FOUND FLAG
0445 01FE 8103          C    R3,R4           DONE YET ?
0446 0200 1ACD          JL   FIND            IF NOT, CONTINUE!
0447              EOT1
0448              *                                 OTHERWISE
0449 0202 0460          B    @TIBUG          EXIT
     0204 0080
```

```
0450 0206   0D  MBOUND BYTE >D,>A
     0207   0A
0451 0208   4D         TEXT 'MEMORY FROM '
0452 0214   00         BYTE 0
0453 0215   20  TO     TEXT ' TO '
0454 0219   00         BYTE >0
```



```
0456            *
0457            ***************************************************************
0458 021A   4E  MSG16  TEXT 'NO RAM MEMORY FROM >2000 TO >F000'
0459 023B   0D         BYTE >D,>A,>0
     023C   0A
     023D   00
0460 023E   50  PWRUP  TEXT 'POWER UP IN PROGRESS'
0461 0252   0D         BYTE >D,>A,>0
     0253   0A
     0254   00
0462                   END
```



```
' AGAIN1  011E     BA9901  0100     BLWPC   0420    ' CHARCK  012E
' CRLF    0189    ' EINT2  0162    ' ENDADD 01E4    ' ENDCK1  01CE
' ENDPAR  00CA    ' EOI2   015E    ' EOINT2 0154    ' EOT1    0202
' FIND    019C    ' FIXRET 00BA    X HEXO    000A    ' INT1    0058
' INT1PC  005C    ' INT2   0102    ' INT2CK 0170    ' INT2GO  0126
' INT2PC  0106    ' INTMSG 00D2      INTWS  FF20    ' MBOUND  0206
' MIX     012E    ' MIX1   0136    ' MSG16  021A    ' NONE    017D
' PASS    01EE    ' PC     00F2    ' POWER  0000    X PRNT    000E
' PWRUP   023E      R0     0000      R1     0001      R10     000A
  R11     000B      R12    000C      R13    000D      R14     000E
  R15     000F      R2     0002      R3     0003      R4      0004
  R5      0005      R6     0006      R7     0007      R8      0008
  R9      0009      RTWPC  0380    ' SETP   0050    ' SETPAR  004E
' SPACE   00FE    ' ST     00F8    ' TEST1  018C      TIBUG   0080
' TO      0215      TSTWS  FF00    ' WS     00EC    ' WSOUT   00A0
' WSOUT1  00A4
```

0000 ERRORS

AGAIN1 0305 0345
BA9901 0096 0130
BLWPC 0099 0166
CHARCK 0316
CRLF 0363 0225 0226 0246 0310 0349 0443
EINT2 0348 0321 0342
ENDADD 0433 0426
ENDCK1 0422 0404 0410 0415
ENDPAR 0251 0248
EOI2 0346 0344
EOINT2 0340 0328
EOT1 0447 0430 0432
FIND 0395 0418 0421 0428 0446
FIXRET 0242 0240
HEXO 0090
INT1 0208 0168
INT1PC 0210 0209
INT2 0294 0175
INT2CK 0357 0299
INT2GO 0308 0338
INT2PC 0296 0295
INTMSG 0259 0214 0298
INTWS 0083 0209 0295
MBOUND 0450 0437
MIX 0311 0339
MIX1 0325 0333
MSG16 0458 0431
NONE 0360 0347
PASS 0438 0436
PC 0266 0219
POWER 0115
PRNT 0091
PWRUP 0460 0117
R0 0232 0243 0309 0337 0399 0412 0423
R1 0164 0165 0165 0167 0169 0171 0172 0173 0173
0174 0176 0177 0216 0218 0220 0221 0223 0224
0234 0237 0300 0329 0331 0343 0343 0389 0400
0405 0409
R10 0434 0442
R12 0130 0212 0302 0304
R13 0215
R2 0166 0167 0174 0230 0247 0247 0249 0306 0326
0327 0329 0331 0336 0390 0398 0403 0406
R3 0168 0169 0175 0176 0189 0191 0191 0192 0228
0236 0238 0239 0241 0307 0327 0391 0396 0397
0398 0399 0400 0401 0402 0405 0406 0407 0408
