Owner's Manual

Model 2116

16K Static RAM Module

California
Computer
Systems

CCS MODEL 2116
16K STATIC RAM MODULE
OWNER'S MANUAL

CALIFORNIA COMPUTER SYSTEMS
250 CARIBBEAN DRIVE
SUNNYVALE CA 94086

MANUAL NO. 89000-02116

TABLE OF CONTENTS

FEATURES

Uses Popular 2114 Static RAMs

Available with 200, 300, or 450 nsec RAMs

Berg Jumpers Used for Selectable Features

4K Memory Blocks Individually Addressable to Any 4K Boundary

Bank Selection by Bank Port and Bank Byte

4K Blocks Individually Bank-Enabled

LEDs Indicate Board Active and Bank Active States

Wait State Jumper

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

CHAPTER 1

SETTING THE 2116 JUMPERS

The CCS 2116 is a 16K-byte static RAM board designed for use on the S-100 bus. 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 bank-enabled, 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 2116 with selectable addressing and several optional features. Selections are hard-wired with reliable, easy-to-use Berg jumpers. The addresses for each of the 4K memory groups, the bank port address and bank byte, and the bank-dependence or -independence of each memory group are jumper-set 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 3, "Theory of Operation."

1.1 SETTING THE MEMORY GROUP ADDRESSES

In order to provide maximum flexibility in the location of the 2116's memory groups within a bank, CCS has made the addresses of the four memory groups jumper-selectable. The jumper-set address for a memory group is compared with the high-order address lines A12-A15, and if the address matches, the memory group will be selected. Set the jumpers 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 jumpers to 0100. 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 jumpers are reset. This allows you to configure the 2116 to 4, 8, or 12K without removing RAMs.

1.2 SETTING THE BANK BYTE

The bank-byte jumpers allow you to hardware-map the 2116 memory board to whichever of the eight memory bank levels 0-7 you choose. 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. 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 jumpers corresponding to each desired bank to 1.

1.3 SETTING THE BANK PORT ADDRESS

In order to assign the 2116 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. Remember that A7 is the high-order bit; thus 40h is selected by setting jumper A6 to 1 and jumpers A0-A5 and A7 to 0.

1.4 SETTING MEMORY GROUP BANK-INDEPENDENCE

Each of the memory groups 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. To make a memory group independent, set its bank-dependence jumper to ME (Memory Enable). To make it bank-dependent, set the jumper to BE (Bank Enable).

1.5 SETTING THE BANK RESET JUMPER

If the Bank Reset jumper is set to B, all 16K 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. Bank-independent memory groups will be enabled with each reset no matter which position the Bank Reset jumper is set to.

1.6 SETTING THE PHANTOM JUMPER

Setting the Phantom jumper to ON allows a device that generates a -PHANTOM signal to overlay portions of the 2116 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 2116 board, the Phantom signal will block the output from the 2116 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 2116 would not be able to locate a memory group in that block because the 2116 and the peripheral control board would both put data on the bus when a shared location was addressed.

Setting the Phantom jumper to OFF disables the -PHANTOM line.

1.7 SETTING THE WAIT JUMPER

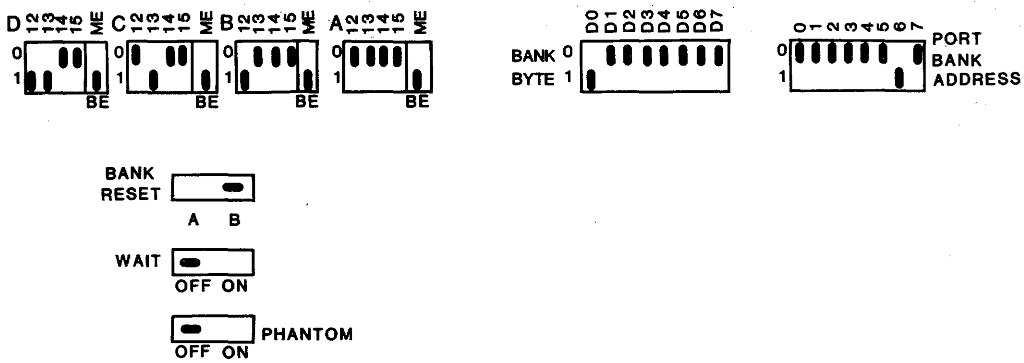
The Wait jumper allows you to slow down your processor every time the board is addressed. This will be necessary if your processor allows less memory access time than your RAMs require.

