TABLE OF CONTENTS

FEATURES

Available Assembled and Tested, As a Kit, and As a Bare Board

Available with 200, 300, or 450 nsec RAMs

Uses Popular 2114 Static RAMs

DIP Switches Used for Selectable Features

4K Memory Blocks Individually Addressable to Any 4K Boundary

Bank Selection by Bank Port and Bank Byte

Board Can Be Set Bank-Dependent or -Independent

LED's Indicate Board Active and Bank Active States

Wait State Switch

Phantom Line Capability

Selectable Board-Enable/Disable 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

Bare Board Purchasers Can Build Up Memory 4K at a Time

CHAPTER 1

SETTING UP

The CCS 2016B is a 16K byte static RAM board designed for use on S-100 busses. Thirty-two 1K x 4-bit static RAM chips are arranged in columns of two in order to provide an 8-bit byte, and the sixteen 8-bit columns are divided into 4-column memory groups A through D. Each memory group is individually addressed, and up to three memory groups can be buried to reconfigure the board to 4, 8, or 12K. 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 2016B with selectable addressing and several optional features. Selections are hard-wired with convenient DIP switches or by the installation of jumper wires. The addresses for each of the 4K memory groups, the bank port address, and the bank byte are set with switches, so you can choose the addresses that will best suit your system. The Bank Disable switch allows you to make the board independent of bank selection, while the Bank 0 switch lets you bring the board up active when powered on or cleared. For users with fast CPUs and slow RAMs, a Wait switch has been provided. Finally, Phantom and Clear options can be jumper-enabled as desired. Each switch- and jumper-selectable feature is discussed individually below. Further explanation of what the different settings cause to happen can be found in Chapter 3, "Theory of Operation."

1.1 SETTING THE MEMORY GROUP ADDRESSES

In order to provide maximum flexibility in the location of the 2016B's memory groups within a bank, CCS has made the addresses of the four memory groups switch-selectable. The switch-set addresses are compared with the high-order address lines A12-A15, and if an address matches, the corresponding memory group will be selected. Set the switches of each group to the binary equivalent of the high-order hex digit that specifies the 4K block of addresses in which you wish to locate the group. For example, the addresses of the block between 16K and 20K are 4000h-4FFFh, so you would locate a group in that block by setting its switches to 0100 (see section 1.9, the ADDRESS SWITCHES TABLE). Remember that A15 is the high-order binary digit, so you will set the binary addresses from right to left on the board.

The memory groups are fully prioritized, with A highest and D lowest. This allows you to give two (or more) memory groups the same address. Only the highest-priority group will be selected by that address; the RAMs of the other group(s) will be buried, permanently inaccessible and occupying no memory space until the address switches are reset. This allows you to configure the 2016B to 4, 8, or 12K without removing RAMs.

1.2 SETTING THE BANK BYTE

The bank-byte switches allow you to hardware-map the 2016B memory board to whichever of the eight memory bank levels 0-7 you choose. To select a bank level, set to 1 the switch that corresponds to the desired bank level and set all other switches to 0. For example, to select bank 3 you would set switch 3 to 1 and switches 0-2 and 4-7 to 0. Remember that on the board high-order is to the right rather than the left.

You may cause the board to be activated with more than one bank by setting the switches corresponding to each desired bank to 1.

1.3 SETTING THE BANK PORT ADDRESS

In order for bank-selection to be enabled, the correct address must be sent to the bank port. Most presently-marketed S-100 products using the bank port/bank byte scheme address the bank port at 40h. Therefore we recommend that you use this bank port address unless you have a strong reason for doing otherwise. Remember that A7 is the high-order bit; thus 40h is selected by setting switch 6 to 1 and switches 0-5 and 7 to 0.

1.4 SETTING BANK-INDEPENDENCE

The 2016B can be made independent of bank selection, causing it to be enabled whenever it is addressed regardless of which bank is active. This allows you to address the board without going through the bank select procedure. To make the 2016B bank-independent, close its Bank Disable switch (BD).

1.5 INSTALLING A CLEAR JUMPER

The Bank Byte Register is reset by -POC when the power is turned on. This causes the Register to represent a bank byte of 00h. You can cause the bank byte register to be cleared additionally by either -pRESET or -EXT CLR by installing a jumper wire. Install jumper W1 if you want the bank byte register to be cleared whenever you reset your peripherals (-EXT CLR). Install W2 if you want the register cleared whenever you reset your CPU (-pRESET). Do not install both jumpers.

1.6 SETTING THE BANK 0 SWITCH

If you want the board to come up active when the Bank Byte Register is cleared, close the B0 switch, along with the bit 0 switch of Bank Byte Select. When the computer is powered on (as well as when the CPU or the peripherals are reset if one of the Clear jumpers is installed), the board will come up active. Normal bank port/bank byte operation will not be affected.

1.7 INSTALLING THE PHANTOM JUMPER

Installing the Phantom jumper allows a device that generates a PHANTOM signal to overlay portions of the 2016B 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 2016B board, the Phantom signal will block the output from the 2016B of the contents of that location. This allows you to access the rest of the memory locations within the 4K block that contains the overlayed portion. Without Phantom capability the 2016B would not be able to locate a memory group in that block because the 2016B and the peripheral control board would both put data on the bus when a shared location was addressed.

1.8 SETTING THE WAIT SWITCH

The Wait switch allows you to slow down your processor every time the 2016B is addressed. This will be necessary if your processor allows less memory access time than your RAMs require.

If you have the 2016B 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 could, in theory at least, need to enable Wait. You should experiment, however; in most 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 2016B Wait feature. The 2016B 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. If you have 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.

