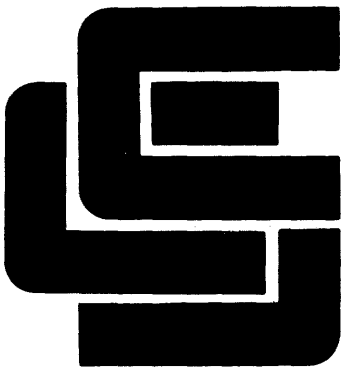
Owner's Manual

Model 2032

32K Static RAM Module



California Computer Systems

CCS MODEL 2032

32K STATIC RAM MODULE

OWNER'S MANUAL

CALIFORNIA COMPUTER SYSTEMS
250 CARIBBEAN DRIVE
SUNNYVALE CA 94086

MANUAL NO. 89000-02032

TABLE OF CONTENTS

FEATURES

Uses Popular 2114 Static RAMs

Available with 200, 300, or 450 nsec RAMs

Berg Jumpers Used for Selectable Features

8K Memory Blocks Individually Addressable to Any 8K Boundary

Bank Selection by Bank Port and Bank Byte

8K Blocks Individually Bank-Enabled

LEDs Indicate Board Active and Bank Active States

Wait State Jumper

Phantom Line Capability

Optional Board-Enabling on Reset

Operates on +8 Volts

Fully Buffered

Meets IEEE Proposed S-100 Signal Standards

Diagnostic Software Included

FR-4 Epoxy PC Board Solder-Masked on Both Sides

Silk Screen of Part Numbers and Reference Designations

CHAPTER 1

SETTING THE 2032 JUMPERS

The CCS 2032 is a 32K byte static RAM board designed for use on S-100 busses. Sixty-four popular 2114 static RAM chips make up the four 8K memory groups A through D. Each memory group is individually addressed and bank-enabled, and up to three memory groups can be buried to reconfigure the board to 8K, 16K, or 24K. The bank select feature, using a bank port and bank byte, is compatible with Alpha-Micro and Cromemco as well as with other systems. Board Active and Bank Active states are indicated by LEDs.

To provide optimum compatibility with a variety of systems, CCS has equipped the 2032 with selectable addressing and several optional features. Selections are hardwired with easy-to-use, reliable Berg jumpers. The addresses for each of the 8K memory groups, the bank port address and bank byte, and the bank-dependence or bank-independence of each memory group are jumper-selected by the user to best suit his system. Phantom, Wait, and Reset features can be jumper-enabled as desired. Each jumper-selectable feature is discussed individually below. Further explanation can be found in Chapter 2, "Theory of Operation." Illustrations showing jumper settings and relative locations are provided in Section 1.8.

1.1 SETTING THE MEMORY GROUP ADDRESSES

In order to provide maximum flexibility in the location of the 2032's memory groups within a bank, CCS has made the addresses of the four memory groups jumper-selectable. The jumper-set address of a memory group is compared with the high-order address lines A13-A15, and if the address

matches, the group is selected. The Group Address (GRP ADDR) jumpers are in the upper left corner of the board (with the connector pins at the bottom). Set the jumpers of each group to the three high-order binary digits that specify the multiple of 8K at which you wish to locate the group. For example, the addresses of the block between 16K and 24K are 4000h-5FFFh, so you would locate a group in that block by setting its jumpers to 010. Since a memory group's base address must be a multiple of 8K, an easy way to calculate the jumper settings is to divide the base address by 8K. You can then set the jumpers to the binary equivalent of the result.

The memory groups are fully prioritized, with A having the highest priority and D the lowest. This allows you to give two (or more) memory groups the same address. Only the higher-priority group will be selected by that address; the RAMs of the other group(s) will be buried, inaccessible and occupying no memory space until the address jumpers are reset. This allows you to configure the 2032 to 8, 16, or 24K without removing RAMs.

1.2 SETTING THE BANK BYTE

The Bank Byte jumpers allow you to hardware-map the 2032 memory board to whichever of the eight memory bank levels 0-7 you choose. They are located at the top of the board. To select a bank level, jumper-set a 1 in the bit that corresponds to the desired bank level and jumper-set all other bits to 0. For example, to select bank 3 you would set bit D3 to 1 and D0-D2 and D4-D7 to 0.

You may enable the 2032 with more than one bank. Set to 1 the Bank Byte jumper corresponding to any bank with which you want the board to be enabled.

1.3 SETTING THE BANK PORT ADDRESS

In order to assign the board to a bank, you must output the bank byte to the bank port. Most presently-marketed S-100 products using the bank port/bank byte scheme address the bank port at 40h. We recommend that you use this bank port address unless you have a strong reason for doing otherwise. The Bank Byte jumpers are at the bottom of the board, just above the connector pins. Remember that A7 is

the high-order bit; thus you will set the binary bank port address from right to left on the board. 40h is selected by jumper-setting A6 to 1 and A0-A5 and A7 to 0.

1.4 SETTING MEMORY GROUP BANK-INDEPENDENCE

Each memory group can be made independent of bank selection, causing it to be enabled whenever it is addressed regardless of which bank is active. This makes it possible, in time-sharing situations, for some groups to be commonly accessible while the remaining bank-dependent groups are reserved for individual users. The bank-independence jumpers are located at the bottoms of the GRP ADDR columns. Setting a jumper to BE (Bank Enable) makes the corresponding memory group bank-dependent. To enable a memory group independent of bank selection, set its bank-independence jumper to ME (Memory Enable).

1.5 SETTING THE RESET JUMPER

The Reset jumper, at the top center of the board, controls the activating of the bank-dependent memory groups during system resets. If the Reset jumper is set to B, all 32K of memory will be enabled each time the power is turned on or the system is reset. If the Reset jumper is set to A, the bank-dependent memory groups will be enabled only when the board's bank has been selected.

Due to lack of room on the board, the Reset jumper labels may be hard to find. The B position is to the right; the A position is to the left.

1.6 SETTING THE PHANTOM JUMPER

The Phantom jumper is in the lower right corner of the board. Setting the jumper to B allows a device that generates a PHANTOM signal to overlay portions of the 2032 memory. For example, CCS peripheral control boards generate Phantom signals when certain ROM locations are addressed; these locations contain code to drive the peripherals. If an identically-addressed location exists on the 2032 board, the Phantom signal will block the output from the board of

the contents of that location. This allows you to access the rest of the memory locations within the 8K block that contains the overlayed portion. Without Phantom capability the 2032 would not be able to locate a memory group in that block because the 2032 and the peripheral control board would both put data on the bus when a shared location was addressed.

Setting the Phantom jumper to A disables the -PHANTOM signal.

1.7 SETTING THE WAIT JUMPER

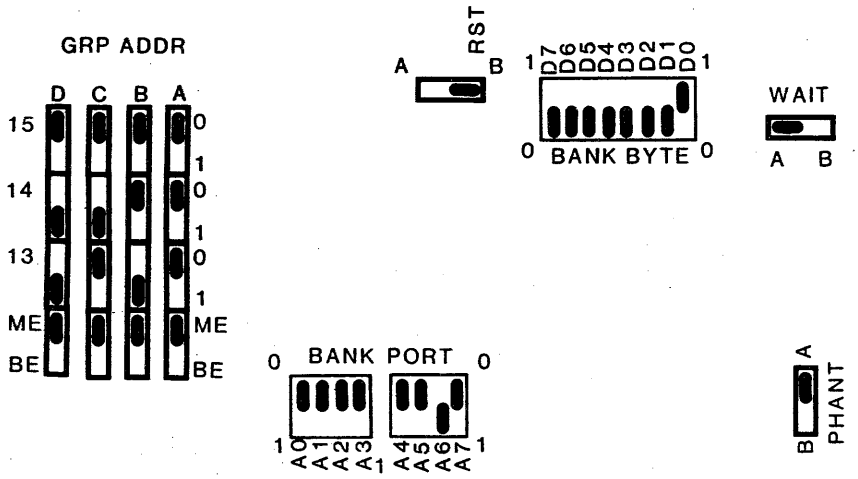
The Wait jumper allows you to slow down your processor every time the board is addressed. This will be necessary if your processor allows a shorter memory access time than your RAMs require. The jumper is in the upper right corner of the board. Off is the A position; on is B.

If you have a 2032 with 200 nsec or 300 nsec RAMs, you should not need to enable the Wait feature for use with presently-available microprocessors. If you have the 450 nsec RAMs and a processor that operates at 4mHz you will, in theory at least, need to enable Wait. You should experiment, however; in many cases the 450 nsec RAMs will work successfully with a 4mHz processor without a Wait state.

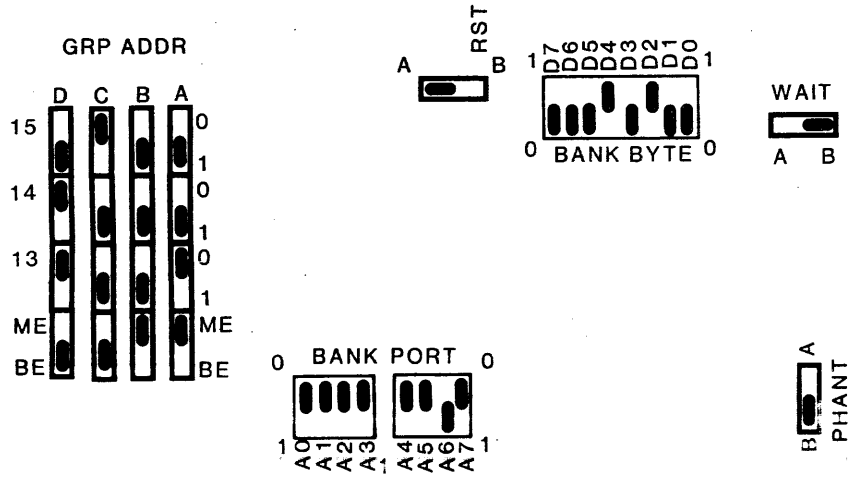
Some Z-80 CPU boards, including the CCS 2810, provide a jumper-selectable Wait feature. Enabling this feature may be preferable to enabling the 2032 Wait feature. The 2032 Wait causes a Wait state to occur in every memory cycle in which the board is addressed; the CCS CPU Wait feature causes a Wait state to occur during the M1 cycle only. Because memory access time in the M1 cycle is half a clock cycle shorter than in the other machine cycles, a Wait state in this cycle effectively increases the time allowed for memory response without unnecessarily slowing the processor in other memory cycles. However, if your system includes memory boards operating at different speeds, you probably will want to enable the Wait features as necessary on the slower memories rather than enable the processor Wait. This will allow you to operate at maximum speed with the faster memories. To find out what is best for your system, check your CPU manual and, if you're not sure, experiment.