If you have a 2116 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.

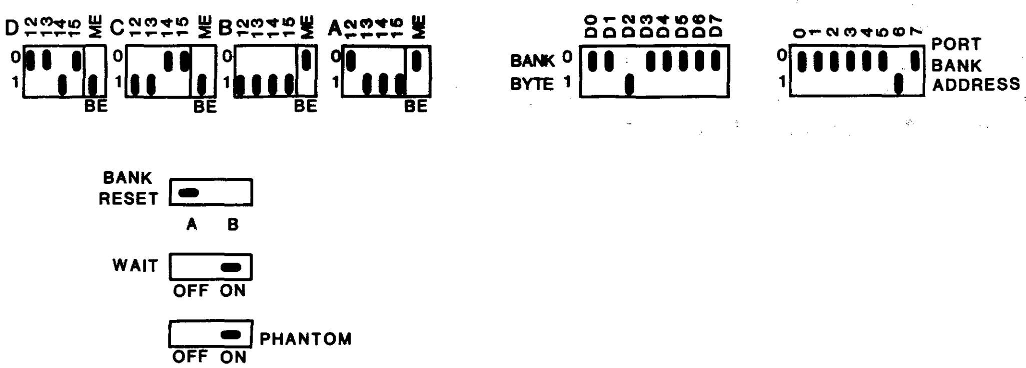
Some Z-80 CPU boards, including the CCS 2810, provide a jumper-selectable Wait feature. Enabling this feature may be preferable to enabling the 2116 Wait feature. The 2116 Wait causes a Wait state to occur in every memory cycle in which the board is addressed; the CCS CPU Wait feature causes a Wait state 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.

1.8 EXAMPLES OF JUMPER SELECTION

The first diagram shows jumper settings for a basic CCS system consisting of a 2810 Z-80 CPU, a 2422 disk controller, and the 2116. The bank port address must be 40h. The board is activated with bank 0 as well as on start-up and reset. Memory is located between 0 and 16K. Phantom and Wait are disabled.



In the last diagram memory groups A and B are bank-independent and located in the last 8K of memory. Groups C and D reside in bank 2 between 12K and 20K. The bank port address is 40h. Only groups A and B are enabled on start-up and reset. Phantom and Wait are enabled.



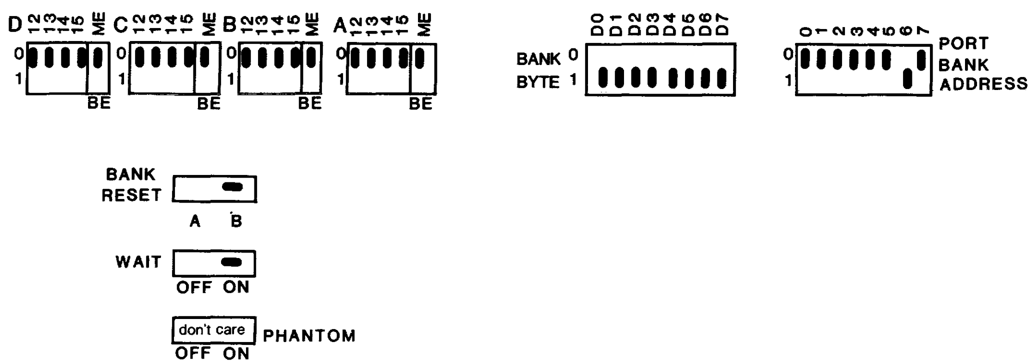
CHAPTER 2

TESTING AND TROUBLESHOOTING THE 2116

2.1 FRONT PANEL QUICK CHECKOUT

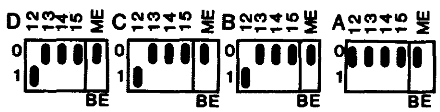
(If your computer does not have a front panel, skip this section.)

Before powering on the computer, set the 2116 jumpers as follows:



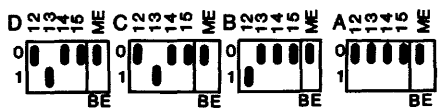
The priority feature will cause Group A to be selected. Set the Front Panel Adress Switches A0-A15 to the off position (0000H). Examine that address. Set the Data Switches D1-D7 to the off position and D0 to the on position (01H). Deposit (write) into memory and compare the Data readout with the switch 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 the jumpers of groups B, C, and D to 1h.