0411 0412 0413 0414 0420 0423 0427 0434 0445
R4 0170 0171 0177 0229 0301 0335 0341 0341 0388
0414 0427 0445
R5 0402 0403 0408 0409
R6 0393 0416 0416 0419 0424 0424 0444
R7 0397 0411
R8 0215 0227 0227 0234 0394 0429 0429 0435 0435
0439
R9 0420 0440

```
 TXXREF          2.3.0  78.244     00:05:45     01/01/00     PAGE 0002

RTWPC    0098       0170
SETP     0190       0193
SETPAR   0188       0119  0120  0252  0350
SPACE    0272       0235  0334
ST       0269       0222
TEST1    0387       0181
TIBUG    0097       0449
TO       0453       0441
TSTWS    0082       0116
WS       0263       0217
WSOUT    0231       0250
WSOUT1   0233       0245

THERE ARE 0058 SYMBOLS
```

APPENDIX E

PARTS LIST

TABLE E-1. PARTS LIST FOR TM 990/203 MODULE

Symbol	Description
C1-C4, C6-C9, C11, C12, C15-C28, C31-C37, C41, C42, C44-C52, C54, C57-C90	*Capacitor, 10 uF, 50 V
C5, C29, C38	Capacitor, 68 pF, 15 V
C13	Capacitor, 1200 pF, 25 V, 2%
C14	Capacitor, 1000 pF, 25 V, 2%
C39, C53	Capacitor, 22 uF, 35 V
C92	Capacitor, 330 pF, 25 V, 2%
C93	Capacitor, 2200 pF, 25 V, 2%
C94	Capacitor, 680 pF, 25 V, 2%
CR1	Diode, Silicon Zener (TI E7918, Motorola .4MS.1AZ1)
DS1	Optoelectric Device, TIL200
R1, R3-R6, R11, R17, R19, R21, R34	Resistor, 10 Kohms, 1/4 W, 5%
R2, R7, R18, R22, R9	Resistor, 1 Kohms, 1/4 W, 5%
R8	Resistor, 4.7 Kohms, 1/4 W, 5%
R9	Resistor, 1.5 Kohms, 1/4 W, 5%
R10, R12, R23	Resistor, 330 ohms, 1/4 W, 5%
R13	Resistor, 453 ohms, 1/4 W, 1%
R14	Resistor, 324 ohms, 1/4 W, 1%
R20	Resistor, 680 ohms, 1/2 W, 5%
R24, R27-R31	Resistor, 680 ohms, 1/4 W, 5%
R25, R26	Resistor, 15 ohms, 1/4 W, 5%
R35, R37, R39	Resistor, 249 ohms, 1/4 W, 1%
R40	Resistor, 422 ohms, 1/4 W, 1%
S1, S2, S3, S4	Switch, Dual In Line, 4 position
U1, U13, U18, U33	IC, SN74S85N
U2	IC, SN74LS125N
U3	IC, SN74S112N
U4, U6	IC, SN74LS241N
U7	IC, SN74S240
U8	IC, SN74S03N
U9, U15, U31	IC, SN74S114N
U10, U37	IC, SN74LS393N
U11, U40	IC, SN74S132N
U12, U26	IC, SN74S09N
U14, U25	IC, SN74S51N
U16, U44	IC, SN74S74N
U17	Resistor pack, 10 Kohms, 16 pins
U19	IC, SN74LS00N

*Exceptions for model TM 990/203-21A are listed in Table E-3.

TABLE E-1. PARTS LIST FOR TM 990/203 MODULE (CONCLUDED)

Symbol	Description
U20, U27	IC, SN74S280N
U21, U28	IC, SN74LS245N
U22, U30, U32, U39	IC, SN74S00N
U29	IC, SN74S11N
U35, U36, U41, U42	IC, SN74S373
U43	IC, SN74LS74N
XU5, XU45-XU78	*Socket, 16 pin IC

TABLE E-2. MEMORY DEVICES

Symbol	Description	Model
U45-U78	IC, 16 K RAM, TMS 4116	TM 990/203-23
U45-U61	IC, 16 K RAM, TMS 4116	TM 990/203-22
U45-U78	IC, 4K X 1 RAM, TMS 4027	TM 990/203-21
U45-U61	IC, 8K X 1 RAM, TMS 4108	TM 990/203-21A

TABLE E-3. EXCEPTIONS TO TABLE E-1 FOR MODEL TM 990/203-21A

Symbol	Description
C1-C4, C6-C9, C11, C12, C15-C28, C31-C37, C41, C42, C54, C57-C74	Capacitor, 10 uF, 50 V (Deletes C44-C52 and C75-C90)
XU5, XU45-XU61	Socket, 16 pin IC (Deletes XU62-XU78)

*Exceptions for model TM 990/203-21A are listed in Table E-3.