1.8 EXAMPLES OF JUMPER SETTINGS

The first diagram shows jumper settings for a basic CCS system consisting of a 2810 Z-80 CPU, a 2422 disk controller, and the 2032. The bank port address must be 40h. The board is enabled with bank 0 and on start-up. Memory is located between 0 and 32K. Phantom and Wait are disabled.



In the second diagram, memory groups A and B are bank-independent and located in the last 16K of memory. Groups C and D reside in banks 2 and 4 between 24K and 40K. The bank port address is 40h. Only groups A and B are enabled on start-up. Phantom and Wait are enabled.



CHAPTER 2

THEORY OF OPERATION

This chapter is provided for those users who want a more thorough understanding of the 2032 operation than they need just to make the board function in their systems. Used in conjunction with the Logic Diagram and the Control ROM Truth Table in Chapter 4, it should give you a sound understanding of the design and features of the board. Additional information, if desired, can be obtained from data sheets for the individual chips.

2.1 MEMORY

The 2032 uses sixty-four 2114-type RAMs. The memory chips are arranged in two-chip columns in order to provide an eight-bit byte, and the thirty-two columns are divided into four 8K memory groups A through D. Because the 2114 provides 4096 bits of storage organized 1024x4, each RAM requires ten address inputs and four bi-directional data lines. A Chip Select input (-CS) provides for the selection of individual chips in the memory array. To prevent erroneous data from getting into the chip a R/-W input inhibits the data input buffer when high. Thus data can be written to a memory chip only when both -CS and R/-W are low. The 2032 controls -CS through the Column Select Decoders; R/-W is controlled by the Control ROM through the Read/Write Decoder.

2.2 MEMORY ADDRESSING

Addressing a specific memory location on the 2032 involves addressing a location on each chip while enabling only one two-chip column. Address lines A0-A9 address one location on each chip through a common address bus. Column selection is handled by four 3-to-8 decoders. Each decoder selects one of eight columns depending on the conditions of inputs A, B, and C, which are controlled by address lines A10-12. Inputs G1, G2A, and G2B determine whether an individual decoder will be enabled, G2A and B low and G1 high enabling a decoder.

The three highest-order address lines determine the 8K block in which a memory group resides. Jumpers are used to select each memory group's base address (see Section 1.1). The jumper settings are compared with the top three bits of the incoming address, and if a group's settings correspond to the address bits that group's output line is pulled low. Each group's output line is tied directly to input G2A of the decoder for that group. Also, low outputs from Group A and Group C disable through G1 the decoders for Groups B and D respectively. In addition, groups C and D are disabled through G1 if the output of the ANDing of Groups A and B is low--i.e., if either Group A or Group B has been addressed. This provides full prioritizing of the memory groups, with A the highest priority and D the lowest. Whenever two or more memory groups are given the same base address, only the highest-priority group will be enabled by that address. The other groups will effectively be buried; they will be unaddressable and will occupy no memory space.

The final input for each decoder, G2B, is determined by the Control ROM through -CSE (Column Select Enable). See Section 4.4 for the specific conditions under which -CSE will be low.

2.3 BANK SELECTION

The CCS 2032 is bank-selectable by bank port address and bank byte. Thus it is fully compatible with Cromemco, AM100, and other port-bank-select systems. IT IS NOT COMPATIBLE WITH ADDRESS-SELECT SYSTEMS SUCH AS IMSAI.

You assign the 2032 to a bank by jumper-setting the bank port address and the bank byte. The 2032 compares A0-A7 with the jumper-set bank port address using an open

collector set of exclusive-OR gates. A pull-up resistor holds the output high unless a wrong address pulls the output low. The BANK PORT ADDRESS line inputs to the Control ROM. If the conditions of the BANK PORT ADDRESS line and the other Control ROM inputs are right (see Section 4.4), BANK CLK will pulse low, clocking the Bank Enable flip-flop when it rises to high again. In the meantime the bank byte becomes present on DIO-7 and is inverted. Setting a Bank Byte Select jumper to 1 connects the corresponding inverted DATA IN line to the BANK DATA line. Thus a low signal on an inverted DATA IN line, indicating a 1 in the bank byte, will pull BANK DATA low if the corresponding Bank Byte Select jumper is set to 1.

When the flip-flop is clocked, the condition of BANK DATA, the flip-flop's D input, determines the outputs Q and -Q. Q takes the value of D and -Q is D's complement. When BANK DATA is low, indicating that the bank byte and the Bank Byte Select jumpers specify at least one bank in common, the -Q output is high. The -Q output is tied to BANK ENABLE. When BANK ENABLE is high, selection of bank-dependent memory groups is enabled. At the same time, the low output at Q lights the Bank Select LED and pulls -BANK ACTIVE low. When -PORT READ and -BANK ACTIVE are both low, -ACK will be low, acknowledging to the processor that a bank has been enabled.

When BANK DATA is high, the low on BANK ENABLE forces all bank-dependent memory groups' slect lines (-GROUP A-D) high. The low on -Q also turns of the Bank Select LED, while the high on -BANK ACTIVE (from Q) ensures that -ACK will be high.

Because flip-flop outputs do not change until the flip-flop is re-clocked, BANK ENABLE, -BANK ACTIVE, and the Bank Select LED will maintain the same states until the bank port is addressed again, when another bank byte will determine whether a high or a low gets clocked into the Bank Enable flip-flop.

2.4 BANK-INDEPENDENCE

The 2032 allows you to make any memory group independent of bank disabling by setting a jumper so that the BANK ENABLE line is not connected to the memory-address-comparison circuitry of the memory group you want to make independent. This prevents that memory group's output from being pulled low when the BANK ENABLE line is low. The memory group will therefore be enabled whenever it is addressed, independent of which bank has been selected.

2.5 DATA BUFFERS

The DATA IN and DATA OUT lines from the data bus are tied together to form the bi-directional data lines for the RAM chips. DI0-7 and DO0-7 are buffered by 3-state bus drivers. If the drivers are in the high-impedance state, the lines they drive are disabled. The -RD ENABLE and -WR ENABLE lines, which determine whether the DI or DO buffers will be in the high-impedance state, are controlled through the Read/Write Decoder by the Control ROM. See Section 4.4 for the specific conditions under which -RD ENABLE and -WR ENABLE will be low.

2.6 WAIT STATES

A wait state is necessary when a peripheral device takes more time to complete a task than the processor normally allows. Because the 2032 is available with 200, 300, or 450 nsec Rams, and because processor speeds vary, the Wait feature on the 2032 has been made jumper-selectable. If the Wait jumper is set to B, pSYNC is inverted and ORed with -CSE, with the output being the pRDY line. When pRDY goes low, the processor adds an extra clock cycle to each memory read or memory write machine cycle during which the board is selected, thereby increasing the time that signals remain on the address and data busses. If the jumper is set to A, a high signal is ORed with -CSE, the 2032 does not pull pRDY low, and a Wait state does not occur unless it originates elsewhere.

2.7 RESET

The Reset jumper allows you to choose whether or not the 2032 will be enabled when the system is powered up or reset by determining which input of the Bank Enable flip-flop will be controlled by pRESET. Pull-up resistors normally hold both the Preset and Clear inputs high, which they must be for the flip-flop to set and reset normally. The -pRESET line can be jumper-connected so that either the Preset input or the Clear input is pulled low whenever the power is turned on or the system is reset. If the Reset jumper is set to position A, -pRESET active pulls the Preset input low, the flip-flop is set, BANK ENABLE is low, and the bank-dependent memory groups are disabled. If the jumper is set to position B, -pRESET active pulls the Clear input low, the flip-flop is reset, BANK ENABLE is high, and the bank-dependent memory groups are enabled.

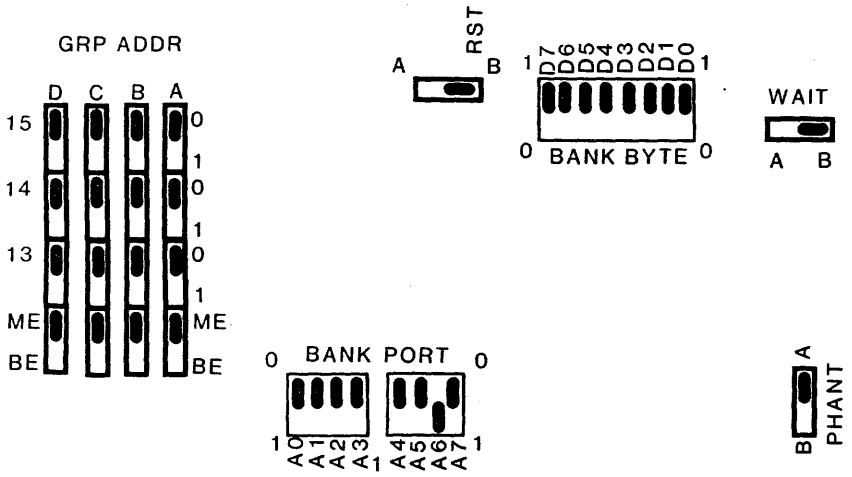
CHAPTER 3

TESTING AND TROUBLESHOOTING THE 2032

3.1 FRONT PANEL QUICK CHECKOUT

(If your computer does not have a front panel, skip this section.)

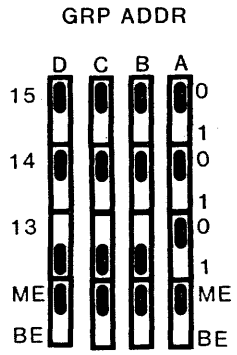
Before powering on the computer, set the 2032 jumpers as follows:



The priority feature will cause Group A to be selected. Set the Front Panel Adress Switches A0-A15 to the off position (0000H). Examine that address. Set the Data Switches D1-D7 to the OFF position and D0 to the ON position (01H). Deposit (write) into memory and compare the Data readout with the switch settings. Now switch D0 to OFF and D1 to ON, deposit into memory again, and compare the result with the switch settings. Continue the pattern of one Data

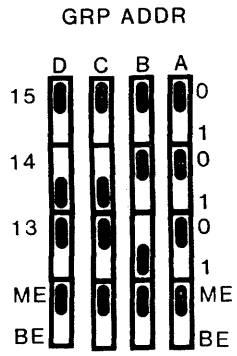
Switch ON and the rest OFF until all data bits have been checked. If any data does not match the switch settings, isolate the malfunction with a logic probe or voltmeter before continuing.

After Group A has been checked, power down the computer and set Groups B-D to 001 as shown:



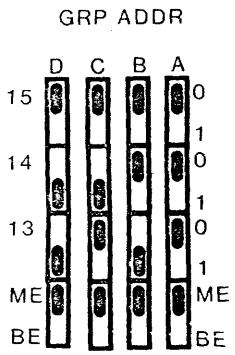
Group B will be selected. Examine 2000H (A13 ON, the rest OFf), and deposit the same data bytes as for Group A. Isolate and correct any malfunctions as they become apparent.