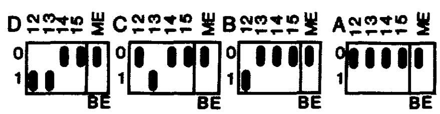
Group B will be selected. Examine 1000H (A12 on, the rest off), and deposit the same data bytes as was done with Group A. Isolate and correct any malfunctions as they become apparent.

To check Group C, power down the computer and set the jumpers 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 jumpers 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 2116 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:

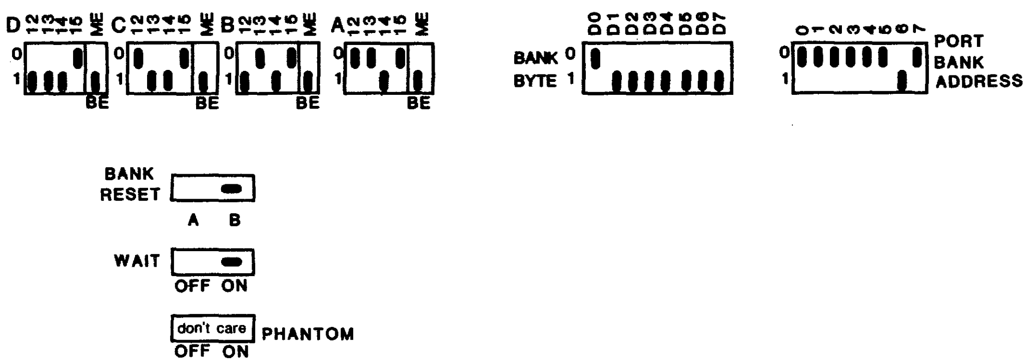
> CONIN: Console input. This routine reads one ASCII character from the console keyboard and sets the parity bit (bit 7) equal to 0. The character is returned in the accumulator (A register).
>
> CONOUT: Console output. This routine writes one ASCII character to the console display device. The character 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 (OFFH) 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 jumpers as illustrated:



At this point you are ready to put the 2116 into 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, 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	U18, U35	A
4400H	U19, U36	A
4800H	U20, U37	A
4C00H	U21, U38	A
5000H	U14, U31	B
5400H	U15, U32	B
5800H	U16, U33	B
5C00H	U17, U34	B
6000H	U26, U43	C
6400H	U27, U44	C
6800H	U28, U45	C
6C00H	U29, U46	C
7000H	U22, U39	D
7400H	U23, U40	D
7800H	U24, U41	D
7C00H	U25, U42	D

TABLE 2.1

If errors are logged, replace the appropriate chip(s). The above table narrows any error to two chips. If the bad 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 14 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 jumpers and run a 16K block starting at 8000H. This tests the group-select circuitry completely. The primary chips tested here are U2-U6.

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.

ERROR CONDITION	PROBABLE CAUSE	SUSPECT PARTS
Bad data=0FFH, all groups	a) bank select b) board select	U49, U56, U59 U6, U56-59
Random data or all 0 data, all groups	bad write control	U53-54, U57-58
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, U10 U2, U6, U7, U10
One address line hung (printout: good data, bad address)	address buffers	U50 (A0,1,4,5) U51 (A2,3,6,7,12,15) U52 (A8-11,13-14)
One data line hung a) hung 0 (good address, bad data=0)	grounded data line	U53, U54, U55
b) hung 1 (good address, bad data=1)	a) open data line b) data line shorted to +5V	U53, U54, U55 U53, U54, U55, memory chips
Soft errors (random addresses and data, non-repeatable)	a) memory chip access time b) heat-sensitive parts	Try setting Wait jumper ON and rerunning tests. Treat as a hard error and replace suspect parts.
Hard memory errors	bad memory chip	See TABLE 2.1 to identify chip.

