3.2 Hardware Description - Memory

There are two types of memory expansion modules available with the Technico system - RAM and EPROM. Each of these modules is described in the following subparagraphs.

3.2.1 Hardware Description - Static Memory Module

The T99ST Static Memory Module is a self-contained static memory sstem. It contains all necessary address decoding and control circuitry for direct interface to the T99SS CPU Module. The T99ST memory board features 4045, or 2114 1Kx4 RAM memory chips. The 4045, 2114's have an access time of <450 nanoseconds. This is more than adequate response time for a 9900 microprocessor system running at 3.3 MHz.

The storage capacity of the board is 16K bytes, or 32K bytes, depending on the number of memory chips installed. Address selection allows the user to specify where the memory system starts in the memory address space. The starting address is selectable in 2K byte (1K word) increments by use of convenient DIP switches. The ending address can also be selected in increments of 2K bytes (1K words).

The T99ST write protect feature allows the user to inhibit writing into static RAM in 2K byte (1K word) increments.

A board select input is available to the user to allow expansion of the system memory beyond 64K bytes (in 32K byte increments). Using the CRU output bits from the T99SS CPU Module the user can expand the memory up to 425,984 bytes. To power a memory system of this size would require a +5VDC power supply of 78 amps for the memory alone, if all boards were maintained at full power.

The T99ST memory board is desmgned to allow the addition of a battery back up circuit. In addition to the 18 pin 2114, 4045 RAM chips the user has the option of using the TI 4047, 20 pin RAM chips. For battery operation, Synertek 2114LV or the TI 4047 RAM chips are recommended.

3.2.1.1 Data Access Control

U79A, U79B and the associated gates form the data access control circuitry for the T99ST memory module. This circuitry allows the use of fully static RAM chips or edge triggered static RAM chips.

When signal /MEMEN is low, the output of U55 (pin 3) is high. This releases the reset (pin 13) of U79, and also sets the D input of U79 (pin 2). On the next transistion of /Q3 the output of U55 (pin 11) will go high. This will cause U79 (pin 9) to be clocked high and U79 (pin 6) to be clocked low because signal /Q2 is high and so U58 (pin5) is high. This enables the selected memory board. On the next /Q3, U79 (pin 9) will go low which partially enables U58 (pin 4). The next /Q2 that occurs causes U58 (pin 6) to go low forcing U79 (pin 6) back high again.

3.2.1.2 Memory Buffer Control

The memory buffer control is performed by U77 and U78. If the output of U77 (pin 6 or pin 3) is low, data can be read or written into the memory. The direction of data flow is controlled by DBIN. If DBIN is high the output of U78 (pin 12) will be low enabling U77 (pin 6) and allowing data to be read from memory. Since DBIN is high U77 (pin 2) will be high forcing U77 (pin 3) high thereby disabling the write buffer. When signal DBIN is low U78 (pin 13) will be low forcing U77 (pin 6) high thus disabling the read buffers. Since signal DBIN is low U77 (pin 2) will be low and U77 (pin 3) will go low allowing data to be written into the memories.

3.2.1.3 Memory Bank Selections

U35 controls which bank of 2K bytes of memory are selected by the address presented to the T99ST memory board. If U35 is enabled (pin 18, pin 19 low) the bank selected depends on the 4 binary inputs (pins 20 thru 23). These four bits can select any one of the 16 2K bytes on the memory board. U34 and U36 are two 8 position dip switches which can disable a particular bank. To enable a bank the appropriate switch must be closed. When an output of U35 goes low U34 or U36 will pass this low to one of the four quad input gates U33 or U38. If any input of U37 or U38 is low it will force a low on the output which will cause one of the inputs to U37 (a quad input and-gate) to go low which will cause U37 (pin 6) to be low thus causing a memory select to occur(/MEMSEL low). Signal /MEMSEL partially enables the buffer control circuitry for the bidirectional data bus in the write protect circuit. Signal /MEMSEL also goes to U80. The outputs of any open switched U34 or U36 are pulled up by resistors to 5 volts causing that line to be high. This high inhibits the decoding network U33, U38, and U37. Therefore the output buffers or input buffers will not be enabled for that particular bank of memory.

3.2.1.4 Selection of Memory Board Starting Address

By properly setting DIP switches S1 through S5 of U56, the user can specify a starting address offset of the form S

=2048*Z, Thus the starting address can be selected in 2K byte (1K word) increments.

The address offset circuit will add the starting address offset to the address sent in by the CPU. The resulting sum is then applied to the rest of the memory board as its address. The starting address offset is a 2's complement number of the form:

```
XXXX   X000   0000   0000
```

where the X's represent the 5 bit binary number Z.

Since the offset is a two's complement number, determination of the proper address offset can be confusing. The proper procedure is to determine what offset will produce zero when added to the desired starting address. This is, of course, the two's complement of the start. For example, if a starting address of >8800 is desired, the offset is the two's complement of this or >7800 since >7800+>8800 = 0000 (ignore final carry out).

U57 is 4-bit binary full adder. U55 is a quad 2 input Exclusive-OR gate. U55B and U55C together form a 1-bit full adder ignoring the output carry . The value of Z is entered into the switches S1 to S5 according to the bit weights:

16	8	4	2	1
S6	S1	S2	S8	S7

If the starting address is selected to be greater than >8000 the memory will "wrap around" to zero. For example, if the start is >E000 then address >E000+>2000 = >10000 or >0000. This may cause an overlap with other areas of memory (i.e. T99SS CPU Module). This will cause memory address conflicts which will cause hardware failures. Any overlapping banks must be disabled by opening the appropriate switches on U34, U36. This is necessary even if the corresponding RAM IC's are not installed, otherwise the data buffers would be improperly enabled. For example, if the offset is >7800, then the starting address is >8800. This starting address is subtracted from incoming addresses to form the internal address applied to the memories on the board. Thus external addresses 8800 through FFFF map into internal addresses 0000 through 77FF, and external addresss 0000 through 07FF map into internal addresses 7800 through 7FFF. If other memory in the system will respond to any of these addresses, then the memory board must be inhibited from having its data biffers respond to those external addresses . Otherwise the two memories would be contending for the bus. As discussed earlier, the decoding can be inhibited in 2K byte blocks. In this example, to stop responding to external addresses 0000 to 07FF, and E800 to FFFF, the last 4 banks of 2114 would be removed and switches U34 S1, S2, S3, & S4 would be opened, disabling internal addresses 6000 through 7FFF (external addresses E800 through FFFF and 0000 through 07FF). The top 6K of the address space of the memory board is then not being used.