To check Group C, power down the computer and set Groups C and D to 010:



Examine 4000H (A14 ON, the rest OFF), and test as with Groups A and B.

Finally, to test Group D, power down and set Group D to 011:



Examine 6000H (A14 and A13 ON, the rest OFF), and test as before. When all malfunctions have been corrected, proceed to the next test.

3.2 DIAGNOSTIC TEST OVERVIEW:

These memory diagnostics run on 8080 or Z-80 systems and provide a practical test of the 2032 memory board. Two diagnostics are provided: a walking bit test and a burn-in test. The routines have been written so that they do not require RAM other than the system stack and the RAM under test. The routines may be executed from either RAM or ROM.

Diagnostics in general can be divided into three classes: fault detection, fault isolation, and fault correction. These routines perform fault detection and provide sufficient data for fault isolation. After a fault is isolated, correction is a hardware matter.

Errors are displayed on the console device when they are detected. Two formats are used. The first, used by the burn-in test and the first stage of the walking bit test, shows errors as follows:

```
xx yyyy zz
```

Each character is a hexadecimal digit; xx is the bad data, yyyy is the address where the bad data occurred, and zz is what the data should have been.

The second stage of the walking bit test logs errors as follows:

```
wwww xx yyyy zz
```

Again, each character is a hexadecimal digit; wwww is the address where the error was found, xx is the bad data, yyyy is the address where data was last written, and zz is the last written data.

These error displays provide enough information for the problem to be isolated.

3.3 PREPARING DRIVER ROUTINES

Except for the system-unique input/output drivers, the memory test routines are capable of standing alone. The drivers must be provided by the user. Three routines are needed:

CONIN: Console input. Reads one ASCII character from the console keyboard and sets the parity bit (bit 7) equal to 0. The character is returned in the accumulator (A register).

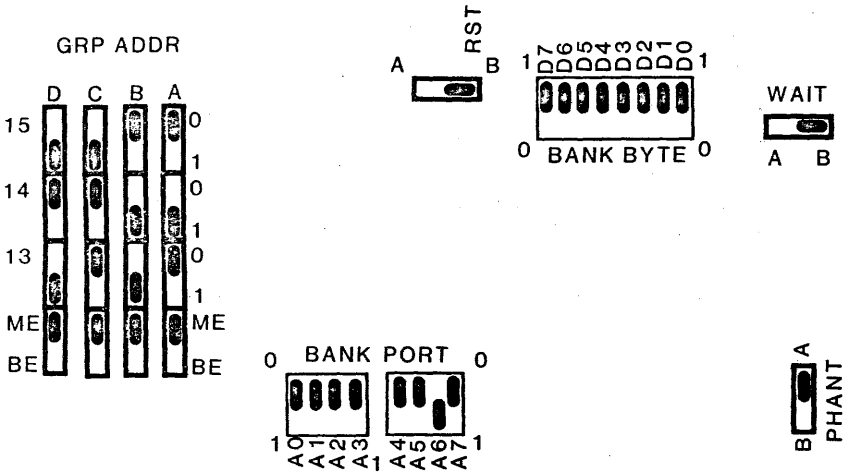
CONOUT: Console output. Writes one ASCII character to the console display device. The character to be output is passed to CONOUT in the C register. If the console output device is sensitive to bit 7, then the user must set/reset bit 7 to what is needed in the CONOUT routine.

CONST: Console status. This routine reads the console input status. If data is not available, then the accumulator is set to 0 and the status flags must match. If data is pending, then a -1 (0FFH) should be returned in the accumulator (A register). The status flags must show at least a non-zero condition on the return.

After these routines have been prepared they must be loaded into memory. To allow the diagnostics to find them, three jump instructions are located at the front of the diagnostic: 0103H for CONIN, 0106H for CONOUT, and 0109H for CONST. The user should put the addresses of his I/O routines into these locations. See lines 51, 52, and 53 in the assembly listings.

3.4 SETTING UP FOR THE TEST:

When you are ready to begin the test, set the 2032 jumpers as illustrated:



At this point you are ready to install the 2032 in your computer. Make sure that no other memory will respond to addresses in the range 4000H-0BFFFH.

3.5 LOADING THE DIAGNOSTIC:

No special precautions are necessary. Use your standard method to load the routines. Load the diagnostic into your system at location 0100H. The diagnostic is small enough to fit into the first 1K of memory. It was assembled assuming a 16K block of memory would be available starting at 0000H; if less memory is available, the only change necessary is to alter the stack location. The stack is currently initialized to 3F76H; a good alternate location would be 0100H.

3.6 RUNNING THE DIAGNOSTIC

Transfer control of the computer to location 0100H. The computer will type out:

DIAGNOSTIC:

You can now select which diagnostic you want. Current options are "C" for continuous burn-in or "W" for walking bit test. Any other selection will cause ???? to be displayed, after which "DIAGNOSTIC:" will again be printed. For the initial test, type in W. The computer will respond:

```
DIAGNOSTIC: WALKING BIT TEST
BLOCK SIZE:
```

Select a small block size initially. This way the read/write circuitry can be checked out without a flood of error printouts. A block size of 2 is suggested. To terminate entry type in a space, a comma, or a carriage return. If you type in the wrong number, continue typing in until the last four digits are correct.

The computer will now ask for

```
BASE ADDRESS:
```

Type in the desired base address. (Note: The base address must be a multiple of 1024 (0400H). For the board setup suggested, a base address of 4000H is indicated.) At this time the diagnostic will do its test. On completion it will type out

```
TEST DONE
DIAGNOSTIC:
```

It is now ready for the next test. If errors were logged, see the troubleshooting section and correct the malfunction. Rerun the diagnostic until an error-free run is achieved.

Rerun the walking bit test with a block size of 1K (400H) and a base address of 4000H. Repeat the test, increasing the base address in 1K (4000H) increments, until base address BC00H has been tested. This tests all memory chips. If errors are logged, replace the appropriate chip(s). Table 3.1 narrows any error to two chips. If the bad data is in the upper half of the byte, replace the odd-numbered chip. If the bad data is in the lower half of the byte, replace the even-numbered chip. For example, the following error printout indicates chip 71 bad:

```
5C02    84      5C02 04
```

After a good run for all thirty-two 1K increments, run the walking bit test with a block size of 32K (8000H).

BASE ADDRESS	CHIPS TESTED	MEMORY GROUP
4000H	U67, U68	A
4400H	U65, U66	A
4800H	U63, U64	A
4C00H	U61, U62	A
5000H	U77, U78	A
5400H	U75, U76	A
5800H	U73, U74	A
5C00H	U71, U72	A
6000H	U49, U50	B
6400H	U47, U48	B
6800H	U45, U46	B
6C00H	U43, U44	B
7000H	U57, U58	B
7400H	U55, U56	B
7800H	U53, U54	B
7C00H	U51, U52	B
8000H	U32, U33	C
8400H	U30, U31	C
8800H	U28, U29	C
8C00H	U26, U27	C
9000H	U40, U41	C
9400H	U38, U39	C
9800H	U36, U37	C
9C00H	U34, U35	C
A000H	U16, U17	D
A400H	U14, U15	D
A800H	U12, U13	D
AC00H	U10, U11	D
B000H	U24, U25	D
B400H	U22, U23	D
B800H	U20, U21	D
BC00H	U18, U19	D

TABLE 3.1

At this point, invert the memory group address jumpers and run a 32K block starting at 000H. This tests the group-select circuitry completely. The primary chips tested here are U1-U3.

When all walking bit tests run error-free, type in C for the continuous burn-in test. Specify a block size of 8000H and the appropriate base address (4000H if you follow the above procedure). Let it run for an hour or two to shake out the weak links (infant mortality). To terminate

this test type in Control C. Errors, if any, will be printed out as they occur. The total number of errors will be printed out upon completion of the test.

3.7 ERROR PRINTOUT INTERPRETATION:

Errors may show up in many forms. Table 3.2 on the next page matches typical symptoms with probable causes. The best way to isolate a problem (and correct it at the same time) is to pull out a suspect part and replace it with a part that you know to be good. Then rerun the diagnostic and see if the problem is still present.

If a problem persists after all suspect parts are replaced, set up a controlled test condition and troubleshoot the problem with a logic probe or a voltmeter, using the logic diagram to identify test points.

ERROR CONDITION	PROBABLE CAUSE	SUSPECT PARTS
Bad data=0FFH, all groups	a) bank select b) board select	U5, U6, U85 U3, U5, U85
Random data or all 0 data, all groups	bad write control	U5, U83, U85
0FFH data, one group only	a) group A select b) group B select c) group C select d) group D select	U2, U3, U9 U2, U3, U42 U1, U3, U60 U1, U3, U70
One address line hung (printout: good data, bad address)	address buffers	U81 (A0-6, A15) U82 (A7-14)
One data line hung a) hung 0 (good address, bad data=0)	grounded data line	U83, U84
b) hung 1 (good address, bad data=1)	a) open data line b) data line shorted to +5V	U83, U84 U83, U84, memory chips
Soft errors (random addresses and data, non-repeatable)	a) memory chip access time b) heat-sensitive parts	Try setting Wait jumper to B and rerunning tests. Treat as a hard error and replace suspect parts.
Hard memory errors	bad memory chip	See Table 3.1 to identify chip.