TABLE 2.2

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:
```

```
0000                      TITLE   '2114 MEMORY DIAGNOSTIC VER 1.1'
0000             ;
0000             ;
0000             ; Console input/output support routines
0000             ;
0000             ; These routines are a highly-matured, well-thought-
0000             ; out  set based on Intel's monitor.  They provide a
0000             ; significant capability to converse with  an  8080,
0000             ; 8085,  or  Z-80  based microprocessor system.  The
0000             ; only registers altered are the accumulator and the
0000             ; pass  register  carrying  active  parameters  upon
0000             ; entry   to   a   routine.    The  stack  is  used
0000             ; extensively; sufficient space must be provided  by
0000             ; the  calling  programs.    The  stack  pointer  is
0000             ; returned to its original place on exit  unless  an
0000             ; error   was  detected  (SP=?)  or  parameters  are
0000             ; returned on the stack.  In the  latter  case,  the
0000             ; stack is offset by 2 times the requested number of
0000             ; parameters  and  will  be  set  right  after these
0000             ; parameters are popped off the stack.
0000             ;
0000             ; Register use conforms to  ICOM  and  CP/M  defined
0000             ; conventions:  Output  data  is  passed  in  the  C
0000             ; register and input  data  is  expected  in  the  A
0000             ; register.  These routines require CP/M-compatible
0000             ; CONIN and CONOUT  routines  as  contained  in  the
0000             ; user's  BIOS  program, or CI and CO as in the ICOM
0000             ; Resident ROM.
0000             ;
0000      000A   LF      EQU     0AH      ; ASCII line feed
0000      000D   CR      EQU     0DH      ; ASCII carriage return
0000      0040   CNTL    EQU     40H      ; ASCII Cntl offset
0000      0040   STACK   EQU     40H
0000             ;
0000             ;
0000             ;
0000      0040           ORG     40H
0040             ;
0040 C38F03              JMP     INIT
0043             ;
0043      0100           ORG     0100H
0100             ;
0100             ; SYSTEM LINKAGES
0100             ;
0100      C003   CONIN   EQU     0C003H
0100      C006   CONOUT  EQU     0C006H
0100      C373   CONST   EQU     0C373H
0100      C000   USER    EQU     0C000H
0100             ;
0100 C38F03              JMP     INIT
0103 C303C0      CONI:   JMP     CONIN
0106 C306C0      CONO:   JMP     CONOUT
0109 C373C3      CST:    JMP     CONST
010C C300C0      ERR:    JMP     USER
010F             ;
```

```
010F                ; Routine BLK prints one blank on the current
010F                ; console device.
010F                ;
010F                ; Entry parameters:      None
010F                ; Return parameters:     None
010F                ; Stack usage:           4 bytes
010F                ;
010F C5             BLK:    PUSH    B        ; Save (BC)
0110 0E20                   MVI     C,' '    ; Get an ASCII space
0112 C34901                 JMP     ECH2     ; Go output it
0115                ;
0115                ; Routine CONV converts a 4 bit binary number to its
0115                ; ASCII equivalent. The high-order 4 accumulator
0115                ; bits are lost.
0115                ;
0115                ; Entry parameter:       4 bit binary number in
0115                ;                        lower half of accumulator
0115                ; Exit parameter:        ASCII character in (A)
0115                ; Stack usage:           0 bytes
0115                ;
0115 E60F           CONV:   ANI     0FH      ; Clear high bits
0117 C690                   ADI     90H      ; Insert partial ASCII
0119 27                     DAA              ; Zone
011A CE40                   ACI     40H      ; Insert rest of ASCII
011C 27                     DAA              ; Zone
011D C9                     RET
011E                ;
011E                ; Routine CRLF prints an ASCII carriage return and
011E                ; line feed (in that order) on the console. It
011E                ; follows these with 4 blanks to create a left
011E                ; margin.
011E                ;
011E                ; Entry parameter:       None
011E                ; Exit parameter:        None
011E                ; Stack Usage:           8 bytes
011E                ;
011E E5             CRLF:   PUSH    H        ; Save (H,L)
011F 212701                 LXI     H,CRMSG  ; Get message address
0122 CDAE01                 CALL    PRTWA    ; Print message
0125 E1                     POP     H        ; Restore (HL)
0126 C9                     RET
0127                ;
0127 0D0A20A0       CRMSG:  DB      CR,LF,' ',' '+80H
012B                ;
012B                ; Routine DEPRT prints the contents of the (DE)
012B                ; register pair as a 4-digit hexadecimal number on
012B                ; the console.
012B                ;
012B                ; Entry parameter:       (DE) = 4 digit hex number
012B                ;                        to be printed on console.
012B                ; Exit parameter:        None
012B                ; Stack usage:           10 bytes
012B                ;
012B CD1E01         DEPRT:  CALL    CRLF     ; Print a CR, LF
012E                ; Alternate entry point if no CR, LF wanted
```