APPENDIX F

TM 990 MEMORY MAPPING

F-1 GENERAL

Although the TM 990/203 dynamic RAM memory board has the capability of selecting either of the two memory banks (A and B) as explained in subsection 2.4.3.3, this does not reflect the memory mapping architecture of the TM 990 product line. This appendix outlines the TM 990 memory mapping structure.

The TM 990 address bus contains 20 separate lines, a basic set of 16 (A0 through A15) and an extended set of four (XA0 through XA3). (The address bus is covered in detail in paragraph F.3 of this appendix.) This 20-bit address field permits the TM 990 system bus to support a memory capacity of up to one megabyte (1,048,576 bytes) with memory mapping. This memory can be made up of masked ROM, field-programmable ROM (PROM), erasable PROM (EPROM), static RAM (SRAM), and dynamic RAM (DRAM). The systems designer has the flexibility to choose the combination which best meets his application. The TM 990 product family provides a series of processor and memory expansion modules which can be combined to form the required configuration. Since the address space is shared by processor on-board memory, expansion memory and memory mapped I/O, care must be exercised when configuring a system not to overlap addresses on the various functions, except in specific cases where bus conflicts are avoided (for example it might be desireable to write to a mapped I/O device and a RAM location simultaneously).

F-2 PAGED MEMORY MAPPING

For most systems applications, the 64K-byte memory capacity addressable by the basic set of address lines A0 through A15 will be more than adequate. For those systems requiring additional memory capacity, the total addressable memory can be increased to one megabyte by use of memory mapping. Memory mapping is a technique which enables the processor's basic 16-line address field to be expanded into a 20-line address field. This process is illustrated in Figure F-1.

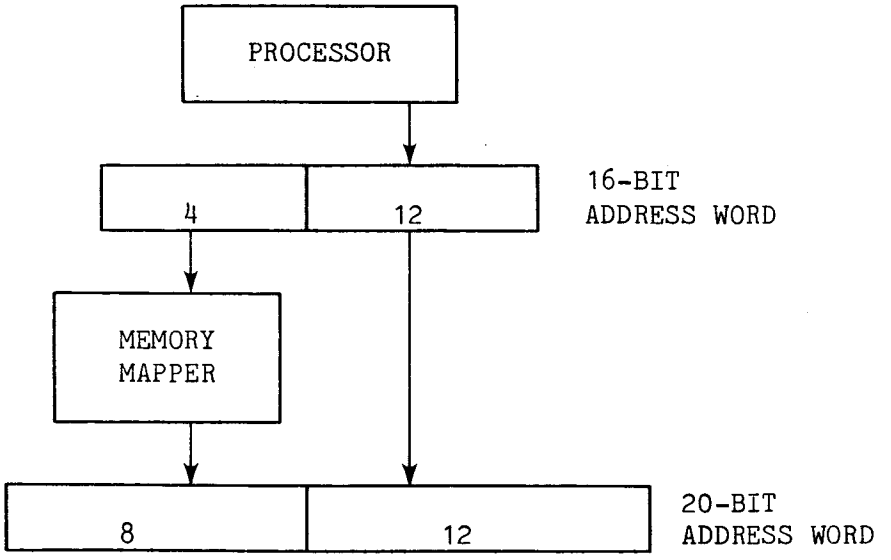


FIGURE F-1. PAGED MEMORY MAPPING

The four most significant bits of the processor's basic address field are used by the memory mapper to access one of sixteen 8-bit codes stored within the mapper. This 8-bit code from the mapper is put onto the extended memory address bus to form the most significant part of the extended memory address. The least significant 12 bits of the extended memory address are the unaltered least significant 12 bits of the basic address from the processor. Figure F-2 shows the structure of this extended memory address word.