TABLE 3.2

3.8 SAMPLE MEMORY DIAGNOSTIC RUN:

```
DIAGNOSTIC: WALKING BIT TEST      Typed in W
BLOCK SIZE: 30                    Block may be any size
BASE ADDRESS: 300
BAD BASE ADDRESS:                 Base address must be multiple
BASE ADDRESS: 400                 of 1K (400H)
TEST DONE
DIAGNOSTIC: WALKING BIT TEST      New test
BLOCK SIZE: 400
BASE ADDRESS: 400                 Equal block size, base address
TEST DONE
DIAGNOSTIC: WALKING BIT TEST
BLOCK SIZE: 1000                  Larger block size test
BASE ADDRESS: 400
TEST DONE
DIAGNOSTIC: WALKING BIT TEST
BLOCK SIZE: 1800
BASE ADDRESS: 400
TEST DONE
DIAGNOSTIC: ????                  Typed in 1
DIAGNOSTIC: WALKING BIT TEST
BLOCK SIZE: 579                   Odd block size
BASE ADDRESS: 400
TEST DONE
DIAGNOSTIC: CONTINUOUS BURNIN     Typed in C
BLOCK SIZE: 3765                  No parameter restrictions
BASE ADDRESS: 3D3
00 ERRORS                         Up to 0FFH (255D) errors shown
TEST DONE
DIAGNOSTIC: CONTINUOUS BURNIN
BLOCK SIZE: 3ABC
BASE ADDRESS: 3EF
00 ERRORS
TEST DONE
DIAGNOSTIC:
```

```
 1 0000                          TITLE   '2114 MEMORY DIAGNOSTIC VER 1.1'
 2 0000                ;
 3 0000                ;
 4 0000                ; Console input/output support routines
 5 0000                ;
 6 0000                ; These routines are a highly-matured, well-thought-
 7 0000                ; out  set based on Intel's monitor.  They provide a
 8 0000                ; significant capability to converse with  an  8080,
 9 0000                ; 8085,  or  Z-80  based microprocessor system.  The
10 0000                ; only registers altered are the accumulator and the
11 0000                ; pass  register  carrying  active  parameters  upon
12 0000                ; entry   to   a   routine.      The  stack  is  used
13 0000                ; extensively; sufficient space must be provided  by
14 0000                ; the  calling  programs.    The  stack  pointer  is
15 0000                ; returned to its original place on exit  unless  an
16 0000                ; error   was   detected   (SP=?)   or   parameters  are
17 0000                ; returned on the stack.  In  the  latter  case,  the
18 0000                ; stack is offset by 2 times the requested number of
19 0000                ; parameters  and  will  be  set  right  after these
20 0000                ; parameters are popped off the stack.
21 0000                ;
22 0000                ; Register use conforms to  ICOM  and  CP/M  defined
23 0000                ; conventions:  Output  data  is  passed  in  the  C
24 0000                ; register and input  data  is  expected  in  the  A
25 0000                ; register.   These routines require CP/M-compatible
26 0000                ; CONIN and CONOUT  routines  as  contained  in  the
27 0000                ; user's  BIOS  program, or CI and CO as in the ICOM
28 0000                ; Resident ROM.
29 0000                ;
30 0000   000A         LF      EQU     0AH     ; ASCII line feed
31 0000   000D         CR      EQU     0DH     ; ASCII carriage return
32 0000   0040         CNTL    EQU     40H     ; ASCII Cntl offset
33 0000   0040         STACK   EQU     40H
34 0000                ;
35 0000                ;
36 0000                ;
37 0000   0040                 ORG     40H
38 0040                ;
39 0040 C38F03                 JMP     INIT
40 0043                ;
41 0043   0100                 ORG     0100H
42 0100                ;
43 0100                ; SYSTEM LINKAGES
44 0100                ;
45 0100   C003         CONIN   EQU     0C003H
46 0100   C006         CONOUT  EQU     0C006H
47 0100   C373         CONST   EQU     0C373H
48 0100   C000         USER    EQU     0C000H
49 0100                ;
50 0100 C38F03                 JMP     INIT
51 0103 C303C0         CONI:   JMP     CONIN
52 0106 C306C0         CONO:   JMP     CONOUT
53 0109 C373C3         CST:    JMP     CONST
54 010C C300C0         ERR:    JMP     USER
55 010F                ;
```

```
 56 010F           ; Routine BLK prints one blank on the current
 57 010F           ; console device.
 58 010F           ;
 59 010F           ; Entry parameters:     None
 60 010F           ; Return parameters:    None
 61 010F           ; Stack usage:          4 bytes
 62 010F           ;
 63 010F C5        BLK:    PUSH    B       ; Save (BC)
 64 0110 0E20              MVI     C,' '   ; Get an ASCII space
 65 0112 C34901            JMP     ECH2    ; Go output it
 66 0115           ;
 67 0115           ; Routine CONV converts a 4 bit binary number to its
 68 0115           ; ASCII equivalent. The high-order 4 accumulator
 69 0115           ; bits are lost.
 70 0115           ;
 71 0115           ; Entry parameter:      4 bit binary number in
 72 0115           ;                       lower half of accumulator
 73 0115           ; Exit parameter:       ASCII character in (A)
 74 0115           ; Stack usage:          0 bytes
 75 0115           ;
 76 0115 E60F      CONV:   ANI     0FH     ; Clear high bits
 77 0117 C690              ADI     90H     ; Insert partial ASCII
 78 0119 27                DAA             ; Zone
 79 011A CE40              ACI     40H     ; Insert rest of ASCII
 80 011C 27                DAA             ; Zone
 81 011D C9                RET
 82 011E           ;
 83 011E           ; Routine CRLF prints an ASCII carriage return and
 84 011E           ; line feed (in that order) on the console. It
 85 011E           ; follows these with 4 blanks to create a left
 86 011E           ; margin.
 87 011E           ;
 88 011E           ; Entry parameter:      None
 89 011E           ; Exit parameter:       None
 90 011E           ; Stack Usage:          8 bytes
 91 011E           ;
 92 011E E5        CRLF:   PUSH    H       ; Save (H,L)
 93 011F 212701            LXI     H,CRMSG ; Get message address
 94 0122 CDAE01            CALL    PRTMSG  ; Print message
 95 0125 E1                POP     H       ; Restore (HL)
 96 0126 C9                RET
 97 0127           ;
 98 0127 0D0A20A0  CRMSG:  DB      CR,LF,' ',' '+80H
 99 012B           ;
100 012B           ; Routine DEPRT prints the contents of the (DE)
101 012B           ; register pair as a 4-digit hexadecimal number on
102 012B           ; the console.
103 012B           ;
104 012B           ; Entry parameter:      (DE) = 4 digit hex number
105 012B           ;                       to be printed on console.
106 012B           ; Exit parameter:       None
107 012B           ; Stack usage:          10 bytes
108 012B           ;
109 012B CD1E01    DEPRT:  CALL    CRLF    ; Print a CR, LF
110 012E           ; Alternate entry point if no CR, LF wanted
```

```
111 012E 7A        DEPRA:  MOV     A,D     ; Get high order byte
112 012F CD3301            CALL    HEX2    ; Print 2 numbers
113 0132 7B                MOV     A,E     ; Get low order byte
114 0133           ; Alternate entry point to print (A) as two hex
115 0133           ; digits
116 0133 F5        HEX2:   PUSH    PSW     ; Save low order byte
117 0134 0F                RRC             ; Move high order nibble
118 0135 0F                RRC             ; to lower half of (A)
119 0136 0F                RRC
120 0137 0F                RRC
121 0138 CD3C01            CALL    HEX1    ; Print the nibble
122 013B F1                POP     PSW     ; Get low nibble back
123 013C           ; Alternate entry point to print low order nibble
124 013C           ; on console
125 013C CD1501    HEX1:   CALL    CONV    ; Convert to ASCII
126 013F C34501            JMP     ECH1    ; Go print it
127 0142           ;
128 0142           ; Routine ECHO reads one character from the calling
129 0142           ; routine and then echoes it back. It is assumed
130 0142           ; that the console is in a full duplex mode.
131 0142           ;
132 0142           ; Entry parameter:      None
133 0142           ; Exit parameter:       (A) = Character read from
134 0142           ;                       the console keyboard
135 0142           ; Stack usage:          4 bytes
136 0142           ;
137 0142 CD0301    ECHO:   CALL    CONI    ; Read a character
138 0145           ; Alternate entry point to print (A)
139 0145 C5        ECH1:   PUSH    B       ; Save (BC)
140 0146 E67F              ANI     7FH     ; Strip off parity bit
141 0148 4F                MOV     C,A     ; Put character into (C)
142 0149           ; Alternate entry point for BLK routine
143 0149 CD0601    ECH2:   CALL    CONO    ; Output it
144 014C C1                POP     B       ; Restore (BC)
145 014D C9                RET
146 014E           ;
147 014E           ; Routine HLPRT prints the contents of the (HL)
148 014E           ; register as 4 hexadecimal digits on the console.
149 014E           ;
150 014E           ; Entry parameter:      (HL) = 4 hex digit number
151 014E           ;                       to be printed
152 014E           ; Exit parameter:       None
153 014E           ; Stack usage:          10 bytes
154 014E           ;
155 014E CD1E01    HLPRT:  CALL    CRLF    ; Print a (CR,LF)
156 0151           ; Alternate entry point if no CR,LF wanted
157 0151 EB        HLPRA:  XCHG            ; Swap (HL), (DE)
158 0152 CD2E01            CALL    DEPRA   ; Go print (DE)
159 0155 EB                XCHG            ; Unswap (HL), (DE)
160 0156 C9                RET
161 0157           ;
162 0157           ; Routine PCHK reads a character from the console
163 0157           ; and checks whether it is a valid delimiter (space,
164 0157           ; comma, or carriage return). If so, a zero is
165 0157           ; returned in the status flags. If the character is
```

```
                        ; a  carriage return, the carry bit is set also.  If
                        ; it  is  not  a  delimiter,  a  non-zero,  no-carry
                        ; indication is required.
                        ;
                        ; Entry parameters:      None
                        ; Exit Parameters:       See description above.
                        ; Stack usage:           6 bytes
                        ;
0157 CD4201             PCHK:   CALL    ECHO    ;Read a character
015A                    ; Alternate entry point if CHAR already in (A)
015A FE20               PCH2:   CPI     ' '     ; Check for a blank
015C C8                         RZ              ; Return if (SO)
015D FE2C                       CPI     ','     ; Check for a comma
015F C8                         RZ              ; Return if (SO)
0160 FE0D                       CPI     'M'-CNTL
0162                                            ; Check for a CAR RET
0162 37                         STC             ; Set the carry flag
0163 C8                         RZ              ; Return if CAR RET
0164 3F                         CMC             ; Reset the carry flag
0165 C9                         RET
0166                    ;
0166                    ; Routine  PRM reads characters from the console and
0166                    ; pushes them onto the stack.  Multiple  parameters
0166                    ; may  be  read:  values are delimited by a space or
0166                    ; comma.  If a carriage return is entered, PRM stops
0166                    ; reading values and returns to  the  caller.  Only
0166                    ; the  last  4  characters of a string are saved; to
0166                    ; correct  an  error,  type  until  the  last  four
0166                    ; characters  are  correct.  The caller may retrieve
0166                    ; the  values  by  popping  them  from  the  stack,
0166                    ; last-entered character first.
0166                    ;
0166                    ; Entry parameter:      (C) = number of expected
0166                    ;                       parameters
0166                    ; Exit parameters:      (C) Parameters on stack:
0166                    ;                       If a bad value was entered,
0166                    ;                       '????' is printed and
0166                    ;                       control transferred to a
0166                    ;                       user provided error handler.
0166                    ;                       The stack pointer value is
0166                    ;                       indeterminate and needs
0166                    ;                       to be reset
0166                    ; Stack usage:          4 + 2 = (C) bytes
0166                    ;
0166                    ; Alternate entry point if only one parameter is
0166                    ; desired.
0166 0E01               PARM1:  MVI     C,1
0168                    ; Normal entry point
0168 210000             PRM:    LXI     H,0     ; Set (HL) = 0
016B CD4201             PRA:    CALL    ECHO    ; Get a character
016E 47                 PRB:    MOV     B,A     ; Save input character
016F CD9901                     CALL    NIBBL   ; Check it and CVB
0172 DA7E01                     JC      PRC     ; Not hex, see if delim
0175 29                         DAD     H       ; Multiply (HL) by 16
0176 29                         DAD     H
```

```
221 0177 29              DAD     H
222 0178 29              DAD     H
223 0179 B5              ORA     L       ; Add on new 4 bits
224 017A 6F              MOV     L,A
225 017B C36B01          JMP     PRA     ; Go get next character
226 017E             ;
227 017E E3          PRC:    XTHL        ; Swap value and RET ADDR
228 017F E5              PUSH    H       ; Resave return address
229 0180 78              MOV     A,B     ; Get last input char
230 0181 CD5A01          CALL    PCH2    ; See if delimiter
231 0184 D28901          JNC     PRD     ; Not a carriage return
232 0187 0D              DCR     C       ; CR, see if all values in
233 0188 C8              RZ              ; Yes, done
234 0189 C2C401      PRD:    JNZ     QPRT    ; Take error exit if not 0
235 018C 0D              DCR     C       ; All in?
236 018D C26801          JNZ     PRM     ; No, go get another
237 0190 C9              RET
238 0191             ;
239 0191             ; Alternate entry point if only one parameter
240 0191             ; wanted and first character already in (A).
241 0191 0E01        PRF:    MVI     C,1
242 0193 210000          LXI     H,0     ; Set up (HL)
243 0196 C36E01          JMP     PRB     ; Go get rest of parameter
244 0199             ;
245 0199             ; Routine  NIBBL  strips  the  ASCII  zone  off  a
246 0199             ; character in the (A) register and verifies that it
247 0199             ; is a valid hex digit.  If so, the binary value  is
248 0199             ; returned  to the lower half of the A register; the
249 0199             ; upper half is set to zero.  If not, the carry flag
250 0199             ; is set and control returned to the caller.
251 0199             ;
252 0199             ; Entry Parameter:      (A) = ASCII CHAR
253 0199             ; Exit parameters:      See description above
254 0199             ; Stack usage:          None
255 0199             ;
256 0199 D630        NIBBL:  SUI     '0'     ; Strip off 0-9 Zone
257 019B D8              RC              ; Invalid value RET
258 019C C6E9            ADI     'O'-'G' ; Strip off (AF) zone
259 019E D8              RC              ; Invalid value RET
260 019F C606            ADI     6       ; Sort out in-between values
261 01A1 F2A701          JP      NIO     ; Jump if (AF)
262 01A4 C607            ADI     7       ; Insure it is 0-9
263 01A6 D8              RC              ; wasn't: Return
264 01A7 C60A        NIO:    ADI     10      ; Adjust binary value
265 01A9 B7              ORA     A       ; Reset carry bit
266 01AA C9              RET
267 01AB             ;
268 01AB             ; Routine PRTWD prints a  character  string  on  the
269 01AB             ; console.    Depending on the entry point, a CR and
270 01AB             ; LF  may  be  printed  first.    Three  forms  of
271 01AB             ; message-end  delimiters  are  accepted: Bit 7=1 in
272 01AB             ; last character to be output; ASCII ETX  (CNTRL  C)
273 01AB             ; following  the last character; or a user-specified
274 01AB             ; delimiter following the last  character.   If  the
275 01AB             ; last  option  is used, (B) must have the delimiter
```

```
                        ; on entry to PRTA.
                        ;
                        ; Entry Parameters:     (HL) = Message start address
                        ;                       (B) = ETX delimiter (See
                        ;                       description above.)
                        ; Exit Parameters:      None - (HL) is altered
                        ; Stack usage:          12 bytes MAX
                        ;
                        ; Entry point for CR,LF (will not work with user
                        ; defined ETX delimiter).
01AB CD1E01             PRTWD:  CALL    CRLF
                        ; Entry point for No. CR,LF and a bit 7 or ASCII
                        ; ETX Delimiter.
01AE C5                 PRTWA:  PUSH    B       ; Save (BC)
01AF 0603                       MVI     B,3     ; Get an ASCII ETX
01B1 CDB601                     CALL    PRTA    ; Print message
01B4 C1                         POP     B       ; Restore (BC)
01B5 C9                         RET
                        ;
                        ; Entry point for user defined ETX delimiter
01B6 78                 PRTA:   MOV     A,B     ; Put ETX in A
01B7 4E                         MOV     C,M     ; Get next character
01B8 B9                         CMP     C       ; EOM?
01B9 C8                         RZ              ; Yes, done
01BA CD0601                     CALL    CONO    ; No, output it
01BD 79                         MOV     A,C     ; Retrieve CHAR
01BE 23                         INX     H       ; Point to next CHAR
01BF B7                         ORA     A       ; See if bit 7 is set
01C0 F2B601                     JP      PRTA    ; No, continue
01C3 C9                         RET
                        ;
                        ; Routine QPRT prints "????" and transfers control
                        ; to the user's error- recovery routine. (SP) is