3.2.1.4 Write Protect

The T99ST memory module write protect ciruitry consists of a number of IC's: two 74LS259 addressable latches, two 74LS251 multiplexers, and several gates.

To write protect a 2K byte bank of memory, the user must set the CRU output bits connected to signals /WRT DIS and /ENB WRT CLK low. Now, by reading from the 2K bank of memory to be protected, a zero will be stored in the write enable latch for that bank. After the read, both /WRT DIS and /ENB WRT CLK bits must be set high again.

To enable writing to a bank of memory, /ENB WRT CLK must be set low, /WRT DIS set high and a read must be done from the protected bank. After the read, /ENB WRT CLK must be set high again.

U59 and U81 are 74LS259 addressable latches. Each of the output bits of U59 and U81 control the write protection of a 2K bank of memory on the 16K module. If an output bit is a high or logic one, then that bank is write protected. If the output bit is low or logic zero, then that bank is not write protected.

When a reference is made to one of the 2K banks of memory, one of the 16 output bits will be multiplexed to U58 (pin 13) or U80 (pin 12). This is controlled by the four address outputs of U57 (7483 adder). When U57 (pin 15) is low U82 will be enabled and U60 (pin 14) will be disabled. The address inputs of U82 (pins 9, 10, 11) will determine which of the 8 inputs will be passed to the output(pin 5). If the output is low, U80 (pin 12) will be low.$ If a write is to be performed, U80 (pin 13) will go low. With pin 12 low, the output of U80 (pin 11) will go low. This will force the output of U37 (pin 8) to be low. With U91 (pin 10) low, the output of U91 (pin 8) will go low and the write will occur. A high on the output of U60 or U80 will inhibit the write signal from being gated through U58 or U80 thus protecting the memovy from a write.

To write protect a bank of memory, signals /WRT DIS and /ENB WRT CLK are set low. This will force a high on pmn 13 of U59 and U81 (the 'D' input for the addressable latches). Also, pin 10 of U80 will be low. When the output of U80 (pin 6) goes low and then back high again, the clock for U59 or U81 will be togled - depending on the output of U57. U80 (pin 6) will go low and back high each time the signals /MEM SEL and /Q3 both go low at the same time. To clear the protect bit the same sequence must occur with signal /WRT DIS high this time.

Note that each time the system is reset, all of the write protect bits ere cleared.

3.2.1.5 Board Select

The T99ST memory board can be entirely disabled or enabled allowing multiple 32K banks of memory boards to be used in a system. The board disable signal is one of the CRU output bits from J1. This bit is user selectable and must be jumpered to the board disable pad on the PC board. Whenever the signal /BOARD DISABLE is low U78 (pin 5) will be low forcing U78 (pin 6) high. Anytime U78 (pin 6) is high the Q1 outout U35 will be disabled, thus disabling any references to the memory board. To turn the board back on signal /BOARD DISABLE must be returned high.

3.2.1.6 Memory Power Distribution

+5 volts to the memory chips is brought in on two different sets of pads near J34 (a 34 pin ribbon cable connector). This is done to allow implementation of a battery back-up system to power the board. If a battery back-up system is not used and 2114 RAM's are used +5V from the PC board edge connector must be jumpered to the pad near J34 to provide +5 volts to all of the 2114 memory chips. Care must be taken when inserting the 2114 memories into the board. The board is assembled using 20 pin sockets for each of the memory locations. The IC's must alway be inserted at the bottom of the socket. When using a 2114 or a 4045 memory chip then power will be brought in on pin 18 of the IC. When using a TMS 4047 memory chip (which is a 20 pin chip) pin 0 is an output enable pin & pin 19 is the VCC pin for the peripheral drivers on the memory chip. When using the 4047 as a memory in the battery back-up mode, pin 18 can drop to 0V, but pin 19 must be maintained at a minimum of 2.5 volts to maintain data integrity in the memory.

3.2.1.7 Memory Test

Three memory test programs are provided. The first program writes data into the rAM and then enters a loop reading to verify the correct operation of the RAM. When an error occurs RO will be incremented by one. The breakpoint and snap routine may be used to examine RO in the error loop.

The second program will write the memory address into each memory location and then verify that they were written by reading each location.

The last program will perform a comprehensive test of the memory. It takes about 9 minutes to complete a pass. Load the program starting at location 100 Hex with the alter command. Start the program by typing G 100 (CR). It will request the START and END address. Should an error be found it will print:

```
Error      Data       Errored
Address    Read       Bits=1's
XXXX       XXXX       XXXX
```