```
0012E 7A      DEPRA:  MOV     A,D      ; Get high order byte
012F CD3301           CALL    HEX2     ; Print 2 numbers
0132 7B               MOV     A,E      ; Get low order byte
0133                  ; Alternate entry point to print (A) as two hex
0133                  ; digits
0133 F5       HEX2:   PUSH    PSW      ; Save low order byte
0134 0F               RRC              ; Move high order nibble
0135 0F               RRC              ; to lower half of (A)
0136 0F               RRC
0137 0F               RRC
0138 CD3C01           CALL    HEX1     ; Print the nibble
013B F1               POP     PSW      ; Get low nibble back
013C                  ; Alternate entry point to print low order nibble
013C                  ; on console
013C CD1501   HEX1:   CALL    CONV     ; Convert to ASCII
013F C34501           JMP     ECH1     ; Go print it
0142                  ;
0142                  ; Routine ECHO reads one character from the  calling
0142                  ; routine  and  then  echoes it back.  It is assumed
0142                  ; that the console is in a full duplex mode.
0142                  ;
0142                  ; Entry parameter:      None
0142                  ; Exit parameter:       (A) = Character read from
0142                  ;                       the console keyboard
0142                  ; Stack usage:          4 bytes
0142                  ;
0142 CD0301   ECHO:   CALL    CONI     ; Read a character
0145                  ; Alternate entry point to print (A)
0145 C5       ECH1:   PUSH    B        ; Save (BC)
0146 E67F             ANI     7FH      ; Strip off parity bit
0148 4F               MOV     C,A      ; Put character into (C)
0149                  ; Alternate entry point for BLK routine
0149 CD0601   ECH2:   CALL    CONO     ; Output it
014C C1               POP     B        ; Restore (BC)
014D C9               RET
014E                  ;
014E                  ; Routine HLPRT prints  the  contents  of  the  (HL)
014E                  ; register as 4 hexadecimal digits on the console.
014E                  ;
014E                  ; Entry parameter:      (HL) = 4 hex digit number
014E                  ;                       to be printed
014E                  ; Exit parameter:       None
014E                  ; Stack usage:          10 bytes
014E                  ;
014E CD1E01   HLPRT:  CALL    CRLF     ; Print a (CR,LF)
0151                  ; Alternate entry point if no CR,LF wanted
0151 EB       HLPRA:  XCHG             ; Swap (HL), (DE)
0152 CD2E01           CALL    DEPRA    ; Go print (DE)
0155 EB               XCHG             ; Unswap (HL), (DE)
0156 C9               RET
0157                  ;
0157                  ; Routine  PCHK  reads  a character from the console
0157                  ; and checks whether it is a valid delimiter (space,
0157                  ; comma, or carriage return).  If  so,  a  zero  is
0157                  ; returned in the status flags.  If the character is
```

```
0157                ; a  carriage return, the carry bit is set also.  If
0157                ; it  is  not  a  delimiter,  a  non-zero,  no-carry
0157                ; indication is required.
0157                ;
0157                ; Entry parameters:     None
0157                ; Exit Parameters:      See description above.
0157                ; Stack usage:          6 bytes
0157                ;
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
```

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

```
01AB                ;
01AB                ; Entry Parameters:     (HL) = Message start address
01AB                ;                       (B) = ETX delimiter (See
01AB                ;                       description above.)
01AB                ; Exit Parameters:      None - (HL) is altered
01AB                ; Stack usage:          12 bytes MAX
01AB                ;
01AB                ; Entry point for CR,LF (will not work with user
01AB                ; defined ETX delimiter).
01AB CD1E01         PRTWD:  CALL    CRLF
01AE                ; Entry point for No. CR,LF and a bit 7 or ASCII
01AE                ; 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                ;
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                ;
01C4                ; Routine QPRT prints "????" and  transfers  control
01C4                ; to  the  user's  error- recovery routine.  (SP) is
01C4                ; indeterminate 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                ;
01D1                ; Hardware diagnostics can be divided into 3 stages:
01D1                ;         1) fault detection
01D1                ;         2) fault isolation
01D1                ;         3) fault correction
01D1                ; These routines automate the first stage only.  See
01D1                ; the user's manual for guidelines  for  the  second
01D1                ; stage.  After the second step is completed, fault
01D1                ; correction should be no trouble.
01D1                ;
01D1                ;
01D1                ;
01D1                ; SUBROUTINES FOR THE MEMORY DIAGNOSTICS
01D1                ;
01D1                ; When a bad memory cell is detected,  this  routine
```