                        ; indeterminate on exit.
                        ;
01C4 21CD01             QPRT:   LXI     H,QMSG  ; Message address
01C7 CDAE01                     CALL    PRTWA   ; Print it
01CA C30C01                     JMP     ERR     ; Go to error recovery
                        ;
01CD 3F3F3FBF           QMSG:   DB      '???','?'+80H
                        ;
                        ;
                        ; Hardware diagnostics can be divided into 3 stages:
                        ;         1) fault detection
                        ;         2) fault isolation
                        ;         3) fault correction
                        ; These routines automate the first stage only. See
                        ; the user's manual for guidelines for the second
                        ; stage. After the second step is completed, fault
                        ; correction should be no trouble.
                        ;
                        ;
                        ; SUBROUTINES FOR THE MEMORY DIAGNOSTICS
                        ;
                        ; When a bad memory cell is detected, this routine
```

```
331 01D1                ; is called to print the bad address, bad data, test
332 01D1                ; address, and test data (in that order).  With this
333 01D1                ; error log,  the  fault  isolation  process can be
334 01D1                ; conducted.
335 01D1                ;
336 01D1 CD2B01     ADPRT:  CALL    DEPRT   ; Print bad address
337 01D4 CD0F01             CALL    BLK     ; Print a blank
338 01D7 78                 MOV     A,B     ; Get a bad data
339 01D8 C3E001             JMP     ADPRB
340 01DB                ;
341 01DB                ; Alternate entry point when bad address is
342 01DB                ; meaningless
343 01DB F5         ADPRA:  PUSH    PSW
344 01DC CD1E01             CALL    CRLF    ; Do a (CR,LF)
345 01DF F1                 POP     PSW
346 01E0 CD3301     ADPRB:  CALL    HEX2    ; Print bad data
347 01E3 CD0F01             CALL    BLK
348 01E6 CD0F01             CALL    BLK
349 01E9 CD5101             CALL    HLPRA   ; Print test address
350 01EC CD0F01             CALL    BLK
351 01EF 79                 MOV     A,C     ; Get test data
352 01F0 C33301             JMP     HEX2    ; Print it
353 01F3                ;
354 01F3                ; Routine BREAK tests the console status to see if a
355 01F3                ; character has been typed in.  If so, it checks  to
356 01F3                ; see  if  it  is an ASCII ETX (CNTRL C).  If so, it
357 01F3                ; types an "ABORT" message and  returns  control  to
358 01F3                ; the calling routine.
359 01F3                ;
360 01F3 CD0901     BREAK:  CALL    CST     ; Character waiting?
361 01F6 C8                 RZ              ; No, return
362 01F7 CD0301             CALL    CONI    ; Yes, get it
363 01FA FE03               CPI     'C'-CNTL
364 01FC                                    ; See if Cntl C
365 01FC C0                 RNZ             ; No, return
366 01FD 210702             LXI     H,ABMSG ; Print out the
367 0200 CDAB01             CALL    PRTWD   ; 'ABORT' message
368 0203 313E00             LXI     SP,STACK-2
369 0206                                    ; Reset the stack
370 0206 C9                 RET             ; Return to exec
371 0207                ;
372 0207 41424F52   ABMSG:  DB      'ABOR','T'+80H
    020B D4
373 020C                ;
374 020C                ; Routine  PARM reads in the desired test block size
375 020C                ; and  block  base  address.  Both  parameters  are
376 020C                ; pushed onto the stack.
377 020C                ;
378 020C CDAE01     PARM:   CALL    PRTWA   ; Print caller's name
379 020F 212402             LXI     H,BZMSG ; Print BLOCK SIZE message
380 0212 CDAB01             CALL    PRTWD
381 0215 CD6601             CALL    PARM1   ; Get block size
382 0218 E1                 POP     H       ; Retrieve it
383 0219 E3                 XTHL
384 021A E5                 PUSH    H       ; Save return address
```

```
385 021B 213002     PARMA:  LXI     H,BAMSG ; Print BASE ADDRESS
386 021E CDAB01             CALL    PRTWD   ; message
387 0221 C36601             JMP     PARM1   ; Get it and return
388 0224            ;
389 0224 424C4F43   BZMSG:  DB      'BLOCK SIZE:',' '+80H
    0228 4B205349
    022C 5A453AA0
390 0230 42415345   BAMSG:  DB      'BASE'
391 0234 20414444   ADMSG:  DB      ' ADDRESS:',' '+80H
    0238 52455353
    023C 3AA0
392 023E            ;
393 023E            ; Routine MADT performs a "Walking Bit" test on both
394 023E            ; the data and address lines of a 2114 pair at the
395 023E            ; same time. First, it zeros all cells in the
396 023E            ; specified block, then ensures that they are all
397 023E            ; zero. It tests each 1K section separately.
398 023E            ; Detected errors are logged on the console as they
399 023E            ; occur.
400 023E            ;
401 023E            ; The base address, when asked for, must be on a 1K
402 023E            ; boundary or it will be rejected and another
403 023E            ; address asked for.
404 023E            ;
405 023E            ; The operator can abort the test at any time by
406 023E            ; typing ETX (CNTRL C) should too many errors be
407 023E            ; detected. Allowing the test to complete will
408 023E            ; ensure adequate data for thorough fault isolation.
409 023E            ;
410 023E            ; Without errors, this diagnostic tests a 1K cell in
411 023E            ; approximately 2 seconds.
412 023E            ;
413 023E 217F02     MADT:   LXI     H,WBMSG ; Sign on
414 0241 CD0C02             CALL    PARM    ; Get parameters
415 0244 E1         MADTA:  POP     H       ; Retrieve BASE ADDRESS
416 0245 D1                 POP     D       ; Retrieve BLOCK SIZE
417 0246 7C                 MOV     A,H     ; Test for 1K boundary
418 0247 E603               ANI     3
419 0249 B5                 ORA     L
420 024A CA6002             JZ      MADTB   ; OK, jump
421 024D D5                 PUSH    D       ; Save block size
422 024E 217B02             LXI     H,BEMSG ; Reject base address
423 0251 CDAB01             CALL    PRTWD
424 0254 213002             LXI     H,BAMSG
425 0257 CDAE01             CALL    PRTWA
426 025A CD1B02             CALL    PARMA   ; Ask for another
427 025D C34402             JMP     MADTA   ; Test it again
428 0260            ;
429 0260 CD9902     MADTB:  CALL    ZTBK    ; Zero the block
430 0263 D5         MADTC:  PUSH    D       ; Save block size
431 0264 3E04               MVI     A,4     ; Set 1K sections
432 0266 BA                 CMP     D       ; See if < 1K
433 0267 F26B02             JP      MADTD   ; Yes, test it
434 026A 57                 MOV     D,A     ; No, set to 1K
435 026B CDBB02     MADTD:  CALL    WLKAD   ; Test it
```

```
436 026E E1                 POP     H       ; Get remaining size
437 026F 7D                 MOV     A,L     ; Subtract tested size
438 0270 93                 SUB     E
439 0271 6F                 MOV     L,A
440 0272 7C                 MOV     A,H
441 0273 9A                 SBB     D
442 0274 67                 MOV     H,A
443 0275 C8                 RZ              ; Return if done
444 0276 EB                 XCHG            ; (DE) = untested
445 0277                                    ; (HL) = previous increment
446 0277 09                 DAD     B       ; Set new base address
447 0278 C36302             JMP     MADTC   ; Do it again
448 027B            ;
449 027B 424144A0   BEMSG:  DB      'BAD',' '+80H
450 027F 57414C4B   WBMSG:  DB      'WALKING BIT TEST',' '+80H
    0283 494E4720
    0287 42495420
    028B 54455354
    028F A0
451 0290 54455354   TDMSG:  DB      'TEST DON','E'+80H
    0294 20444F4E
    0298 C5
452 0299            ;
453 0299            ; Routine ZTBK zeros and tests for a contiguous
454 0299            ; block of memory.  On entry, the (DE) register must
455 0299            ; have the block size and the (HL) register must
456 0299            ; have the base address.  These values are restored
457 0299            ; to the registers on exit from the routine.
458 0299            ;
459 0299 D5         ZTBK:   PUSH    D       ; Save block size
460 029A E5                 PUSH    H       ; Save base address
461 029B 0E00               MVI     C,0
462 029D 71         ZTBKA:  MOV     M,C     ; Write into the block
463 029E 23                 INX     H       ; Next address
464 029F 1B                 DCX     D       ; Loop control
465 02A0 7B                 MOV     A,E
466 02A1 B2                 ORA     D
467 02A2 C29D02             JNZ     ZTBKA   ; Loop if not zeroed
468 02A5 E1                 POP     H       ; Restore registers
469 02A6 D1                 POP     D
470 02A7 D5                 PUSH    D       ; Save parameters
471 02A8 E5                 PUSH    H
472 02A9 7E         ZTBKB:  MOV     A,M     ; Read  a cell
473 02AA B9                 CMP     C       ; Same as written?
474 02AB C4DB01             CNZ     ADPRA   ; Log error if necessary
475 02AE CDF301             CALL    BREAK   ; See if abort wanted
476 02B1 23                 INX     H       ; Next address
477 02B2 1B                 DCX     D       ; Loop control
478 02B3 7B                 MOV     A,E
479 02B4 B2                 ORA     D
480 02B5 C2A902             JNZ     ZTBKB   ; Loop if more to do
481 02B8 E1                 POP     H       ; Restore base address
482 02B9 D1                 POP     D       ; Restore block size
483 02BA C9                 RET
484 02BB            ;
```

```
485 02BB           ; Routine WLKAD walks a single high bit through each
486 02BB           ; data bit of all addresses in a controlled  manner.
487 02BB           ; After  a  bit  is written, all other locations are
488 02BB           ; tested for zeros.  When an error is  detected,  it
489 02BB           ; is  logged  as  described above.  If excess errors
490 02BB           ; occur, abort the test by typing CNTRL C.
491 02BB           ;
492 02BB D5        WLKAD:  PUSH    D       ; Save block size
493 02BC E5                PUSH    H       ; Save address
494 02BD 23                INX     H       ; Set A0
495 02BE 0E11      WLKDA:  MVI     C,11H   ; Set D0, D4 (2114)
496 02C0 C5        WLKC:   PUSH    B       ; Save it
497 02C1 71                MOV     M,C     ; Write byte into memory
498 02C2 E5                PUSH    H       ; Save current address
499 02C3 33                INX     SP      ; Adjust stack to
500 02C4 33                INX     SP      ; find base address
501 02C5 33                INX     SP
502 02C6 33                INX     SP
503 02C7 E1                POP     H       ; Retrieve base address
504 02C8 E5                PUSH    H       ; Restore it
505 02C9 3B                DCX     SP      ; Readjust stack
506 02CA 3B                DCX     SP
507 02CB 3B                DCX     SP
508 02CC 3B                DCX     SP
509 02CD 7E        WLKB:   MOV     A,M     ; Read byte
510 02CE 47                MOV     B,A     ; Save byte in (B)
511 02CF A7                ANA     A       ; Test data
512 02D0 EB                XCHG
513 02D1 E3                XTHL            ; Get test address
514 02D2                                   ; Save loop control
515 02D2 C2DE02            JNZ     DNZT    ; Non-zero data, jump
516 02D5 CD1703            CALL    CHLDE   ; Test addresses
517 02D8 CCD101            CZ      ADPRT   ; Bad cell
518 02DB C3E802            JMP     CONT    ; Continue test
519 02DE           ;
520 02DE B9        DNZT:   CMP     C       ; See if same as test data
521 02DF C2E502            JNZ     BADD    ; Jump if bad data
522 02E2 CD1703            CALL    CHLDE   ; Test addresses
523 02E5 C4D101    BADD:   CNZ     ADPRT
524 02E8 CDF301    CONT:   CALL BREAK      ; See if abort wanted
525 02EB E3                XTHL            ; Unscramble registers
526 02EC EB                XCHG
527 02ED 23                INX     H       ; Next address
528 02EE 1B                DCX     D
529 02EF 7B                MOV     A,E
530 02F0 B2                ORA     D       ; Done on this cell?
531 02F1 C2CD02            JNZ     WLKB    ; No, jump
532 02F4 E1                POP     H       ; Get test address
533 02F5 C1                POP     B       ; Get data
534 02F6 33                INX     SP
535 02F7 33                INX     SP
536 02F8 D1                POP     D       ; Get block size
537 02F9 D5                PUSH    D
538 02FA 3B                DCX     SP
539 02FB 3B                DCX     SP
```

```
02FC 79              MOV     A,C     ; Get data into (A)
02FD 07              RLC             ; Shift for next pattern
02FE 4F              MOV     C,A
02FF D2C002          JNC     WLKC    ; Not done yet
0302 C1              POP     B       ; Get base address
0303 D1              POP     D       ; Get block size
0304 3600            MVI     M,0     ; Reset test cell
0306 7D              MOV     A,L     ; Strip off base
0307 91              SUB     C       ; address
0308 6F              MOV     L,A
0309 7C              MOV     A,H
030A 98              SBB     B
030B 67              MOV     H,A
030C 29              DAD     H       ; Go to next address bit
030D CD1703          CALL    CHLDE   ; See if done
0310 F0              RP              ; Yes, return
0311 09              DAD     B       ; Build next address
0312 D5              PUSH    D       ; Save block size
0313 C5              PUSH    B       ; Save base address
0314 C3BE02          JMP     WLKDA   ; Go do it again
0317            ;
0317            ; Compare (HL) register to (DE) register and set
0317            ; flags on result.
0317            ;
0317 7C         CHLDE:  MOV     A,H
0318 92                 SUB     D
0319 C0                 RNZ
031A 7D                 MOV     A,L
031B 93                 SUB     E
031C C9                 RET
031D            ;
031D            ; Routine BRNIN continuously writes a sequence of
031D            ; non-zero numbers into a specified memory block and
031D            ; reads them back for comparison. If errors occur,
031D            ; they are logged on the console. A running error
031D            ; total is also maintained. The test may be
031D            ; terminated at any time with a CNTRL C; the error
031D            ; total at this time will be displayed on the
031D            ; console. The test data steps from 1 to 255
031D            ; decimal, then repeats itself, always skipping 0.
031D            ;
031D            ;
031D 217703     BRNIN:  LXI     H,CBMSG ; Get message address
0320 CD0C02             CALL    PARM    ; Write it, get parameters
0323 E1                 POP     H       ; Get base address
0324 D1                 POP     D       ; Get block size
0325 0E01               MVI     C,1     ; Seed the data
0327 0600               MVI     B,0     ; Initialize error count
0329 C5         BRNA:   PUSH    B       ; Save data, error count
032A D5                 PUSH    D       ; Save block size
032B E5                 PUSH    H       ; Save base address
032C 71         BRNB:   MOV     M,C     ; Write the data byte
032D 0C                 INR     C       ; Advance data patern
032E C23203             JNZ     BRNC    ; Skip 0
0331 0C                 INR     C       ; Set to 1
```

```
595 0332 23          BRNC:   INX     H       ; Go to next address
596 0333 1B                  DCX     D       ; Do loop control
597 0334 7B                  MOV     A,E
598 0335 B2                  ORA     D
599 0336 C22C03              JNZ     BRNB
600 0339 E1                  POP     H       ; Get base address
601 033A D1                  POP     D       ; Get block size
602 033B C1                  POP     B       ; Get data seed, error cou
603 033C D5                  PUSH    D       ; Restore them
604 033D E5                  PUSH    H
605 033E 7E          BRND:   MOV     A,M     ; Read data byte
606 033F B9                  CMP     C       ; Check it
607 0340 CA4703              JZ      BRNE    ; Skip if OK
608 0343 04                  INR     B       ; Error count
609 0344 CDDB01              CALL    ADPRA   ; Log the error
610 0347 0C          BRNE:   INR     C       ; Change test data
611 0348 C24C03              JNZ     BRNF    ; Skip if not zero
612 034B 0C                  INR     C       ; Reset to 1
613 034C 23          BRNF:   INX     H       ; Next address
614 034D 1B                  DCX     D       ; Loop control
615 034E 7B                  MOV     A,E
616 034F B2                  ORA     D
617 0350 C23E03              JNZ     BRND
618 0353 E1                  POP     H       ; Reset base address
619 0354 D1                  POP     D       ; and block size
620 0355 CD0901              CALL    CST     ; Time to quit
621 0358 CA2903              JZ      BRNA    ; No, do it again
622 035B CD0301              CALL    CONI    ; Get character
623 035E FE03                CPI     'C'-CNTL
624 0360                                     ; ETX (Cntl C)?
625 0360 C22903              JNZ     BRNA    ; No, continue
626 0363 CD1E01              CALL    CRLF
627 0366 78                  MOV     A,B     ; Error count
628 0367 CD3301              CALL    HEX2    ; Print it
629 036A 217003              LXI     H,ERMSG ; Get error message addres
630 036D C3AE01              JMP     PRTWA   ; Print it and return to E
631 0370             ;
632 0370 20455252    ERMSG:  DB      ' ERROR','S'+80H
    0374 4F52D3
633 0377 434F4E54    CBMSG:  DB      'CONTINUOUS BURNIN',' '+80H
    037B 494E554F
    037F 55532042
    0383 55524E49
    0387 4EA0
634 0389             ;
635 0389             ; Routines INIT and EXEC initialize the computer a
636 0389             ; monitor  the  console for a command.  When a val
637 0389             ; command is received, control is transferred to t
638 0389             ; appropriate routine.
639 0389             ;
640 0389 219002      RETN:   LXI     H,TDMSG ; Print 'TEST DONE'
641 038C CDAB01              CALL    PRTWD
642 038F 3140009FE7  INIT:   LXI     SP,STACK        ; Set stack point
643 0392 21AC03      EXEC:   LXI     H,DIMSG ; Print diag message
644 0395 CDAB01              CALL    PRTWD
```