If it passes the test, it will print TEST COMPLETED.

```
                              TITL 'MEMORY TEST ONE'
  0000                        DREG
  0800              START     EQU  >800              ; START OF MEMORY
  87FE              ENDM      EQU  >87FE             ; END OF MEMORY
  0000 04C0         TST100 CLR  R0                   ; RESET ERROR COUND
  0002 0202 FFFF           LI   R2,>FFFF             ; R2=PATTERN
  0006 0201 0800           LI   R1,START             ; R1=START ADDRESS
  000A CC42         TST110 MOV  R2,*R1+              ; STORE DATA
  000C 0281 87FE           CI   R1,ENDM              ; END? (SET TO LAST ADDRESS)
  0010 12FC                JLE  TST110               ; NO- CONTINUE
  0012 0201 0800           LI   R1,START             ; R1=START ADDRESS
  0016 8C42         TST120 C    R2,*R1+              ; DATA OK?
  0018 1604                JNE  TST140               ; NO- ERROR
  001A 0281 87FE    TST130 CI   R1,ENDM              ; END? (SET TO LAST ADDRESS)
  001E 12FB                JLE  TST120               ; NO- CONTINUE
  0020 10F4                JMP  TST110               ; START ANOTHER PASS
  0022 0580         TST140 INC  R0                   ; UPDATE ERROR COUNT
  0024 2C00                XOP  0,0                  ; ERROR- RETURN TO MONITOR
  0026 10F9                JMP  TST130               ; CONTINUE ON
  0028                     END


 87FE ENDM     0000 R0      0001 R1      *000A R10     *000B R11
*000C R12     *000D R13    *000E R14     *000F R15      0002 R2
  003 R3      *0004 R4     *0005 R5      *0006 R6      *0007 R7
*0008 R8      *0009 R9      0800 START   *0000 TST100   000A TST110
 0016 TST120   001A TST130  0022 TST140
EDIT/ASM/LOAD?
```

```
                            TITL 'MEMORY TEST TWO'
 0000                       DREG
 0800             START     EQU  >800              ; START OF MEMORY
 87FE             ENDM      EQU  >87FE             ; END OF MEMORY
 0000 0201 0800   TST200 LI      R1,START          ; R1=START ADDRESS
 0004 CC41        TST210 MOV     R1,*R1+           ; STORE DATA
 0006 0281 87FE          CI      R1,ENDM           ; END? (SET TO LAST ADDRESS)
 000A 12FC               JLE     TST210            ; NO- CONTINUE
 000C 0201 0800          LI      R1,START          ; R1=START ADDRESS
 0010 8C41        TST220 C       R1,*R1+           ; DATA OK?
 0012 1604               JNE     TST240            ; NO- ERROR
 0014 0281 87FE   TST230 CI      R1,ENDM           ; END? (SET TO LAST ADDRESS)
 0018 12FB               JLE     TST220            ; NO- CONTINUE
 001A 10F4               JMP     TST210            ; START ANOTHER PASS
 001C 0580        TST240 INC     R0                ; UPDATE ERROR COUNT
 001E 2C00               XOP     0,0               ; ERROR- RETURN TO MONITOR
 0020 10F9               JMP     TST230            ; CONTINUE ON
 0022                    END


 87FE ENDM      0000 R0      0001 R1      *000A R10     *000B R11
*000C R12      *000D R13    *000E R14     *000F R15     *0002 R2
*0003 R3       *0004 R4     *0005 R5      *0006 R6      *0007 R7
*0008 R8       *0009 R9      0800 START   *0000 TST200   0004 TST210
 0010 TST220    0014 TST230  001C TST240
EDIT/ASM/LOAD?
```

```
0000                MEMTST IDT
                    *
                    *   MEMORY TEST PROGRAM (SELF RELOCATING FORM)
                    *
0000                       DREG
                           REF  RDNUMB,TYPEN,TYPEWD,TYPEX
                    * THE VALUES FOR VER 3 AND VER 4 MONITOR ARE:
                    *
                    *      LABEL     V3      V4
                    *
                    *      RDNUMB   FE1E    FE22
                    *      TYPEN    FCB4    FCB8
                    *      TYPEWD   FDBA    FCBE
                    *      TYPEX    FC98    FC9C
                    *
                    *
                    *                                 ;  CHANGE NEXT 3 LINES OF CODE
                    *                                 ;    TO CHECK LOW MEMORY
00D0                RETURN EQU  >D0
00D2                STASH  EQU  >D2
                    *
0000 02E0 00B0      TEST   LWPI >B0
0004 06A0 0000             BL   @TYPEX              ; TO PUT ADDR 'BASE' IN R11
0008 C3CB           BASE   MOV  R11,R15             ; R15 IS BASE REGISTER
                    *
000A 0201 0274             LI   R1,STARTM-BASE
000E 06AF 026A             BL   @PRINT-ASE(R15) ; PRINT 'START: '
0012 0205 0082             LI   R5,>82              ; FOR RDNUMB
0016 0201 0200             LI   R1,>200             ; DEFAULT START ADDRESS
001A 06A0 0000             BL   @RDNUMB             ; READ START ADDRESS
001E C381                  MOV  R1,R14              ; R14 = START ADDRESS
0020 06A0 0000             BL   @TYPEN
0024 0201 027C             LI   R1,ENDM-BASE
0028 06AF 026A             BL   @PRINT-BASE(R15) ; PRINT 'END: '
002C 0205 0082             LI   R5,>82              ; FOR RDNUMB
0030 0201 87FF             LI   R1,>87FF            ; DEFAULT END ADDRESS
0034 0241 FFFE             ANDI R1,>FFFE            ; END ADDR MUST BE WORD
0038 06A0 001C             BL   @RDNUMB             ; READ END ADDRESS
003C C181                  MOV  R1,R6               ; R6 = END ADDRESS
003E 06A0 0022             BL   @TYPEN
                    *
0042 0208 0016      MEM002 LI   R8,22               ; OK COUNTER
0046 C309           MEM004 MOV  R9,R12              ; SAVE LEFT OF SEED