1.9 ADDRESS SWITCHES TABLE

GROUP A = S4
GROUP B = S3
GROUP C = S2
GROUP D = S1

Starting Addr. Hex	Octal	A15	A14	A13	A12	Binary Number
0000	000:000	closed	closed	closed	closed	0000
1000	020:000	closed	closed	closed	open	0001
2000	040:000	closed	closed	open	closed	0010
3000	060:000	closed	closed	open	open	0011
4000	100:000	closed	open	closed	closed	0100
5000	120:000	closed	open	closed	open	0101
6000	140:000	closed	open	open	closed	0110
7000	160:000	closed	open	open	open	0111
8000	200:000	open	closed	closed	closed	1000
9000	220:000	open	closed	closed	open	1001
A000	240:000	open	closed	open	closed	1010
B000	260:000	open	closed	open	open	1011
C000	300:000	open	open	closed	closed	1100
D000	320:000	open	open	closed	open	1101
E000	340:000	open	open	open	closed	1110
F000	360:000	open	open	open	open	1111

closed = 0
open = 1

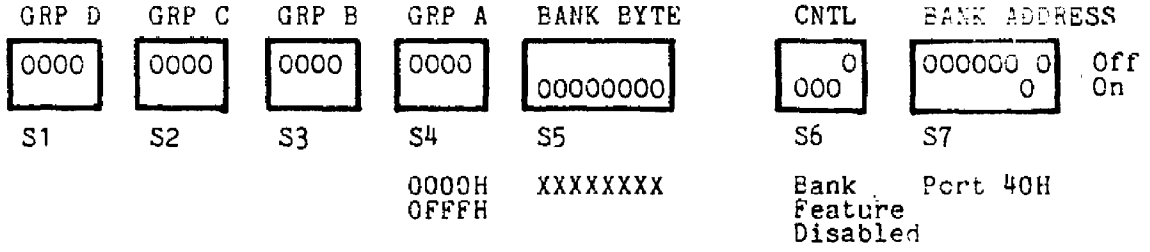
CHAPTER 2

TESTING AND TROUBLESHOOTING THE 2016B

2.1 FRONT PANEL QUICK CHECKOUT

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

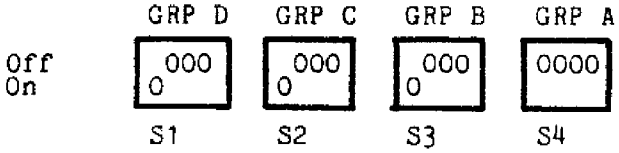
Before powering on the computer, set the 2016B switches as follows:



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,

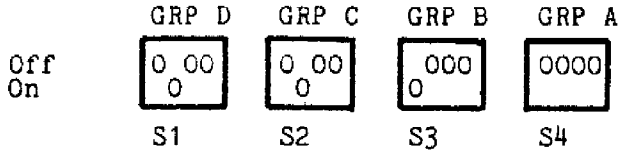
isolate the malfunction with a logic probe or voltmeter before continuing.

After Group A has been checked, power down the computer and set the switches of groups B, C, and D to 1h.



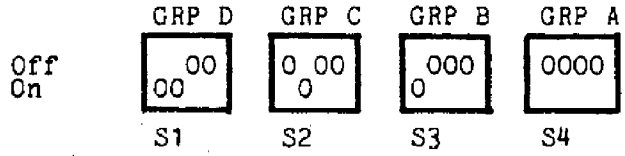
Group B will be selected. Examine 1000H (A12 on, the rest off), and deposit the same data bytes as was done with Group A. Isolate and correct any malfunctions as they become apparent.

To check Group C, power down the computer and set the switches of groups C and D to 2h.



Examine 2000H (A13 on, the rest off), and test as with Groups A and B.

Finally, to test Group D, power down and set the switches of group D to 3h.



Examine 3000H (A12 and A13 on, the rest off), and test as before. When all malfunctions have been corrected, proceed to the next test.

2.2 DIAGNOSTIC TEST OVERVIEW

These memory diagnostics run on 8080 or Z-80 systems and provide a practical test of the 2016B 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 the fault detection and provide sufficient data for fault isolation. After a fault is isolated, correction is a practical 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.

2.3 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. This routine 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).

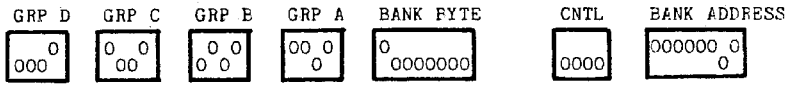
CONOUT: Console output. This routine writes one ASCII character to the console display device. The character to be output is passed to CONOUT in the C register. If the console output device is sensitive to bit 7, then the user must set/reset bit 7 to what is needed in the CONOUT routine.

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.

2.4 SETTING UP FOR THE TEST

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



At this point you are ready to install the 2016B in the computer. Make sure that no other memory will respond to addresses in the range 4000H-0BFFFH.

2.5 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.

2.6 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,

ERROR CONDITION	PROBABLE CAUSE	SUSPECT PARTS
Bad data=0FFH, all groups	a) bank select b) board select	U10, U11 U6-7, U14, U57-60
Random data or all 0 data, all groups	bad write control	U56-57, U60-61
0FFH data, one group only	a) group A select b) group B select c) group C select d) group D select	U5, U6, U7, U9 U4, U6, U7, U9 U3, U6, U7, U8 U2, U6, U7, U8
One address line hung (printout: good data, bad address)	address buffers	U53 (A0,1,4,5) U54 (A2,3,6,7,12,15) U55 (A8-11,13,14)
One data line hung a) hung 0 (good address, bad data=0)	grounded data line	U56, U57, U58
b) hung 1 (good address, bad data=1)	a) open data line b) data line shorted to +5V	U56, U57, U58 U56, U57, U58, memory chips
Soft errors (random addresses and data, non-repeatable)	a) memory chip access time b) heat-sensitive parts	Try closing Wait switch and rerunning tests. Treat as a hard error and replace suspect parts.
Hard memory errors	bad memory chip	See earlier table to identify chip.

When all walking bit tests run error-free, type in C for the continuous burn-in test. Specify a block size of 4000H and the appropriate base address (8000H 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.

2.7 ERROR PRINTOUT INTERPRETATION

Errors may show up in many forms. The table 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.

```
                        TITLE   '2114 MEMORY DIAGNOSTIC VER 1.1'
                ;
                ;
                ; Console input/output support routines
                ;
                ; These routines are a highly matured, well thought
                ; out set based on Intel's monitor.  They provide a
                ; significant capabillity to converse with an 8080,
                ; 8085, or Z-80 based microprocessor system.  The
                ; only registers altered are the accumulator and
                ; the pass register carrying active parameters upon
                ; entry to routine.  The stack is used extensively;
                ; sufficient stack space must be provided by the
                ; calling programs.  The stack pointer is returned
                ; pointing to same place on exit unless an error
                ; was detected (SP=?) or parameters are returned on
                ; the stack.  In the last case, the stack is offset
                ; by 2 times the requested number of parameters and
                ; will be set right after popping these parameters
                ; off the stack.
                ;
                ; Register usage conforms to ICOM and CP/M defined
                ; conventions:  Output data is passed in the C
                ; register and input data is expected in the A
                ; register.  These routines require CP/M compatable
                ; CONIN and CONOUT routines as contained in the
                ; user`s BIOS program; or CI and CO as in the ICOM
                ; Resident PROM.
                ;
000A    LF      EQU     0AH     ; ASCII line feed
000D    CR      EQU     0DH     ; ASCII carriage return
0040    CNTL    EQU     40H     ; ASCII Cntl offset
0040    STACK   EQU     40H
                ;
                ;
                ;
0040            ORG     40H
                ;
C38F03          JMP     INIT
                ;
0100            ORG     0100H
                ;
                ; SYSTEM LINKAGES
                ;
C003    CONIN   EQU     0C003H
C006    CONOUT  EQU     0C006H
C373    CONST   EQU     0C373H
C000    USER    EQU     0C000H
                ;
C38F03          JMP     INIT
C303C0  CONI:   JMP     CONIN
C306C0  CONO:   JMP     CONOUT
C373C3  CST:    JMP     CONST
C300C0  ERR:    JMP     USER
                ;
```