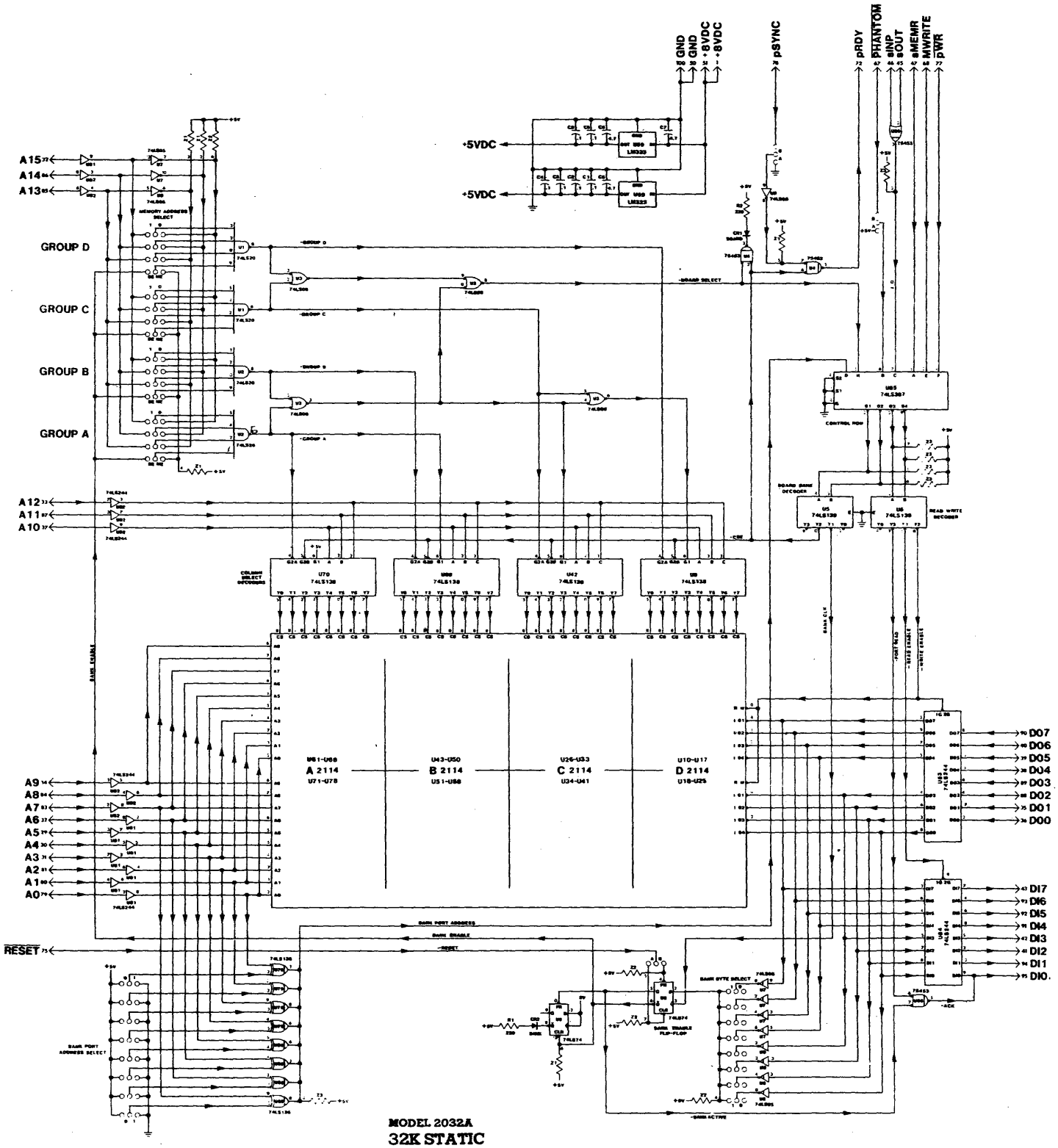
```
 645 0398 218903             LXI     H,RETN  ; Set up return address
 646 039B E5                 PUSH    H
 647 039C CD0301             CALL    CONI    ; Wait for command
 648 039F FE43               CPI     'C'     ; Continuous burn-in
 649 03A1 CA1D03             JZ      BRNIN
 650 03A4 FE57               CPI     'W'     ; Walking bit
 651 03A6 CA3E02             JZ      MADT
 652 03A9 C3C401             JMP     QPRT
 653 03AC            ;
 654 03AC 44494147   DIMSG:  DB      'DIAGNOSTIC:',' '+80H
     03B0 4E4F5354
     03B4 49433AA0
 655 03B8            ;
 656 03B8   0000             END

TOTAL ERRORS=00
```