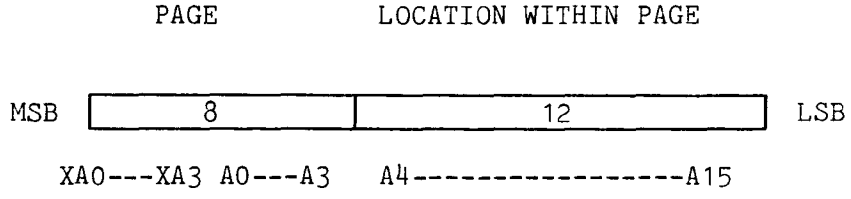


FIGURE F-2. EXTENDED MEMORY ADDRESS WORD

For each of the 8-bit codes obtainable from the memory mapper, there are 2^{12} or 4096-byte addresses available to the processor. Each set of 4096 (4K) addresses is considered to be one page of memory. Changing the 8-bit code from the mapper is equivalent to changing the page of memory. The 8-bit field from the memory mapper allows up to 256 pages in the system. The four most significant address bits from the processor, however, can address only one out of 16 pages at a time. This is illustrated in Figure F-3.

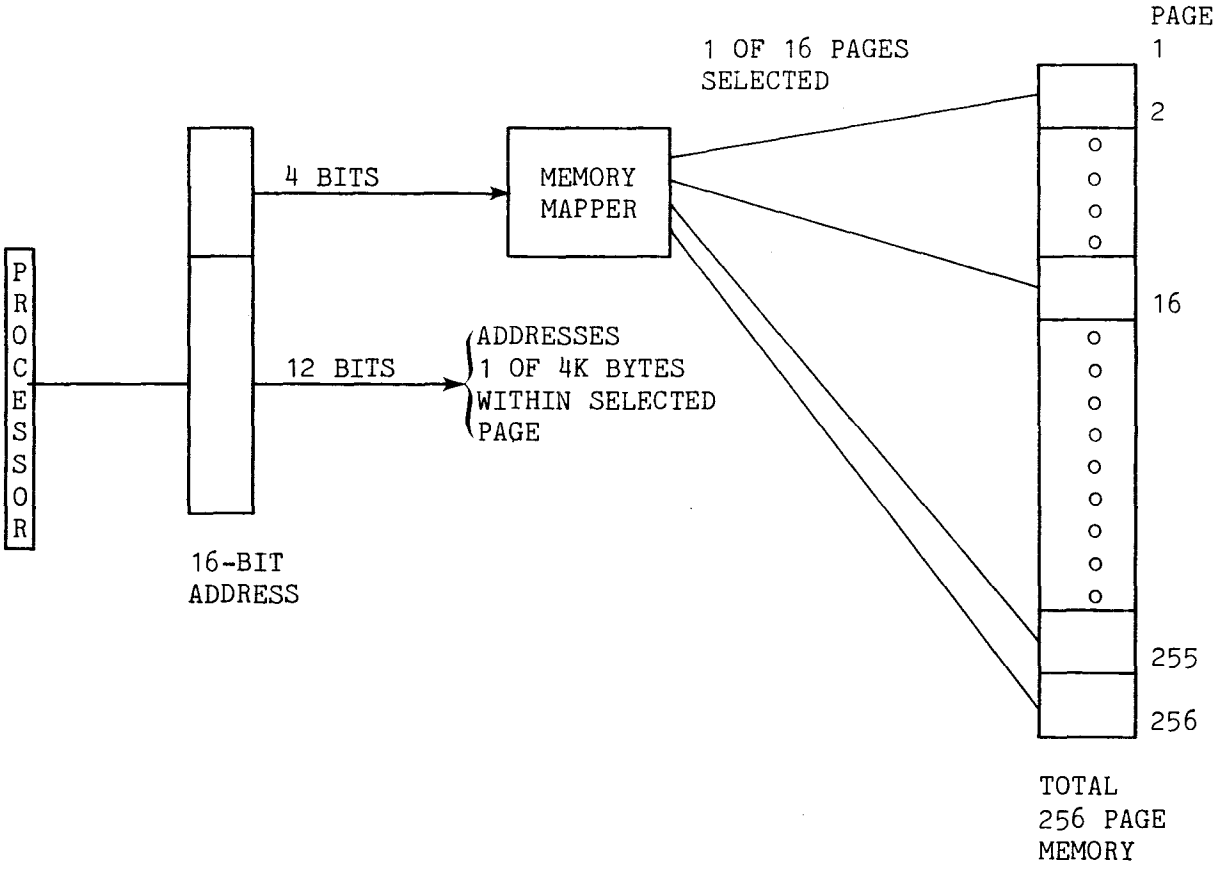


FIGURE F-3. MEMORY ADDRESSING WITH PAGED MEMORY MAPPING

To utilize the entire 256 pages of memory the 16 8-bit codes of the memory mapper are dynamically altered by the processor under software control.