2.8 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:
```

```
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             ; program and then echos it back. It is assumed the
130 0142             ; 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 if it is a valid delimiter (' ', ',',
164 0157             ; or carriage return.)  If valid, it returns a zero
165 0157             ; indication in the status flags.   Further, if it
```

```
 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
 68 0115             ; its ASCII equivelant.  The high order 4
 69 0115             ; accumulator 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 lefthand
 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    PRTWA   ; 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
```

increasing the base address in 1K (4000H) increments, until base address 7C00H has been tested. This tests all memory chips.

BASE ADDRESS	CHIPS TESTED	MEMORY GROUP
4000H	U30, U47	A
4400H	U31, U48	A
4800H	U32, U49	A
4C00H	U33, U50	A
5000H	U26, U43	B
5400H	U27, U44	B
5800H	U28, U45	B
5C00H	U29, U46	B
6000H	U22, U39	C
6400H	U23, U40	C
6800H	U24, U41	C
6C00H	U25, U42	C
7000H	U18, U35	D
7400H	U20, U36	D
7800H	U21, U37	D
7C00H	U22, U38	D

If errors are logged, replace the appropriate chip(s). The above table narrows any error to two chips. If the bada data is in the upper half of the byte, replace the lower-numbered chip (physically higher on the board). If the bad data is in the lower half of the byte, replace the higher-numbered chip. For example, the following error printout indicates chip 29 bad:

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

After a good run for all sixteen 1K increments, run the walking bit test with a block size of 16k (4000H).

At this point, invert the memory group address switches and run a 16K block starting at 8000H. This tests the group-select circuitry completely. The primary chips tested here are U2-U6.

```
166 0157                ; is a carriage return, then the carry bit is set.
167 0157                ; If it is not a delimiter; a non-zero, no carry
168 0157                ; indication is required.
169 0157                ;
170 0157                ; Entry parameters:     None
171 0157                ; Exit Parameters:      See description above.
172 0157                ; Stack usage:          6 bytes
173 0157                ;
174 0157 CD4201         PCHK:   CALL    ECHO    ;Read a character
175 015A                ; Alternate entry point if CHAR already in (A)
176 015A FE20           PCH2:   CPI     ' '     ; Check for a blank
177 015C C8                     RZ              ; Return if (SO)
178 015D FE2C                   CPI     ','     ; Check for a comma
179 015F C8                     RZ              ; Return if (SO)
180 0160 FE0D                   CPI     'M'-CNTL
181 0162                                        ; Check for a CAR RET
182 0162 37                     STC             ; Set the carry flag
183 0163 C8                     RZ              ; Return if CAR RET
184 0164 3F                     CMC             ; Reset the carry flag
185 0165 C9                     RET
186 0166                ;
187 0166                ; Routine PRM reads characters from the console and
188 0166                ; pushes them onto the stack.  Multiple parameters
189 0166                ; may be read:  values are delimited by a ' ' or a
190 0166                ; ','.  If a carriage return is entered, it stops
191 0166                ; reading values and returns to the caller.  Only
192 0166                ; the last 4 characters of a string are saved:  An
193 0166                ; error may be corrected by typing more characters
194 0166                ; until the last 4 are correct.  The caller may
195 0166                ; retrieve the values by popping them off the stack
196 0166                ; in reverse order to which they were entered.
197 0166                ;
198 0166                ; Entry parameter:      (C) = number of expected
199 0166                ;                       parameters
200 0166                ; Exit parameters:      (C) Parameters on stack:
201 0166                ;                       If a bad value was entered,
202 0166                ;                       '????' is printed and
203 0166                ;                       control transferred to a
204 0166                ;                       user provided error handler.
205 0166                ;                       The stack pointer value is
206 0166                ;                       indeterminant and needs
207 0166                ;                       to be reset
208 0166                ; Stack usage:          4 + 2 = (C) bytes
209 0166                ;
210 0166                ; Alternate entry point if only one parameter is
211 0166                ; desired.
212 0166 0E01           PARM1:  MVI     C,1
213 0168                ; Normal entry point
214 0168 210000         PRM:    LXI     H,0     ; Set (HL) = 0
215 016B CD4201         PRA:    CALL    ECHO    ; Get a character
216 016E 47             PRB:    MOV     B,A     ; Save input character
217 016F CD9901                 CALL    NIBBL   ; Check it and CVB
218 0172 DA7E01                 JC      PRC     ; Not hex, see if delim
219 0175 29                     DAD     H       ; Multiply (HL) by 16
220 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) reguster and verifies it is
247 0199             ; a valid hex digit.  If so, the binary value is
248 0199             ; returned in the lower half of the A register: the
249 0199             ; upper half is set to zero.  If not, the carry
250 0199             ; flag is set and return control 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     '0'-'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, LF
270 01AB             ; may be printed first.  Three forms of message end
271 01AB             ; delimiters are accepted: Bit 7 = 1 in last
272 01AB             ; character to be output: ASCII ETX (CONTROL 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 C5                         POP     B       ; Restore (BC)
01B5 C9                         RET
01B6                    ;
                        ; 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
01C4                    ;
                        ; Routine QPRT prints '????', and transfers control
                        ; to the user's error recovery routine.  (SP) is
                        ; indeterminant on exit.
01C4                    ;
01C4 21CD01             QPRT:   LXI     H,QMSG  ; Message address
01C7 CDAE01                     CALL    PRTWA   ; Print it
01CA C30C01                     JMP     ERR     ; Go to error recovery
01CD                    ;
01CD 3F3F3FBF           QMSG:   DB      '???','?'+80H
01D1                    ;
                        ; Hardware diagnostics can be divided into 3
                        ; categories:
                        ;       1) Fault detection
                        ;       2) Faul isolation
                        ;       3) Fault correction
                        ; These automate the first category only.  See the
                        ; user's manual for guidelines to the second
                        ; category.  After the second step is done the
                        ; fault correction should be obvious.
                        ;
                        ;
                        ; SUBROUTINES FOR THE MEMORY DIAGNOSTICS
                        ;
                        ; When a bad memory cell is detected, this routine
```