0048 026A 0001             ORI  R10,1               ; FORCE RIGHT HALF ODD
004C C34A                  MOV  R10,R13             ; SAVE RIGHT HALF OF SEED
004E 0207 ABCD             LI   R7,>ABCD            ; GENERATOR FOR RANDOM #
                    *
                    *   SPRINKLE RANDOM NUMBERS OUT INTO MEMORY
                    *
0052 C0CE                  MOV  R14,R3              ; GET START ADDRESS
0054 06AF 025E      MEM010 BL   @RANDOM-BASE(R15)
0058 CCC9                  MOV  R9,*R3+             ; PUT OUT THE RANDOM #
005A 8183                  C    R3,R6
005C 12FB                  JLE  MEM010
                    *
                    *   FOOL AROUND A LITTLE TO GIVE REFRESH A CHANCE
                    *   TO FAIL . . .
                    *
```

```
005E 04C3              CLR   R3                ; FOR DIVIDE
0060 0204 FFFF         LI    R4,>FFFF          ; FOR MAX. DIVIDE
0064 3CEF 0292         DIV   @ONE-BASE(R15),R3         ; DIVIDE BY ONE
0068 C009              MOV   R9,R0             ; PUT RANDOM # IN R0
006A 0803              SRA   R3,R0             ; SHIFT R3 A RANDOM AMOUNT
006C 09C0              SRL   R0,12             ; RANDOM # BETWEEN 0 AND 15
006E 1304              JEQ   MEM025            ; SKIP DELAY IF IT IS 0
0070 0601       MEM020 DEC   R1                ; DELAY LOOP
0072 16FE              JNE   MEM020
0074 0600              DEC   R0                ; DECREMENT THE RAND #
0076 16FC              JNE   MEM020            ; REPEAT IF NON-ZERO
                *
                *  NOW GO CHECK TO SEE IF RANDOM NUMBERS STILL THERE
                *
0078 C24C       MEM025 MOV   R12,R9            ; GET LEFT SEED
007A C28D              MOV   R13,R10           ; GET RIGHT SEED
007C C0CE              MOV   R14,R3            ; GET START ADDRESS
007E 06AF 025E  MEM030 BL    @RANDOM-BASE(R15)         ; GET RANDOM # IN R
0082 C049              MOV   R9,R1
0084 2853              XOR   *R3,R1            ; XOR DATA ONTO RAND #
0086 1606              JNE   BAD               ; IF NOT ZERO, MEMORY IS BAD
0088 05C3       MEM040 INCT R3
008A 8183              C     R3,R6             ; AT END YET?
008C 12F8              JLE   MEM030            ; CONTINUE IF NOT
008E 0608              DEC   R8                ; DEC THE OK COUNTER
0090 16DA              JNE   MEM004            ; GO THRU LOOP 22 TIMES
0092 1010              JMP   MTEST             ; GO DO OTHER TESTS
                *
                *  PRINT ADDRESS OF BAD LOCATION, RANDOM #, AND
                *  (RANDOM #) XOR (BAD DATA)
                *
0094 C143       BAD    MOV   R3,R5             ; BAD MEMORY, SO TYPE:
0096 06A0 0000         BL    @TYPEWD           ; ADDRESS OF FAILING WORD
009A 2CAF 0280         OUT   @SPACE-BASE(R15)
009E C149              MOV   R9,R5
00A0 06A0 0098         BL    @TYPEWD           ; NUMBER THAT SHOULD BE THERE
00A4 2CAF 0280         OUT   @SPACE-BASE(R15)
00A8 C141              MOV   R1,R5
00AA 06A0 00A2         BL    @TYPEWD           ; XOR OF R9 AND MEMORY
00AE 06A0 0040         BL    @TYPEN
00B2 10EA              JMP   MEM040
                *
                *  GALLOPING 1/0 MEMORY TEST
                *
00B4 C1C6       MTEST  MOV   R6,R7             ; STORE END ADDRESS + 2
00B6 05C7              INCT R7
00B8 0708              SETO R8                 ; GALLOPING 1
00BA 06AF 0122         BL    @GALPAT-BASE(R15)
00BE 04C8              CLR   R8                ; GALLOPING 0
00C0 06AF 0122         BL    @GALPAT-BASE(R15)
                *
                *  ROTATING 1/0 MEMORY TEST
                *
00C4 06AF 01F2         BL    @TRAV-BASE(R15)   ; ROTATING 1
00C8 8000              DATA >8000
00CA 06AF 01F2         BL    @TRAV-BASE(R15)   ; ROTATING 0
00CE 7FFF              DATA >7FFF
                *
```

```
                  *  CHECKERBOARD MEMORY TEST
                  *
 00D0 0208 AAAA          LI   R8,>AAAA          ; FILL MEMORY WITH
 00D4 C0CE               MOV  R14,R3            ;    CHECKER BOARD
 00D6 0548        M10    INV  R8                ; INVERT PATTERN
 00D8 C4C8               MOV  R8,*R3            ; SAVE IT
 00DA 84C8               C    R8,*R3            ; STILL THERE?
 00DC 1302               JEQ  M15
 00DE 06AF 0242          BL   @TYPERR-BASE(R15)         ; NO - ERROR
 00E2 05C3        M15    INCT R3                ; GO TO NEXT ADDR
 00E4 8183               C    R3,R6             ; DONE?
 00E6 1AF7               JL   M10               ; NO - LOOP
 00E8 0643        M20    DECT R3                ; CHECK IN REVERSE
 00EA 84C8               C    R8,*R3            ; ... INVERTING AS WE GO
 00EC 1302               JEQ  M25
 00EE 06AF 0242          BL   @TYPERR-BASE(R15)         ; ERROR - REPORT
 00F2 0548        M25    INV  R8                ; INVERT PATTERN
 00F4 0553               INV  *R3               ; ... AND MEMORY
 00F6 84C8               C    R8,*R3            ; STILL OK?
 00F8 1302               JEQ  M28
 00FA 06AF 0242          BL   @TYPERR-BASE(R15)         ; NO - ERROR
 00FE 8383        M28    C    R3,R14            ; DONE?
 0100 1AF3               JL   M20               ; NO - LOOP
 0102 84C8        M30    C    R8,*R3            ; CHECK IN ASCENDING ORDER
 0104 1302               JEQ  M35               ; ... AGAIN INVERTING
 0106 06AF 0242          BL   @TYPERR-BASE(R15)         ; REPORT IF ERROR
 010A 0548        M35    INV  R8                ; INVERT PATTERN
 010C 0553               INV  *R3               ; ... AND MEMORY