```
01D1                ; is called to print the bad address, bad data, test
01D1                ; address, and test data (in that order).  With this
01D1                ; error  log,  the  fault  isolation  process can be
01D1                ; conducted.
01D1                ;
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
01DB                ;
01DB                ; Alternate entry point when bad address is
01DB                ; 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
01F3                ;
01F3                ; Routine BREAK tests the console status to see if a
01F3                ; character has been typed in.  If so, it checks  to
01F3                ; see  if  it  is an ASCII ETX (CNTRL C).  If so, it
01F3                ; types an "ABORT" message and  returns  control  to
01F3                ; the calling routine.
01F3                ;
01F3 CD0901         BREAK:  CALL    CST     ; Character waiting?
01F6 C8                     RZ              ; No, return
01F7 CD0301                 CALL    CONI    ; Yes, get it
01FA FE03                   CPI     'C'-CNTL
01FC                                        ; 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
0206                                        ; Reset the stack
0206 C9                     RET             ; Return to exec
0207                ;
0207 41424F52       ABMSG:  DB      'ABOR','T'+80H
020B D4
020C                ;
020C                ; Routine  PARM reads in the desired test block size
020C                ; and  block  base  address.   Both  parameters  are
020C                ; pushed onto the stack.
020C                ;
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
```

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

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

```
02BB                ; Routine WLKAD walks a single high bit through each
02BB                ; data bit of all addresses in a controlled  manner.
02BB                ; After  a  bit  is written, all other locations are
02BB                ; tested for zeros.  When an error is  detected,  it
02BB                ; is  logged  as  described above.  If excess errors
02BB                ; occur, abort the test by typing CNTRL C.
02BB                ;
02BB D5             WLKAD:  PUSH    D        ; Save block size
02BC E5                     PUSH    H        ; Save address
02BD 23                     INX     H        ; Set A0
02BE 0E11           WLKDA:  MVI     C,11H    ; Set D0, D4 (2114)
02C0 C5             WLKC:   PUSH    B        ; Save it
02C1 71                     MOV     M,C      ; Write byte into memory
02C2 E5                     PUSH    H        ; Save current address
02C3 33                     INX     SP       ; Adjust stack to
02C4 33                     INX     SP       ; find base address
02C5 33                     INX     SP
02C6 33                     INX     SP
02C7 E1                     POP     H        ; Retrieve base address
02C8 E5                     PUSH    H        ; Restore it
02C9 3B                     DCX     SP       ; Readjust stack
02CA 3B                     DCX     SP
02CB 3B                     DCX     SP
02CC 3B                     DCX     SP
02CD 7E             WLKB:   MOV     A,M      ; Read byte
02CE 47                     MOV     B,A      ; Save byte in (B)
02CF A7                     ANA     A        ; Test data
02D0 EB                     XCHG
02D1 E3                     XTHL             ; Get test address
02D2                                         ; Save loop control
02D2 C2DE02                 JNZ     DNZT     ; Non-zero data, jump
02D5 CD1703                 CALL    CHLDE    ; Test addresses
02D8 CCD101                 CZ      ADPRT    ; Bad cell
02DB C3E802                 JMP     CONT     ; Continue test
02DE                ;
02DE B9             DNZT:   CMP     C        ; See if same as test data
02DF C2E502                 JNZ     BADD     ; Jump if bad data
02E2 CD1703                 CALL    CHLDE    ; Test addresses
02E5 C4D101         BADD:   CNZ     ADPRT
02E8 CDF301         CONT:   CALL BREAK       ; See if abort wanted
02EB E3                     XTHL             ; Unscramble registers
02EC EB                     XCHG
02ED 23                     INX     H        ; Next address
02EE 1B                     DCX     D
02EF 7B                     MOV     A,E
02F0 B2                     ORA     D        ; Done on this cell?
02F1 C2CD02                 JNZ     WLKB     ; No, jump
02F4 E1                     POP     H        ; Get test address
02F5 C1                     POP     B        ; Get data
02F6 33                     INX     SP
02F7 33                     INX     SP
02F8 D1                     POP     D        ; Get block size
02F9 D5                     PUSH    D
02FA 3B                     DCX     SP
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
```