```
436 026E E1                 POP     H       ; Get remaing 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
455 0299            ; must have the block size and (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            ;
```

```
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 5A4453AA0
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
394 023E           ; both the data and address lines of a 2114 pair at
395 023E           ; the same time. First, it zeros all cells in the
396 023E           ; specified block, then insures they are all zero.
397 023E           ; It tests each 1K section separately.  Detected
398 023E           ; errors are logged on the console as they occur.
399 023E           ;
400 023E           ; The base address, when asked for, must be on 1K
401 023E           ; boundaries or it will be rejected and another
402 023E           ; address asked for.
403 023E           ;
404 023E           ; The operator can abort the test at any time by
405 023E           ; typing a ETX (Cntl C) should too many errors be
406 023E           ; detected.  Allowing the test to complete will
407 023E           ; insure adequate data to perform a thorough fault
408 023E           ; isolation.
409 023E           ;
410 023E           ; Without errors, this diagnostic tests a 1K cell
411 023E           ; in 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       ; Retreive BASE ADDRESS
416 0245 D1                POP     D       ; Retreive 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
```

```
540 02FC 79              MOV     A,C     ; Get data into (A)
541 02FD 07              RLC             ; Shift for next pattern
542 02FE 4F              MOV     C,A
543 02FF D2C002          JNC     WLKC    ; Not done yet
544 0302 C1              POP     B       ; Get base address
545 0303 D1              POP     D       ; Get block size
546 0304 3600            MVI     M,0     ; Reset test cell
547 0306 7D              MOV     A,L     ; Strip off base
548 0307 91              SUB     C       ; address
549 0308 6F              MOV     L,A
550 0309 7C              MOV     A,H
551 030A 98              SBB     B
552 030B 67              MOV     H,A
553 030C 29              DAD     H       ; Go to next address bit
554 030D CD1703          CALL    CHLDE   ; See if done
555 0310 F0              RP              ; Yes, return
556 0311 09              DAD     B       ; Build next address
557 0312 D5              PUSH    D       ; Save block size
558 0313 C5              PUSH    B       ; Save base address
559 0314 C3BE02          JMP     WLKDA   ; Go do it again
560 0317            ;
561 0317            ; Compare (HL) register to (DE) register and set
562 0317            ; flags on result.
563 0317            ;
564 0317 7C         CHLDE:  MOV     A,H
565 0318 92             SUB     D
566 0319 C0             RNZ
567 031A 7D             MOV     A,L
568 031B 93             SUB     E
569 031C C9             RET
570 031D            ;
571 031D            ; Routine BRNIN continuously writes a sequence of
572 031D            ; non zero numbers into a specified memory block
573 031D            ; and reads them back for comparing.  If errors
574 031D            ; occur, they are logged on the console.  A running
575 031D            ; error total number is also maintained. The test
576 031D            ; may be terminated at any time by typing a ETX
577 031D            ; (Cntl C).  At this time the running error total
578 031D            ; is displayed on the console.  The test data
579 031D            ; sequences are from 1 to 255 decimal, and then
580 031D            ; repeats itself but always skiping 0.
581 031D            ;
582 031D 217703     BRNIN:  LXI     H,CBMSG ; Get message address
583 0320 CD0C02         CALL    PARM    ; Write it, get parameters
584 0323 E1             POP     H       ; Get base address
585 0324 D1             POP     D       ; Get block size
586 0325 0E01           MVI     C,1     ; Seed the data
587 0327 0600           MVI     B,0     ; Initialize error count
588 0329 C5         BRNA:   PUSH    B       ; Save data, error count
589 032A D5             PUSH    D       ; Save block size
590 032B E5             PUSH    H       ; Save base address
591 032C 71         BRNB:   MOV     M,C     ; Write the data byte
592 032D 0C             INR     C       ; Advance data patern
593 032E C23203         JNZ     BRNC    ; Skip 0
594 0331 0C             INR     C       ; Set to 1
```

```
485 02BB             ; Routine WLKAD walks a single high bit thru all
486 02BB             ; data and addresses in a controled manner.  After
487 02BB             ; writing data, all other locations are tested for
488 02BB             ; a zero.  When an error is detected, it is logged
489 02BB             ; as described above.  If excessive errors occur,
490 02BB             ; the test may be aborted by typing a ETX (Cntl 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       ; Retreive 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
```

```
 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 burnin
 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
```