 010E 84C8               C    R8,*R3            ; STILL OK?
 0110 1302               JEQ  M38
 0112 06AF 0242          BL   @TYPERR-BASE(R15)         ; NO - ERROR
 0116 05C3        M38    INCT R3                ; GO TO NEXT ADDR
 0118 8183               C    R3,R6             ; DONE?
 011A 1AF3               JL   M30               ; NO - LOOP TILL DONE
                  *
                  *  END OF TEST
                  *
 011C 0201 0282          LI   R1,ENDMES-BASE
 0120 06AF 026A          BL   @PRINT-BASE(R15) ; PRINT END MESSAGE
 0124 06A0 00B0          BL   @TYPEN            ; NEW LINE
 0128 108C               JMP  MEM002            ; REPEAT ALL TESTS
                  *
                  *  ROUTINE: GALPAT
                  *  GALLOPS DATA STORED IN R8 THRU MEMORY.  FOR EACH LOCATIO
                  *  ALL LOCATIONS W/ ADDRESSES DIFFERRING BY POWER OF 2 +/-
                  *  ARE CHECKED.
                  *
 012A C28B        GALPAT MOV  R11,R10           ; SAVE RETURN ADDRESS
 012C C0C8               MOV  R8,R3             ; SET UP PATTERN COMPLEMENT
 012E 0543               INV  R3
 0130 C04E               MOV  R14,R1            ; SET UP PTRN COMPL THRUOUT ME
 0132 C443        G10    MOV  R3,*R1            ; SET WORD
 0134 8443               C    R3,*R1            ; DID IT STICK?
 0136 130C               JEQ  G15
 0138 C141               MOV  R1,R5             ; NO - REPORT ERROR
 013A 06AF 0230          BL   @TPWDSP-BASE(R15)
 013E C143               MOV  R3,R5
 0140 06AF 0230          BL   @TPWDSP-BASE(R15)
```

```
0144 C143                 MOV  R3,R5
0146 2951                 XOR  *R1,R5
0148 06AF 0230            BL   @TPWDSP-BASE(R15)
014C 06A0 0126            BL   @TYPEN
0150 05C1         G15     INCT R1                  ; GO TO NEXT ADDR
0152 81C1                 C    R1,R7               ; DONE?
0154 1AEE                 JL   G10                 ; NO - LOOP
                  *
                  *  NOW STEP R1 THRU MEMORY; LOAD WITH PATTERN; AND CHECK
                  *  OTHER LOCATIONS
                  *
0156 C04E                 MOV  R14,R1              ; BEGIN WITH START ADDRESS
0158 0551         G20     INV  *R1                 ; COMPLEMENT THIS WORD
015A 8211                 C    *R1,R8              ; OK?
015C 130C                 JEQ  G21
015E C141                 MOV  R1,R5               ; NO - REPORT ERROR
0160 06AF 0230            BL   @TPWDSP-BASE(R15)
0164 C148                 MOV  R8,R5
0166 06AF 0230            BL                  )
016A C148                 MOV  R8,R5
016C 2951                 XOR  *R1,R5
016E 06AF 0230            BL   @TPWDSP-BASE(R15)
0172 06A0 014E            BL   @TYPEN
                  *
                  *  NOW CHECK ALL LOCATIONS DIFFERRING BY +/- A
                  *  POWER OF 2 +/- 0,1
                  *
0176 0202 4000    G21     LI   R2,>4000            ; START W/ 8K WDS AS PWR OF 2
017A C242         G22     MOV  R2,R9               ; TEST +/- POWER OF 2 WORDS
017C 06AF 0198            BL   @NEIGHB-BASE(R15)
0180 C242                 MOV  R2,R9               ; TEST +/- POWER OF 2 - 1 WORI
0182 0649                 DECT R9
0184 06AF 0198            BL   @NEIGHB-BASE(R15)
0188 C242                 MOV  R2,R9               ; TEST +/- POWER OF 2 + 1 WORI
018A 05C9                 INCT R9
018C 06AF 0198            BL   @NEIGHB-BASE(R15)
0190 0912                 SRL  R2,1                ; GET NEXT POWER OF 2
0192 0282 0002            CI   R2,2                ; DONE IF WAS 4 (2 WORDS)
0196 16F1                 JNE  G22                 ; LOOP TILL DONE
0198 0571                 INV  *R1+                ; RESTORE ORIG LOC & INCT POIN
019A 81C1                 C    R1,R7               ; DONE?
019C 1ADD                 JL   G20                 ; NO - KEEP GOING
019E 045A                 B    *R10                ; RETURN VIA R10
                  *
                  *  ROUTINE: NEIGHBOR
                  *  CALLED FROM GALPAT.
                  *  COMPARES A WORD (ADDR COMPUTED BY ADDING +/- ARG TO POIN
                  *  TO PATTERN COMPLEMENT.  THEN CHECKS THE WORD AT THE POIN
                  *
01A0 C34B         NEIGHB MOV  R11,R13              ; SAVE RETURN ADDRESS
01A2 C141         N2      MOV  R1,R5               ; COMPUTE ADDR OF WORD TO TEST
01A4 A149                 A    R9,R5
01A6 8385                 C    R5,R4              ; TOO LOW?
01A8 1A25                 JL   N10                 ; YES - SKIP TEST
01AA 8185                 C    R5,R6               ; TOO HIGH?
01AC 1B23                 JH   N10                 ; YES - SKIP TEST
01AE 80D5                 C    *R5,R3              ; TEST IT
01B0 130F                 JEQ  N5
```

```
01B2 C805 00D2          MOV  R5,@STASH          ; OUTTA REGS!  STASH R5
01B6 06AF 0230          BL   @TPWDSP-BASE(R15)          ; REPORT ERROR
01BA C143               MOV  R3,R5
01BC 06AF 0230          BL   @TPWDSP-BASE(R15)
01C0 C160 00D2          MOV  @STASH,R5
01C4 C155               MOV  *R5,R5
01C6 2943               XOR  R3,R5
01C8 06AF 0230          BL   @TPWDSP-BASE(R15)
01CC 06A0 0174          BL   @TYPEN
01D0 8211         N5    C    *R1,R8             ; CHECK ORIGINAL WORD
01D2 1310               JEQ  N10
01D4 C805 00D2          MOV  R5,@STASH          ; REPORT ERROR
01D8 C141               MOV  R1,R5
01DA 06AF 0230          BL   @TPWDSP-BASE(R15)
01DE C148               MOV  R8,R5
01E0 06AF 0230          BL   @TPWDSP-BASE(R15)