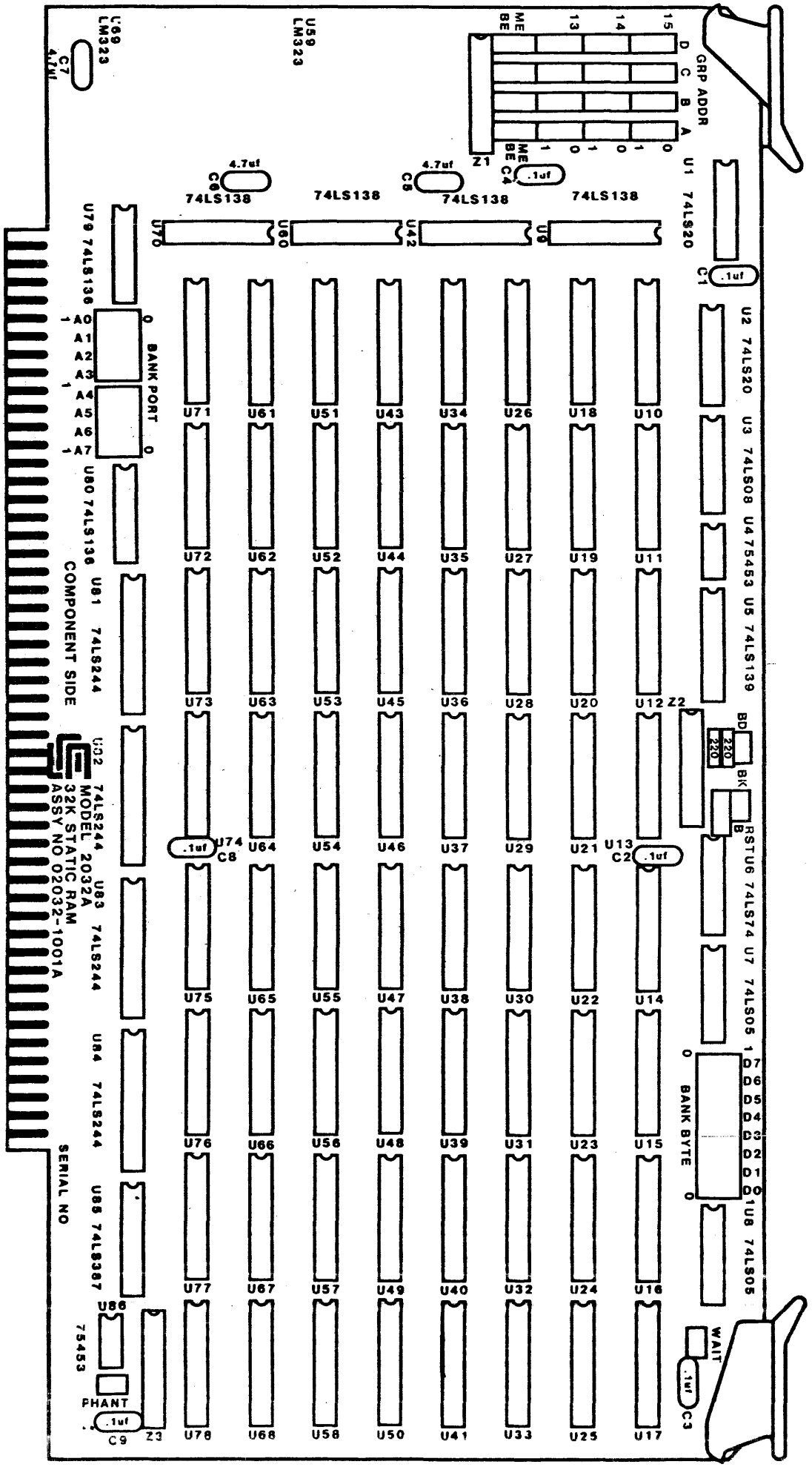
CHAPTER 4

TECHNICAL INFORMATION

4.1 SCHEMATIC/LOGIC DIAGRAM



MODEL 2032A
32K STATIC
RAM BOARD



4.2 ASSEMBLY COMPONENT LAYOUT

4.3 PARTS LIST

QTY	REFERENCE	DESCRIPTION	CCS PART #
CAPACITORS			
3	C5-7	Tantalum, 4.7uf, 35 vdc, 20%	42804-54756
6	C1-4,8-9	Ceramic, .1uf, 50 vdc, 20%	42142-21046
RESISTORS			
3	Z1-3	Network, SIP, 2.7K x 7	40930-72726
INTEGRATED CIRCUITS			
64	U10-41,43-58, 61-68,71-78	MOS 2114 1Kx4 Static RAMS	31900-21142 (200nsec) or -21143 (300nsec) or -21144 (450nsec)
2	U59,69	LM323 +5v regulator	32000-03230
2	U79,80	74LS136 quad ex-OR:OC	30000-00136
2	U1,2	74LS20 dual 4-in NAND	30000-00020
2	U7,8	74LS05 hex inverter:OC	30000-00005
4	U9,42,60,70	74LS138 octal decoder	30000-00138
2	U4,86	75453 dual 2-in OR: OC	30300-00453
1	U6	74LS74 dual D flip-flop	30000-00074
1	U3	74LS08 quad 2-in AND	30000-00008
1	U5	74LS139 2:4 decoders	30000-00139
4	U81-84	74LS244 Tri buffer	30000-00367
1	U85	ROM 5623 256x4	30900-05623