```
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 count
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 address
630 036D C3AE01            JMP     PRTWA   ; Print it and return to EXEC
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           ; Routine INIT and EXEC initialize the computer and
636 0389           ; monitors the console for a command.  When a valid
637 0389           ; command is received, control is  transfered to
638 0389           ; the desired routine.
639 0389           ;
640 0389 219002    RETN:   LXI     H,TDMSG ; Print 'TEST DONE'
641 038C CDAB01            CALL    PRTWD
642 038F 314000    INIT:   LXI     SP,STACK        ; Set stack pointer
643 0392 21AC03    EXEC:   LXI     H,DIMSG ; Print diag message
644 0395 CDAB01            CALL    PRTWD
```

each chip through a common address bus. Column selection is handled by a pair of 3-to-8 decoders. Each decoder selects one of eight columns, depending on the conditions of inputs A, B, and C. Inputs G1, G2A, and G2B determine whether a decoder will be enabled.

Decoder enabling and group selection are performed by the Memory Group Priority Encoder. Address bits A12-A15 are compared with the user-selected four-bit addresses of each of the four memory groups. -A12 through -A15 are parallelled into four quad open collector exclusive-OR gates. Each gate compares -A12, -A13, -A14, or -A15 with the corresponding bit of the memory group address. The output of each exclusive-OR gate in a memory group must be high for the memory group to be selected; one low output will pull the open collector output from that group low. The open collected outputs are inverted to form the -SEL lines A-D. These lines are the Priority Encoder variable inputs; inputs 0-3 are held permanently high. The following table shows the A, B, and C outputs for all possible inputs.

-SEL LINE				OUTPUT		
A	B	C	D	A	B	C
H	H	H	H	H	H	H
L	X	X	X	L	L	L
H	L	X	X	L	L	H
H	H	L	X	L	H	L
H	H	H	L	L	H	H

If the Priority Encoder is enabled and at least one input is low, the Enable Output (-EO) will be low, causing the Board Select LED to light.

The Memory Section Decoders are enabled by lows at G2A and G2B and a high at G1. Input C determines which group the enabled column will be in, while inputs A and B determine which column in a group will be enabled. For example, if group A is addressed (-SELA low), U8 is disabled by a low at G1, U9 is enabled by lows at G2A and G2B, and the high at U9's C input determines that column 4, 5, 6, or

CHAPTER 3

THEORY OF OPERATION

This chapter is intended for those users who want a more thorough understanding of the 2016B than they need to make the board function in their systems. Used in conjunction with the logic diagram in Chapter 4, it should provide 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.

3.1 MEMORY