01E4 C148               MOV  R8,R5
01E6 2951               XOR  *R1,R5
01E8 06AF 0230          BL   @TPWDSP-BASE(R15)
01EC 06A0 01CE          BL   @TYPEN
01F0 C160 00D2          MOV  @STASH,R5          ; GET R5 BACK
01F4 0509         N10   NEG  R9                 ; REPEAT W/ NEGATIVE OF ARG?
01F6 11D5               JLT  N2                 ; YES - DO IT
01F8 045D               B    *R13               ; NO - RETURN
                  *
                  *  ROUTINE: TRAV
                  *  PERFORM ROTATING BIT TEST.  CHECK FOR BAD DATA LINES
                  *  IN MEMORY.
                  *
01FA C0FB         TRAV  MOV  *R11+,R3           ; FETCH ARGUMENT (PATTERN)
01FC C28B               MOV  R11,R10            ; NOW SAVE RETURN ADDRESS
01FE C04E               MOV  R14,R1             ; SET ADDRESS POINTER
0200 C083               MOV  R3,R2              ; MAKE COPY OF PATTERN
0202 C442         TR10  MOV  R2,*R1             ; STORE PATTERN
0204 0208 0020          LI   R8,32              ; PRESET LOOP COUNTER
0208 8451         TR20  C    *R1,*R1            ; KEEP READING DATA IN R1
020A 0608               DEC  R8
020C 16FD               JNE  TR20               ; 32 TIMES
020E 8091               C    *R1,R2             ; SHOULD STILL MATCH
0210 130C               JEQ  TR25
0212 C141               MOV  R1,R5              ; IF NOT, REPORT ERROR
0214 06AF 0230          BL   @TPWDSP-BASE(R15)
0218 C142               MOV  R2,R5
021A 06AF 0230          BL   @TPWDSP-BASE(R15)
021E C142               MOV  R2,R5
0220 2951               XOR  *R1,R5
0222 06AF 0230          BL   @TPWDSP-BASE(R15)
0226 06A0 01EE          BL   @TYPEN
022A 0B12         TR25  SRC  R2,1               ; SHIFT PATTERN
022C 80C2               C    R2,R3              ; BACK TO ORIGINAL?
022E 16E9               JNE  TR10               ; NO - LOOP
0230 05C1               INCT R1                 ; GO TO NEXT ADDRESS
0232 81C1               C    R1,R7              ; PAST END ADDRESS?
0234 16E6               JNE  TR10               ; NO - DO IT AGAIN
0236 045A               B    *R10               ; YES - RETURN
                  *
                  *  ROUTINE: TYPE_WORD_SPACE
                  *  PRINT HEX WORD FOLLOWED BY A SPACE
```

```
                       *
0238 C80B 00D0    TPWDSP MOV  R11,@RETURN       ; SAVE RETURN ADDRESS
023C 06A0 00AC           BL   @TYPEWD           ; PRINT HEX WORD
0240 2CAF 0280           OUT  @SPACE-BASE(R15)  ; TYPE A SPACE
0244 C2E0 00D0           MOV  @RETURN,R11
0248 045B                B    *R11              ; RETURN
                  *
                  *  ROUTINE: TYPE_ERROR
                  *  PRINTS ADDRESS OF BAD LOCATION, GOOD WORD, AND
                  *  (GOOD WORD) XOR (BAD DATA)
                  *
024A C34B         TYPERR MOV  R11,R13           ; SAVE RETURN ADDRESS
024C C143                MOV  R3,R5
024E 06A0 023E           BL   @TYPEWD           ; PRINT ADDRESS
0252 C148                MOV  R8,R5
0254 06A0 0250           BL   @TYPEWD           ; GOOD WORD
0258 C148                MOV  R8,R5
025A 2953                XOR  *R3,R5
025C 06A0 0256           BL   @TYPEWD           ; (GOOD WORD) XOR (BAD DATA)
0260 06A0 0228           BL   @TYPEN            ; NEW LINE
0264 045D                B    *R13
                  *
                  *  RANDOM NUMBER GENERATOR
                  *
0266 C009         RANDOM MOV  R9,R0             ; MOVE VALUES
0268 C24A                MOV  R10,R9            ;   TO PREPARE FOR MULTIPLY
                  *
026A 3A47                MPY  R7,R9             ; MULTIPLY BY GENERATOR
026C 3807                MPY  R7,R0             ; MULTIPLY BY GENERATOR
                  *
026E A241                A    R1,R9             ; ADD PARTIAL PRODUCTS
0270 045B         EXIT   B    *R11              ; GO BACK HOME
                  *
                  *  PRINT ROUTINE
                  *
0272 A04F         PRINT  A    R15,R1            ; ADD THE BASE ADDRESS
0274 D031         PRT010 MOVB *R1+,R0           ; PRINT *R1+ TILL 0
0276 13FC                JEQ  EXIT
0278 2C80                OUT  R0
027A 10FC                JMP  PRT010
                  *
                  *  MESSAGES
                  *
027C 5354 4152    STARTM TEXT 'START: '
0280 543A 20
0283 00                  BYTE 0
0284 454E 443A    ENDM   TEXT 'END: '
0288 20
0288              SPACE  EQU  $-1
0289 00                  BYTE 0
028A 5445 5354    ENDMES TEXT 'TESTS COMPLETED'
028E 5320 434F
0292 4D50 4C45
0296 5445 44
0299 00                  BYTE 0
029A 0001         ONE    DATA 1
029C                     END
```

```
 0094 BAD     0008 BASE    0284 ENDM    028A ENDMES  0270 EXIT
 0132 G10     0150 G15     0158 G20     0176 G21     017A G22
 012A GALPAT  00D6 M10     00E2 M15     00E8 M20     00F2 M25
 00FE M28     0102 M30     010A M35     0116 M38     0042 MEM002
 0046 MEM004  0054 MEM010  0070 MEM020  0078 MEM025  007E MEM030
 0088 MEM040 *0000 MEMTST  00B4 MTEST   01F4 N10     01A2 N2
 01D0 N5      01A0 NEIGHB  029A ONE     0272 PRINT   0274 PRT010
 0000 R0      0001 R1      000A R10     000B R11     000C R12
 000D R13     000E R14     000F R15     0002 R2      0003 R3
 0004 R4      0005 R5      0006 R6      0007 R7      0008 R8
 0009 R9      0266 RANDOM  003A RDNUMB  00D0 RETURN  0288 SPACE
 027C STARTM  00D2 STASH  *0000 TEST    0238 TPWDSP  0202 TR10
 0208 TR20    022A TR25    01FA TRAV    0262 TYPEN   024A TYPERR
 025E TYPEWD  0006 TYPEX
EDIT/AS/LOAD?
```