F-3 ADDRESS BUS

The address bus consists of 20 three-state address lines. These 20 lines are comprised of a set of 16 basic address lines, A0 through A15, and a set of four extended address lines XA0 through XA3. The address bus is used to access memory locations, CRU bits and for memory mapped I/O.

For most applications, a 16-bit byte address is formed with A0 as the most significant bit (MSB) and A15 as the least significant bit (LSB). These 16 bits permit a processor to address up to 64K bytes. For word addresses, line A15 is held low and A14 becomes the LSB. A processor can address up to 32K words. The four extended address lines permit a processor to use memory mapping to address up to 1M bytes. Lines XA0 (MSB) and XA3 (LSB) form the most significant part of the 20-line address bus.

Memory-mapped I/O permits TM 990 I/O modules to be addressed (for data transfer) in the same manner as a memory location; data transfer takes place on the TM 990 data bus. All 20 address lines may be used to access memory mapped TM 990 I/O modules; normally these modules should not have the same addresses as memory locations. Most TM 990 I/O modules, however, because of cost, simplicity and flexibility will use the Communications Register Unit (CRU). Each bit in the CRU is addressed individually or in groups by the processor, using the 12-bit address field formed with address lines A3 through A14.

All 20 address lines are controlled by the master processor unless disabled by the activation of HOLD-. The address lines are not terminated on the system backplane.

F-4 MEMORY CONTROL SIGNALS

MEMEN- Memory Enable: When active (low), it indicates that the address bus contains a memory address, and that a memory access is in progress. MEMEN- may remain active for several consecutive memory cycles. It is inactive (high) during CRU cycles. MEMEN- is a three-state signal controlled by the master processor unless disabled by activation of HOLD-. MEMEN is terminated on the system backplane.

MEMCYC- Memory Cycle: When active (low), it indicates that a memory cycle is in progress. MEMCYC- is activated one BUSCLK cycle after MEMEN- is made active, and remains so only for a single memory cycle. MEMCYC- is normally released on the second BUSCLK- following the receipt of the READY signal. A memory cycle can be extended, however, by holding MEMCYC- active. MEMCYC- is a three-state signal controlled by the master processor unless disabled by activation of HOLD-. MEMCYC- is terminated on the system backplane.

DBIN Data Bus In: When active (high), it indicates that the processor has disabled its output buffers to allow memory to place read data on the data bus during MEMEN-. DBIN remains low in all cases except during a memory read cycle. It can be used in conjunction with an active MEMEN- signal to indicate the data direction during a memory cycle. (When high, a read cycle is in progress, and when low, a write cycle is in progress.) DBIN is a three-state signal controlled by the master processor unless disabled by activation of HOLD-. DBIN is terminated on the system backplane.

WE- Write Enable: When active (low), it indicates that data to be written into memory is present on the data bus. WE- is a three-state signal controlled by the master processor unless disabled by activation of HOLD-. WE- is terminated on the system backplane.

READY Ready: When active (high), this indicates that memory will be ready to complete its read or write operation during the next BUSCLK- cycle. When a not-ready condition is indicated during a memory operation, the processor suspends operation and enters a wait state until the memory indicates that it is ready. READY is driven by an open collector device and is not terminated on the system backplane.