The 2016B uses 2114-type RAMs, which are fully static (i.e., they require no clock or refresh signals) and provide 4096 bits of storage organized 1024 x 4. Each RAM thus requires ten address inputs and four bi-directional data lines. A Chip Select input (-CS) provides for the addressing of individual chips in a 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 2016B controls -CS through the Memory Section decoders; R/W goes low when either -pWR or MWRITE is active.

3.2 MEMORY ADDRESSING

Addressing a specific memory location on the 2016B involves addressing a location on each chip while enabling only one two-chip column. Address lines A0-A15 enter the board and are inverted, A0-A9 addressing one location on

must be closed. If the 2016B bank is activated, the high on BANK ACTIVE, ex-ORed with a permanent high signal, lights the Bank Active LED. If BANK ACTIVE is low, the priority encoder is disabled, thereby disabling the board.

The flip-flop will be reclocked each cycle in which the bank port address is received, pDBIN is high, and -pWR pulses low. Between clockings the BANK ACTIVE line will maintain the state determined by the bank byte circuitry.

3.4 BANK-INDEPENDENCE

Closing the Bank Disable (BD) switch pulls the outputs of the Bank Byte Select switches low, causing BANK ACTIVE to be permanently high. This enables the Priority Encoder independent of the bank byte circuitry, causing the board to be enabled whenever one of its memory groups is addressed.

3.5 DATA BUFFERS

The DI and DO 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. DO0-7 are disabled unless either -pWR or MWRITE is active (-IN ENABLE high). If -IN ENABLE is low the buffer allows data to be written to the RAMs.

Read-enabling is more involved. Basically, if the Phantom jumper is not installed DI0-7 will be enabled whenever a memory group on the board is addressed (MEMORY BOARD ACTIVE high) and the processor is in a memory read cycle (-IN ENABLE and pDBIN high). If the Phantom jumper is installed, a low on -PHANTOM will disable DI0-7 by pulling MEMORY BOARD ACTIVE low. -PHANTOM is generated by another device in the system and allows that device to overlay identically-addressed memory locations on the 2016B board by preventing 2016B data from reaching the data bus. Thus data is read from the overlaying device only.

7 will be enabled. Address lines A10 and A11 are the A and B inputs to each decoder, determining which of the four columns in an addressed group will be enabled.

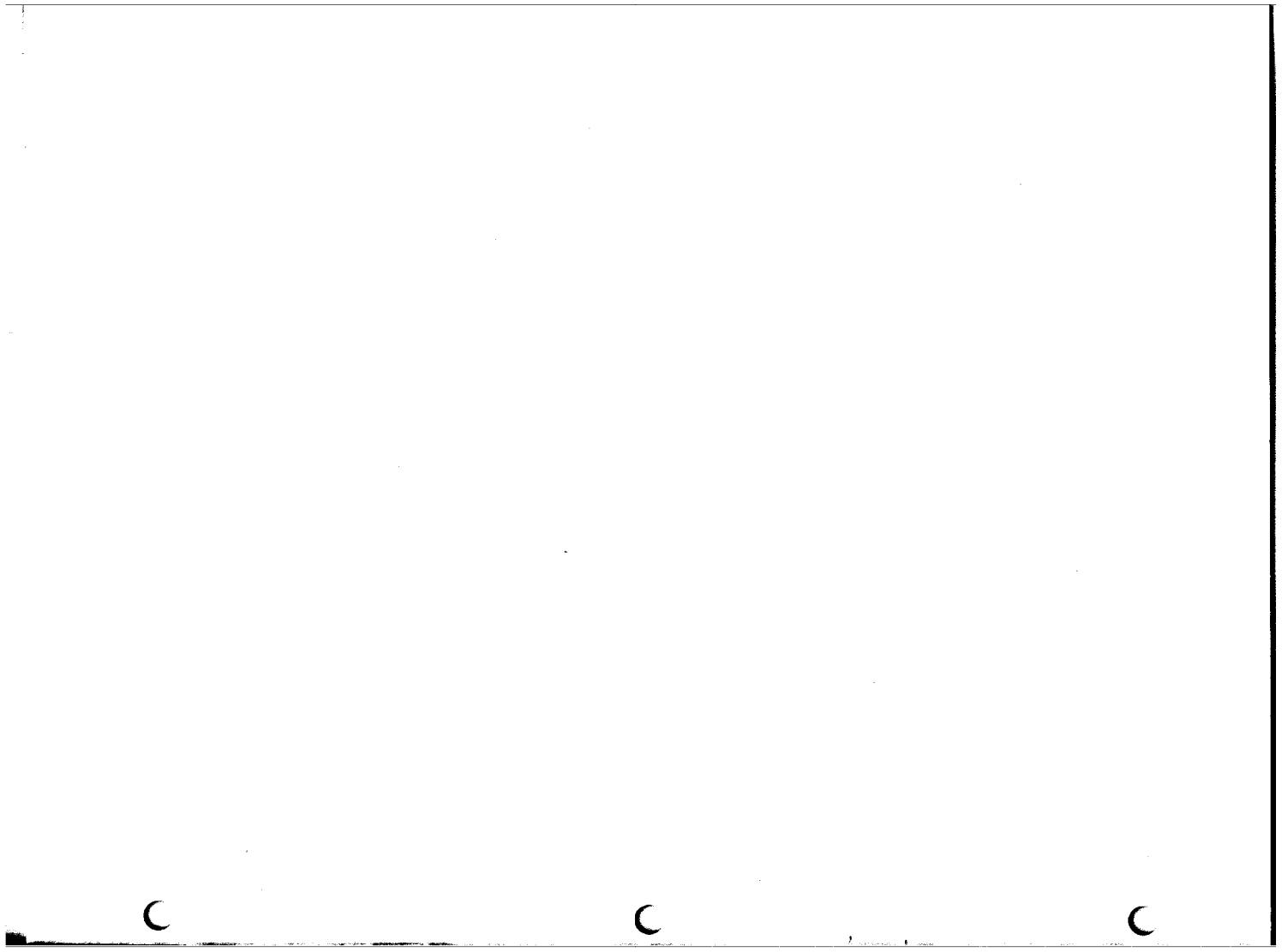
Notice that if Group A is addressed, the Priority Encoder does not care whether or not Groups B-D have been addressed, and so on. Thus the memory groups are fully prioritized, with A highest and D lowest. If two memory groups are given the same address, only the higher-priority group will be enabled by that address. The other groups will effectively be buried; they will be unaddressable and will occupy no memory space.

3.3 BANK SELECTION

The CCS 2016B is bank-selectable by bank port address and bank byte. Thus it is fully compatible with Cromemco, Alpha Micro, and other port-bank-select systems. IT IS NOT COMPATIBLE WITH ADDRESS-SELECT SYSTEMS SUCH AS IMSAI.