3.3.2 T99EP EPROM Memory Board

The T99EP board features up to 32K bytes of 2716 EPROM storage or 16K bytes of 2708 EPROM storage. The starting address of the board is selected in 1K bytes of increments by DIP switches. Memory may be disabled in 2K byte increments (2708) or 4K byte increments (2716) by DIP switches. This means that the T99EP board need not occupy more addressing space than is required by the PROMs actually installed on the board.

3.3.2.1 Theory of Operation

Interface with the rest of the system consists of the address bus, data bus, and control signals (/MEMEM, DBIN). The address bus is buffered by 74LS367 (U4, U5, U6). The outgoing data bus is also buffered by 74367's (U15, U16, U17).

The starting address for the EPROM board is determined by the adding circuitry of U19 and U2. The incoming five address lines (BA0 to BA4) are added to an offset to produce the actual address lines used for decoding. Figure 1 illustrates how the switches U18 are set to obtain various starting addresses. The DIP switches U3 and the remainder of U18 and the buffers U11 and U13 determine whether the board is to be operated in 2708 or 2716 mode. To operate the board in 2708 mode the two buffers (U11, U13) are removed and S1, S2, S3, S4 of U3 and S7 of U18 are closed, S5, S6, S7, S8 of U3 and S6 of U18 are open. To operate the board in 2716 mode, the two buffers are installed (U11, U13) and the DIP switches S1, S2, S3, S4 of U3 are open, S5, S6, S7, S8 of U3 are closed, S6 of U18 is closed, and S7 of U18 is open.

WARNING: The DIP switches U3 and U18 and the buffer (U11 and U13) must be properly installed or damage to the board may result.

Address decoding is accomplished by the 74138 1 of 8 decoder (U7). This decoder (U7) will either produce 2K byte chip selects (2708) or 4K byte select (2716). The chip select output of the decoder are connected to the EPROM positions by DIP switches (U8). This allows the user to select the number of EPROM locations that will be enabled on the board. To enable the EPROM U20, U28 DIP switch S1 of U8 must be closed. The additional logic U14 (a 7404 inverter) and U9 (74LS30 NAND gate) control the data output box. Whenever a chip select signal is low the output of U9 will be high and hence the output of U14 low. This low will enable the data bus buffers (U15, U16, U17).

The user should be cautious not to enable more EPROMs than the addressing space currently allows. For example, if the starting address of the EPROM board were set at, say D000 and all of the EPROMs were enabled (S1 to S8 of U8 all closed) then an address conflict would exist with the IIA and monitor PROMs since they occupy part of that addressing space.

3.3.2.2 Installation and Operation

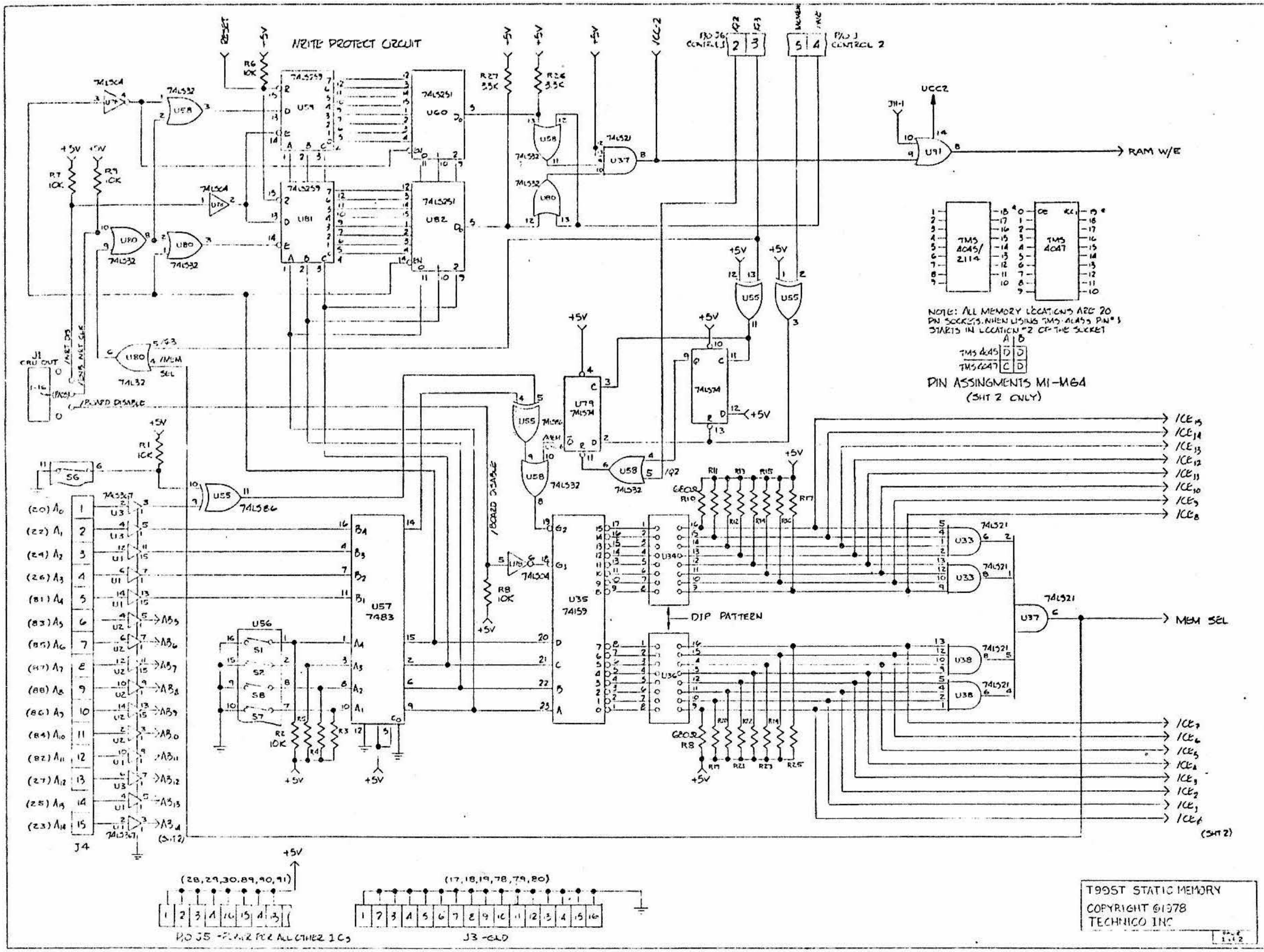
To install the T99EP board in the system, the address lines (J4), the data lines (J9), the control lines (J6) power and ground must be connected