```
0332 23        BRNC:  INX    H        ; Go to next address
0333 1B               DCX    D        ; Do loop control
0334 7B               MOV    A,E
0335 B2               ORA    D
0336 C22C03           JNZ    BRNB
0339 E1               POP    H        ; Get base address
033A D1               POP    D        ; Get block size
033B C1               POP    B        ; Get data seed, error count
033C D5               PUSH   D        ; Restore them
033D E5               PUSH   H
033E 7E        BRND:  MOV    A,M      ; Read data byte
033F B9               CMP    C        ; Check it
0340 CA4703           JZ     BRNE     ; Skip if OK
0343 04               INR    B        ; Error count
0344 CDDB01           CALL   ADPRA    ; Log the error
0347 0C        BRNE:  INR    C        ; Change test data
0348 C24C03           JNZ    BRNF     ; Skip if not zero
034B 0C               INR    C        ; Reset to 1
034C 23        BRNF:  INX    H        ; Next address
034D 1B               DCX    D        ; Loop control
034E 7B               MOV    A,E
034F B2               ORA    D
0350 C23E03           JNZ    BRND
0353 E1               POP    H        ; Reset base address
0354 D1               POP    D        ; and block size
0355 CD0901           CALL   CST      ; Time to quit
0358 CA2903           JZ     BRNA     ; No, do it again
035B CD0301           CALL   CONI     ; Get character
035E FE03             CPI    'C'-CNTL
0360                                  ; ETX (Cntl C)?
0360 C22903           JNZ    BRNA     ; No, continue
0363 CD1E01           CALL   CRLF
0366 78               MOV    A,B      ; Error count
0367 CD3301           CALL   HEX2     ; Print it
036A 217003           LXI    H,ERMSG  ; Get error message address
036D C3AE01           JMP    PRTWA    ; Print it and return to EXEC
0370           ;
0370 20455252  ERMSG:  DB    ' ERROR','S'+80H
0374 4F52D3
0377 434F4E54  CBMSG:  DB    'CONTINUOUS BURNIN',' '+80H
037B 494E554F
037F 55532042
0383 55524E49
0387 4EA0
0389           ;
0389           ; Routines INIT and EXEC initialize the computer and
0389           ; monitor  the  console for a command.  When a valid
0389           ; command is received, control is transferred to the
0389           ; appropriate routine.
0389           ;
0389 219002    RETN:  LXI    H,TDMSG  ; Print 'TEST DONE'
038C CDAB01           CALL   PRTWD
038F 314000    INIT:  LXI    SP,STACK         ; Set stack pointer
0392 21AC03    EXEC:  LXI    H,DIMSG  ; Print diag message
0395 CDAB01           CALL   PRTWD
```

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

TOTAL ERRORS=00
```

CHAPTER 3

THEORY OF OPERATION

This chapter is intended for those users who want a more thorough understanding of the 2116 operation than they need to make the 2116 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 2116 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 selection 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 2116 controls -CS through the address decoders; R/W goes low when either -pWR or MWRITE is active.

3.2 MEMORY ADDRESSING

Addressing a specific memory location on the 2116 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 each chip through a common address bus. Chip 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, G2A and G2B low and G1 high enabling a decoder.

Decoder enabling is controlled by the Address Select circuitry. 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. All of the memory groups are ORed and the output is NANDed with the MEM line (high when sINTA, sINP, and sOUT are all low) to form the -SEL line. -SEL is the G2A input of the U10 and the G2B input of the U9. Thus if no memory group on the board is addressed, both chips are disabled by -SEL high.

If -SEL is low, the ORed outputs of groups A and B and groups C and D determine which decoder is enabled. U9's G2A is permanently pulled low. If the ORed output of groups A and B is high U9 is enabled through G1 and U10 is disabled through G2B. If the ORed output of groups A and B is low and the ORed output of groups C and D (U10's G1 input) is high, U10 is enabled and U9 is disabled.

Chip selection within the enabled decoder is determined by inputs A, B, and C. U9's C input is tied to the output of group A's memory-address-comparison circuitry; if group A is addressed, C is high, and one of the columns enabled by decoder outputs 4-7 will be selected. In the same way group C's memory-address-comparison circuitry determines which group U10 will select. Address lines A10 and A11 are the A and B inputs of the decoders, determining which of the four columns in a group will be selected.