You assign the 2016B to the bank you want by setting the bank port address and bank byte switches. To select a specific bank during operation you address the bank port through the low order byte on the address bus and put the bank byte on the data bus. The 2016C compares -A0 through -A7 with the switch-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-address-comparison line is ANDed with pDBIN, and the resulting output is NANDed with inverted -pWR to form the BANK CLOCK line. This line clocks the D-type positive-edge-triggered flip-flops that make up the Bank Byte Register.

The bank address and pDBIN lines go high first. As long as -pWR is inactive (high, inverted low) the BANK CLOCK line is low. When -pWR goes active (low, inverted high) the BANK CLOCK line goes high, clocking the flip-flops. In the meantime the bank byte is written onto the data bus. The D inputs to the flip-flops are the data lines D0-D7; the -Q outputs are thus the bank byte inverted. The -Q lines for which the Bank Byte Select switches are closed enter an eight-input NAND gate. The output of this gate is the BANK ACTIVE line. For this line to go high, a low signal, inverted from a high bank byte bit, must cross a closed Bank Byte Select switch. In other words, for the 2016B to be bank-enabled, at least one bit of the bank byte must be high, and at least one corresponding Bank Byte Select switch



3.6 WAIT STATES

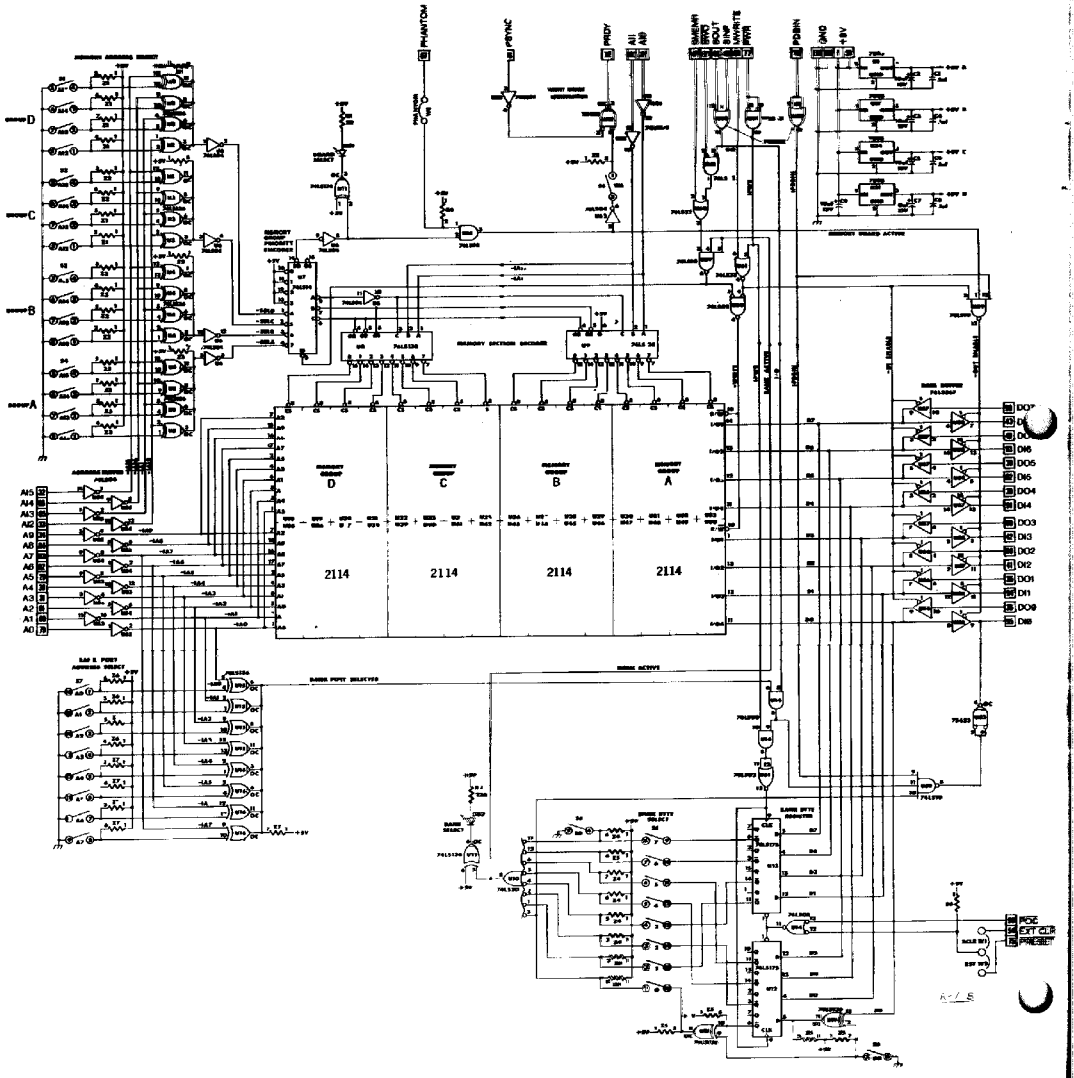
A Wait state is necessary when a peripheral device takes more time to complete a task than the processor normally allows. Because the 2016B is available with 200, 300, or 450 nsec Rams, and because processor speeds vary, the Wait feature on the 2016B has been made switch-enabled. If the Wait switch is closed, pRDY will be pulled low whenever pSYNC goes high and the board is selected (MEMORY BOARD ACTIVE high). This causes an extra clock cycle to be added 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 switch is open the 2016B does not pull pRDY low and a Wait state does not occur unless it originates elsewhere.

3.7 REGISTER CLEARING

A low to pin 1 of the Bank Byte Register flip-flops clears them. Q goes high and -Q goes low, as if a 00h bank byte had been clocked in. If the Bank 0 (B0) switch is closed, however, the -Q output for bit 0 will be low when the flip-flops are cleared. Thus, if the bit 0 switch of Bank Byte Select is also closed, the board will come up active when the register is cleared. If the Bank 0 switch is open, bit 0 functions the same as the other bits, though the signal is inverted twice along the way.

The flip-flops will always be cleared when the system's power is turned on (-POC active). You have the option of adding another clear input to the flip-flops. If jumper W1 is installed, the peripheral-resetting line -EXT CLR will clear the bank byte register. If jumper W2 is installed, -pRESET, which is low when the CPU is reset, will clear the register. Only one of these jumpers may be installed.

4.1 SCHEMATIC/LOGIC DIAGRAM

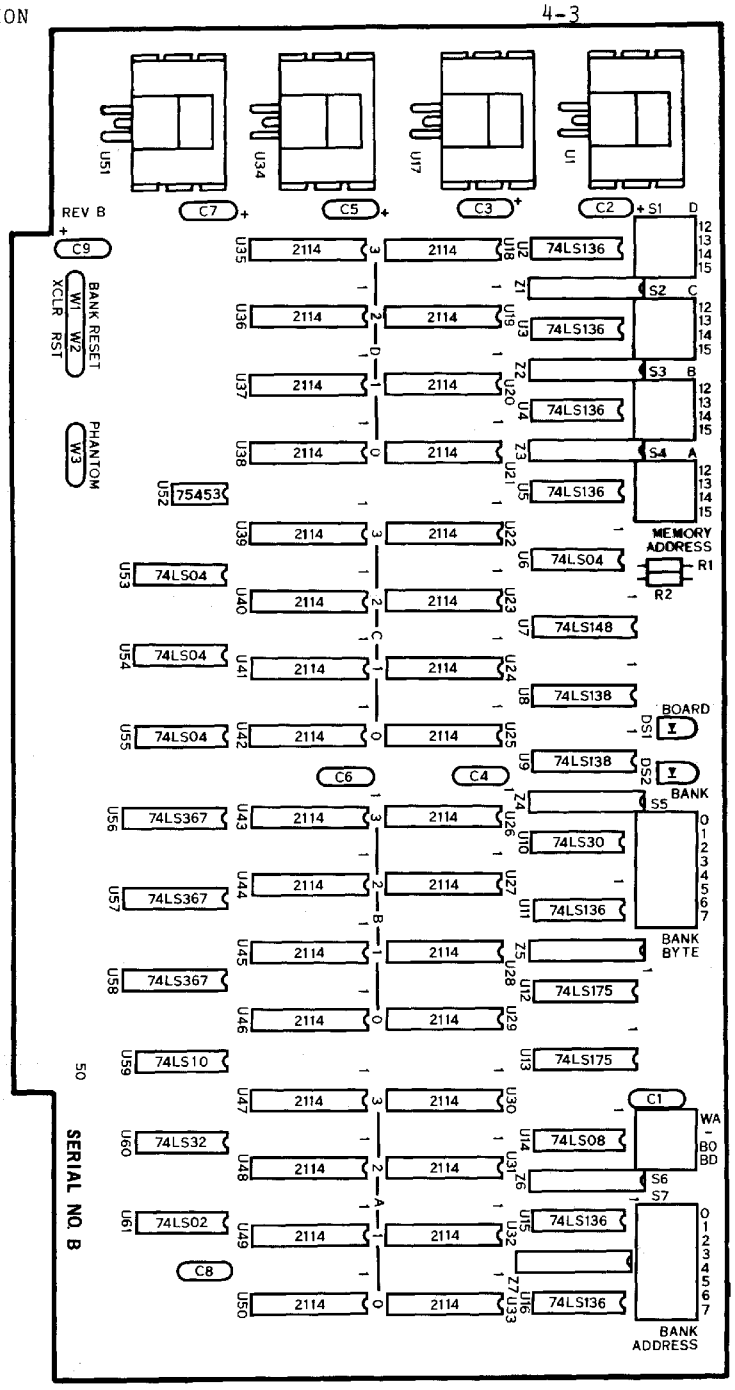


CHAPTER 4

TECHNICAL INFORMATION