QTY	REFERENCE	DESCRIPTION	CCS PART #
IC SOCKETS			
2	XU4,86	IC Socket, 8 pin	58102-00080
8	XU1-3,6-8,79-80	IC Socket, 14 pin	58102-00140
6	XU5,9,42, 60,70,85	IC Socket, 16 pin	58102-00160
64	XU10-41,43-58, 61-68,71-78	IC Socket, 18 pin	58102-00180
4	XU81-84	IC Socket, 20 pin	58102-00200
MISCELLANEOUS			
35		Header Strip, 1x3	56004-01003
35		Berg Jumper	56200-00001
2	CR1,CR2	Diode, Light Emitting	37400-00001
2	XU59,69	Heatsink, Ahamtor 423	60022-00002
4		Screw, Phillips head (SIMS), 6-32x7/16	71006-32071
4		Nut, hex, 6-32 & lock washer (KEPS)	73006-32001
1		PC Board	02032-00002
2		Extractor, PCB Non-locking	60100-00000
2		Roll Pin Extractor Mounting	60100-00001
1		Owner's Manual	89000-02032

4.4 CONTROL ROM TRUTH TABLE

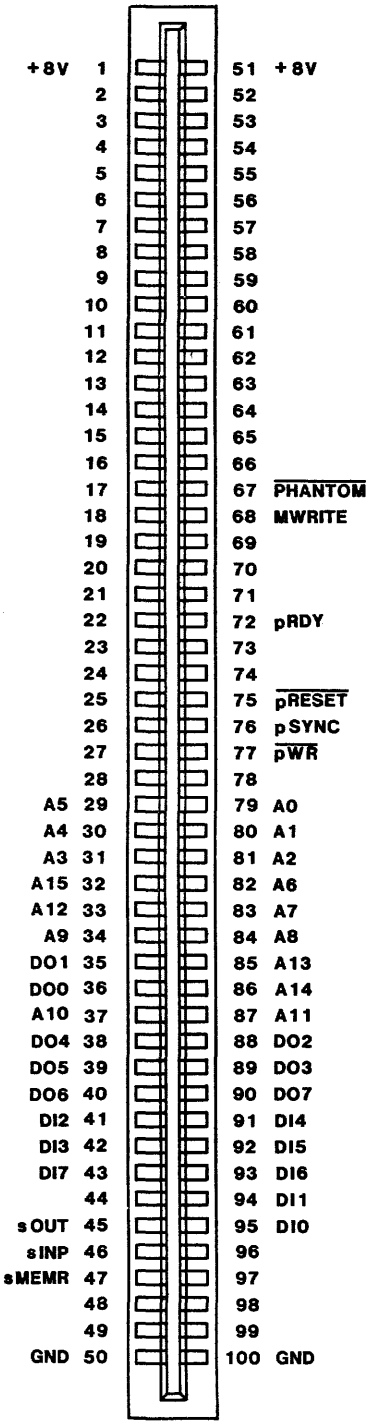
CONTROL ROM INPUTS									ROM OUTPUTS					DECODER OUTPUTS					OPERATION
HEX	-BOARD SEL	BNK PT ADDR	MWRITE	-pWR	GROUND	I/O	-PHANTOM	sMEMR	HEX	R/W DEC B	R/W DEC A	B/B DEC B	B/B DEC A	BANK CLK	-CSE	-RD ENABLE	-WR ENABLE	-PORT READ	
	A7	A6	A5	A4	A3	A2	A1	A0		04	03	02	01	Y1	Y2	Y1	Y2	Y3	
13	0	0	0	1	0	0	1	1	6	0	1	1	0	0	1	0	1	1	Bank-Independent Memory Read
22	0	0	1	0	0	0	1	0	A	1	0	1	0	0	1	1	0	1	Bank-Independent Memory Write (CPU)
32	0	0	1	1	0	0	1	0	A	1	0	1	0	0	1	1	0	1	Bank-Independent Memory Write (FP)
46	0	1	0	0	0	1	1	0	9	1	0	0	1	1	0	1	0	1	Write to Port, Memory Selected
53	0	1	0	1	0	0	1	1	6	0	1	1	0	0	1	0	1	1	Bank-Dependent Memory Read
56	0	1	0	1	0	1	1	0	C	1	1	0	0	1	1	1	1	0	Read from Port, Memory Selected
62	0	1	1	0	0	0	1	0	A	1	0	1	0	0	1	1	0	1	Bank-Dependent Memory Write (CPU)
72	0	1	1	1	0	0	1	0	A	1	0	1	0	0	1	1	0	1	Bank-Dependent Memory Write (FP)
C6	1	1	0	0	0	1	1	0	9	1	0	0	1	1	0	1	0	1	Write to Port, No Memory Selected
D6	1	1	0	1	0	1	1	0	C	1	1	0	0	1	1	1	1	0	Read from Port, No Memory Selected
any other location									0	0	0	0	0	1	1	1	1	1	

4.5 ADDRESS/CHIP TABLE

2032 ADDRESS/CHIP TABLE

LOW NIBBLES	C00-FFF		800-BFF		400-7FF		000-3FF		
	LOW	HIGH	LOW	HIGH	LOW	HIGH	LOW	HIGH	
HIGH NIBBLE									GROUP
W	U10	U11	U12	U13	U14	U15	U16	U17	D
W+1	U18	U19	U20	U21	U22	U23	U24	U25	
X	U26	U27	U28	U29	U30	U31	U32	U33	C
X+1	U34	U35	U36	U37	U38	U39	U40	U41	
Y	U43	U44	U45	U46	U47	U48	U49	U50	B
Y+1	U51	U52	U53	U54	U55	U56	U57	U58	
Z	U61	U62	U63	U64	U65	U66	U67	U68	A
Z+1	U71	U72	U73	U74	U75	U76	U77	U78	

4.6 2032 BUS CONNECTOR PINOUT



TOP VIEW

APPENDIX A

LIMITED WARRANTY

California Computer Systems (CCS) warrants to the original purchaser of its products that

(1) its CCS assembled and tested products will be free from materials defects for a period of one (1) year, and be free from defects of workmanship for a period of ninety (90) days; and

(2) its kit products will be free from materials defects for a period of ninety (90) days.

The responsibility of CCS hereunder, and the sole and exclusive remedy of the original purchaser for a breach of any warranty hereunder, is limited to the correction or replacement by CCS at CCS's option, at CCS's service facility, of any product or part which has been returned to CCS and in which there is a defect covered by this warranty; provided, however, that in the case of CCS assembled and tested products, CCS will correct any defect in materials and workmanship free of charge if the product is returned to CCS within ninety (90) days of original purchase from CCS; and CCS will correct defects in materials in its products and restore the product to an operational status for a labor charge of $25.00, provided that the product is returned to CCS within ninety (90) days in the case of kit products, or one (1) year in the case of CCS assembled and tested products. All such returned products shall be shipped prepaid and insured by original purchaser to:

Warranty Service Department
California Computer Systems
250 Caribbean Drive
Sunnyvale, California
94086

CCS shall have the right of final determination as to the existence and cause of a defect, and CCS shall have the sole right to decide whether the product should be repaired or replaced.

This warranty shall not apply to any product or any part thereof which has been subject to

(1) accident, neglect, negligence, abuse or misuse;

(2) any maintenance, overhaul, installation, storage, operation, or use, which is improper; or

(3) any alteration, modification, or repair by anyone other than CCS or its authorized representative.

THIS WARRANTY IS EXPRESSLY IN LIEU OF ALL OTHER WARRANTIES EXPRESSED OR IMPLIED OR STATUTORY INCLUDING THE WARRANTIES OF DESIGN, MERCHANTABILITY, OR FITNESS OR SUITABILITY FOR USE OR INTENDED PURPOSE AND OF ALL OTHER OBLIGATIONS OR LIABILITIES OF CCS. To any extent that this warranty cannot exclude or disclaim implied warranties, such warranties are limited to the duration of this express warranty or to any shorter time permitted by law.

CCS expressly disclaims any and all liability arising from the use and/or operation of its products sold in any and all applications not specifically recommended, tested, or certified by CCS, in writing. With respect to applications not specifically recommended, tested, or certified by CCS, the original purchaser acknowledges that he has examined the products to which this warranty attaches, and their specifications and descriptions, and is familiar with the operational characteristics thereof. The original purchaser has not relied upon the judgement or any representations of CCS as to the suitability of any CCS product and acknowledges that CCS has no knowledge of the intended use of its products. CCS EXPRESSLY DISCLAIMS ANY LIABILITY ARISING FROM THE USE AND/OR OPERATION OF ITS PRODUCTS, AND SHALL NOT BE LIABLE FOR ANY CONSEQUENTIAL OR INCIDENTAL OR COLLATERAL DAMAGES OR INJURY TO PERSONS OR PROPERTY.

CCS's obligations under this warranty are conditioned on the original purchaser's maintenance of explicit records which will accurately reflect operating conditions and maintenance preformed on CCS's products and establish the nature of any unsatisfactory condition of CCS's products. CCS, at its request, shall be given access to such records

for substantiating warranty claims. No action may be brought for breach of any express or implied warranty after one (1) year from the expiration of this express warranty's applicable warranty period. CCS assumes no liability for any events which may arise from the use of technical information on the application of its products supplied by CCS. CCS makes no warranty whatsoever in respect to accessories or parts not supplied by CCS, or to the extent that any defect is attributable to any part not supplied by CCS.

CCS neither assumes nor authorizes any person other than a duly authorized officer or representative to assume for CCS any other liability or extension or alteration of this warranty in connection with the sale or any shipment of CCS's products. Any such assumption of liability or modification of warranty must be in writing and signed by such duly authorized officer or representative to be enforceable. These warranties apply to the orginal purchaser only, and do not run to successors, assigns, or subsequent purchasers or owners; AS TO ALL PERSONS OR ENTITIES OTHER THAN THE ORIGINAL PURCHASER, CCS MAKES NO WARRANTIES WHATSOEVER, EXPRESS OR IMPLIED OR STATUTORY. The term "original purchaser" as used in this warranty shall be deemed to mean only that person to whom its product is originally sold by CCS.

Unless otherwise agreed, in writing, and except as may be necessary to comply with this warranty, CCS reserves the right to make changes in its products without any obligation to incorporate such changes in any product manufactured theretofore.

This warranty is limited to the terms stated herein. CCS disclaims all liability for incidental or consequential damages. Some states do not allow limitations on how long an implied warranty lasts and some do not allow the exclusion or limitation of incidental or consequential damages so the above limitations and exclusions may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.