Figure 1

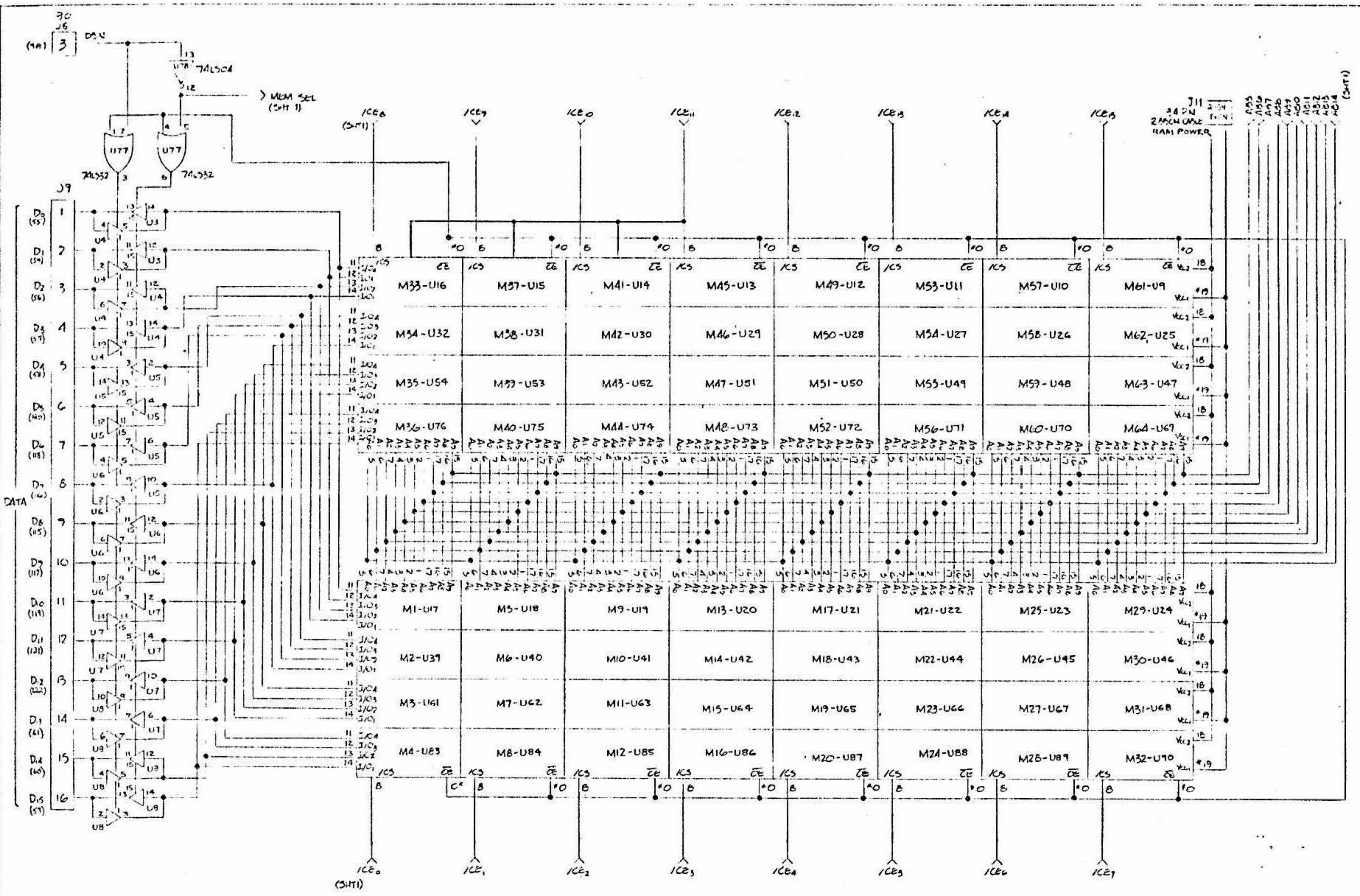
START ADDRESS (S1 = 0)	(S1 = 1)	S2, S3, S4, S5 (1 = open, 0 = closed)
0000	8000	0000
F800	7800	0001
F000	7000	0010
E800	6800	0011
E000	6000	0100
D800	5800	0101
D000	5000	0110
C800	4800	0111
C000	4000	1000
B800	3800	1001
B000	3000	1010
A800	2800	1011
A000	2000	1100
9800	1800	1101
9000	1000	1110
8800	0800	1111

to the board. The board is then configured for either 2708 or 2716 operation as described earlier. The starting address is selected via U18 according to Figure 1. If other memory boards are installed in the system it is very important that the starting address and memory enable switches are set to preclude an address conflict. If an address conflict exists, then the system will not operate properly.

To determine proper operation of the board, simply install a pair of known EPROMs in the first location (U20, U28) and read the data from these EPROMs at the starting address previously selected. If any problems occur, then first check that the board has been properly configured and the starting address properly selected. In addition, be sure that the EPROM enable switch S1 of U8 is closed or the EPROM will not be enabled.



T99ST STATIC MEMORY
COPYRIGHT ©1978
TECHNICO INC



M1-M64 ARE 2114 (4045) OR 4047
REFER TO INSET ON SHEET 1
FOR DEFINITION OF PINS
FOR THE RAMS

T99ST STATIC MEMORY
COPYRIGHT © 1978
TECHNICO INC.

SHT
2 of 2

Oct. 31, 1978