```
                        ; is called to print the bad address, bad data,
                        ; test address, and test data (in that order).
                        ; From this error log, the fault isolation process
                        ; can be conducted.
                        ;
01D1 CD2B01     ADPRT:  CALL    DEPRT   ; Print bad address
01D4 CD0F01             CALL    BLK     ; Print a blank
01D7 78                 MOV     A,B     ; Get a bad data
01D8 C3E001             JMP     ADPRB
                        ;
                        ; Alternate entry point when bad address is
                        ; meaningless
01DB F5         ADPRA:  PUSH    PSW
01DC CD1E01             CALL    CRLF    ; Do a (CR,LF)
01DF F1                 POP     PSW
01E0 CD3301     ADPRB:  CALL    HEX2    ; Print bad data
01E3 CD0F01             CALL    BLK
01E6 CD0F01             CALL    BLK
01E9 CD5101             CALL    HLPRA   ; Print test address
01EC CD0F01             CALL    BLK
01EF 79                 MOV     A,C     ; Get test data
01F0 C33301             JMP     HEX2    ; Print it
                        ;
                        ; Routine BREAK tests the console status to see if
                        ; a character has been typed in.  If so, it checks
                        ; to see if it is an ASCII ETX (Cntl C). If both
                        ; tests are met, it types an 'ABORT' message and
                        ; returns control to the calling routine.
                        ;
01F3 CD0901     BREAK:  CALL    CST     ; Character waiting?
01F6 C8                 RZ              ; No, return
01F7 CD0301             CALL    CONI    ; Yes, get it
01FA FE03               CPI     'C'-CNTL
                                        ; See if Cntl C
01FC C0                 RNZ             ; No, return
01FD 210702             LXI     H,ABMSG ; Print out the
0200 CDAB01             CALL    PRTWD   ; 'ABORT' message
0203 313E00             LXI     SP,STACK-2
                                        ; Reset the stack
0206 C9                 RET             ; Return to exec
                        ;
0207 41424F52   ABMSG:  DB      'ABOR','T'+80H
    020B D4
                        ;
                        ; Routine PARM reads in the desired test block size
                        ; and block base address.  Both parameters are
                        ; pushed on the stack.
                        ;
020C CDAE01     PARM:   CALL    PRTWA   ; Print caller's name
020F 212402             LXI     H,BZMSG ; Print BLOCK SIZE message
0212 CDAB01             CALL    PRTWD
0215 CD6601             CALL    PARM1   ; Get block size
0218 E1                 POP     H       ; Retrieve it
0219 E3                 XTHL
021A E5                 PUSH    H       ; Save return address
```

4.2 ASSEMBLY COMPONENT LAYOUT

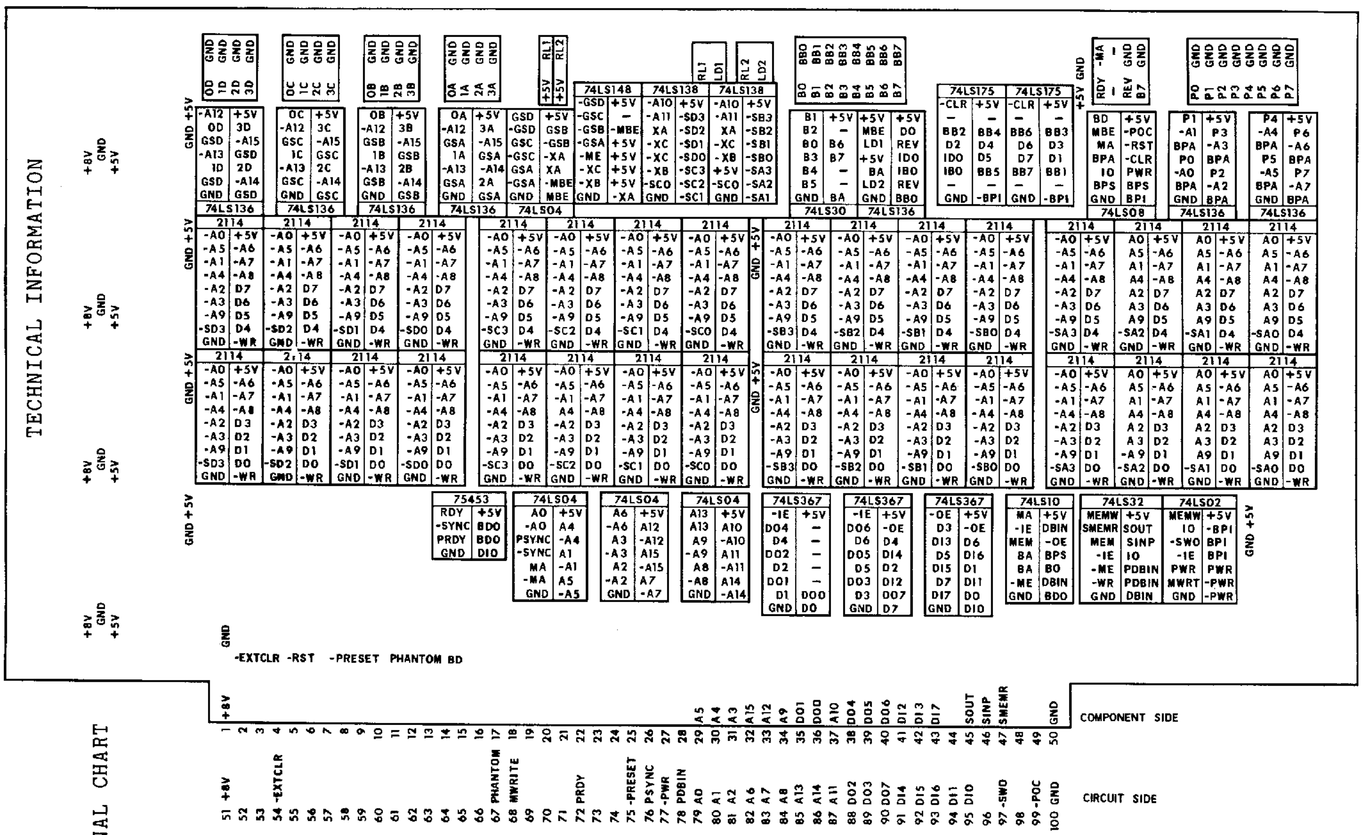


4.3 PARTS LIST

QTY	REF	DESCRIPTION	CCS PART #
CAPACITORS			
5	C2,3,5,7,9	Tantalum, 4.7uf, 35 vdc, 20%	42804-54756
4	C1,4,6,8	Ceramic, .1uf, 50 vdc, 20%	42142-21046
RESISTORS			
2	R1,2	220 ohm, 1/4 w, 5%	40002-02215
1	Z1	Resistor Network, SIP 2.7K ohm x 7	40930-72726
INTEGRATED CIRCUITS			
32	U18-33,35-50	MOS 2114 1Kx4 Static RAMS	31900-21142 (200nsec) or -21143 (300nsec) or -21144 (450nsec)
4	U1,17,34,51	7805 +5v regulator	32000-07805
7	U2-5,11,15,16	74LS136 quad ex-OR:OC	30000-00136
3	U6,53-55	74LS04 hex inverter	30000-00004
2	U8,9	74LS138 octal decoder	30000-00138
1	U10	74LS30 8-in NAND	30000-00030
2	U12,13	74LS175 quad D register	30000-00175
1	U14	74LS08 quad 2-in AND	30000-00008
1	U52	75453 dual 2-in OR: OC	30300-00453
3	U56-58	74LS367 hex bus driver	30000-00367
1	U59	74LS10 tri 3-in NAND	30000-00010

QTY	REF	DESCRIPTION	CCS PART #
1	U60	74LS32 quad 2-in OR	30000-00032
1	U61	74LS02 quad 2-in NOR	30000-00002
IC SOCKETS			
1	XU52	IC Socket, 8 PIN	58102-00080
16	XU2-6,10,11, 14-16,53-55, 59-61	IC Socket, 14 PIN	58102-00140
8	XU7-9,12,13, 56-58	IC Socket, 16 PIN	58102-00160
32	XU18-33,35-50	IC Socket, 18 PIN	58102-00180
MISCELLANEOUS			
5	S1-4,6	DIP Switch, 4pst	27111-41010
2	S5,7	DIP Switch, 8pst	27111-81010
2	CR1,CR2	Diode, Light Emitting	37400-00001
4		Heatsink, to 220	60022-00001
4		Nut, hex, 6-32 & lock washer (KEPS)	73006-32001
4		Screw, Phillips head (SIMS), 6-32x3/8	71006-32061
1		PC Board	02016-00002
1		Owner's Manual	89000-02016

4.4 SIGNAL CHART



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.