The 2116 decoding scheme provides full prioritizing of the memory groups. If either or both of groups A and B are addressed, U9 is enabled and U10 is disabled; whether group C or D is addressed is irrelevant. Group selection by the decoders is determined by whether or not group A or C has been addressed, groups B and D being irrelevant. Thus group A has the highest priority, followed in order by groups B, C, and D. If two or more memory groups are given identical addresses, only the highest priority group will be selected when that address is received. The other groups will effectively be buried; they will be unaddressable and will occupy no memory space.

3.3 BANK SELECTION

The CCS 2116 is bank-selectable by bank port address and bank byte. Thus it is fully compatible with Cromemco, Alpha Micro, and other bank port systems. IT IS NOT COMPATIBLE WITH ADDRESS-SELECT SYSTEMS SUCH AS IMSAI.

You assign the 2116 to a bank by jumper-setting the bank port address and the bank byte. To enable a bank during operation, the processor must address the bank port through the low order byte on the address bus and put the bank byte on the data bus. When the processor is in an I/O cycle (sOUT or sINP high), the 2116 compares the low-order byte on the address bus with the user-selected bank port address. If the two match, the 2116 compares the bank byte on the data bus with the user-selected bank byte. The bank-dependent memory groups are enabled or disabled according to whether or not the two bytes designate the same bank.

The 2116 compares -A0 through -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-address-comparison line is ANDed with the I/O line, and the resulting output is NANDed with inverted -pWR to form the BANK CLK line. This line clocks a D-type positive-edge-triggered flip-flop.

The bank address and I/O lines go high first. As long as -pWR is inactive (high, inverted low) the BANK CLK line is low. When -pWR goes active (low, inverted high) the BANK CLK line goes high, clocking the flip-flop. In the meantime the bank byte is written onto the data bus. A high signal on any of the data lines indicates that the corresponding bank is being selected (data lines DO0-7 corresponding to banks 0-7). The bank byte signals are inverted for comparison with the user-selected bank byte. Jumper-selecting a bank connects the corresponding data line to the BANK DATA line; a low signal on that line pulls BANK DATA low. Other jumpers may also be connected and more than one bit of the bank byte on the data bus can be high; the open-collector output will be pulled low whenever a high-inverted-low data line is jumper-connected.

When the flip-flop is clocked by -pWR going low 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. A low on the BANK DATA line resets Q to low, lighting the Bank Select LED. A high on the BANK DATA line sets Q, and therefore -BANK ENABLE, to high. -BANK ENABLE high is inverted to disable the memory groups that are jumper-set bank-dependent (see BANK-INDEPENDENCE below).

The processor can determine whether a bank has been selected by reading DI0 at the bank port address. When pDBIN is active and the bank port has been addressed, the BANK READ ENABLE line is high. This line is NANDed with -Q, which is high when the 2116's bank has been selected. A low output from the NAND pulls DI0 low, acknowledging to the processor that a bank has been enabled.

The flip-flop will be reclocked the next cycle in which the bank port address is received and the I/O line is high, at which point the new bank byte will be compared and Q and -Q set or reset depending on the BANK DATA line input to D. Until then the bank-dependent memory groups will remain enabled.

3.4 BANK-INDEPENDENCE

The 2116 allows you to make any memory group independent of bank selecting by setting a jumper so that the inverted -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 open collector output from being pulled low when the -Bank Enable line is active. The memory group will therefore be enabled whenever it is addressed, independent of which bank has been selected.

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 (-WR line low). If the -WR line is low the buffer allows data to be written to the RAMs.

Read-enabling is more involved. Basically, if the Phantom jumper is off DIO-7 will be enabled whenever a memory group on the board is addressed and the processor is in a memory read cycle. If the Phantom jumper is on, a low on -PHANTOM will disable DIO-7. -PHANTOM is generated by another device in the system and allows that device to overlay identically-addressed memory locations on the 2116 board by preventing 2116 data from reaching the data bus. Thus data is read from the overlaying device only.

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 2116 is available with 200, 300, or 450 nsec Rams, and because processor speeds vary, the Wait feature on the 2116 has been made jumper-selectable. If the Wait jumper is set to on, pRDY will be pulled low whenever pSYNC goes high and the board is selected (-SEL low). 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 jumper is set to off the 2116 does not pull pRDY low and a Wait state does not occur unless it originates elsewhere.

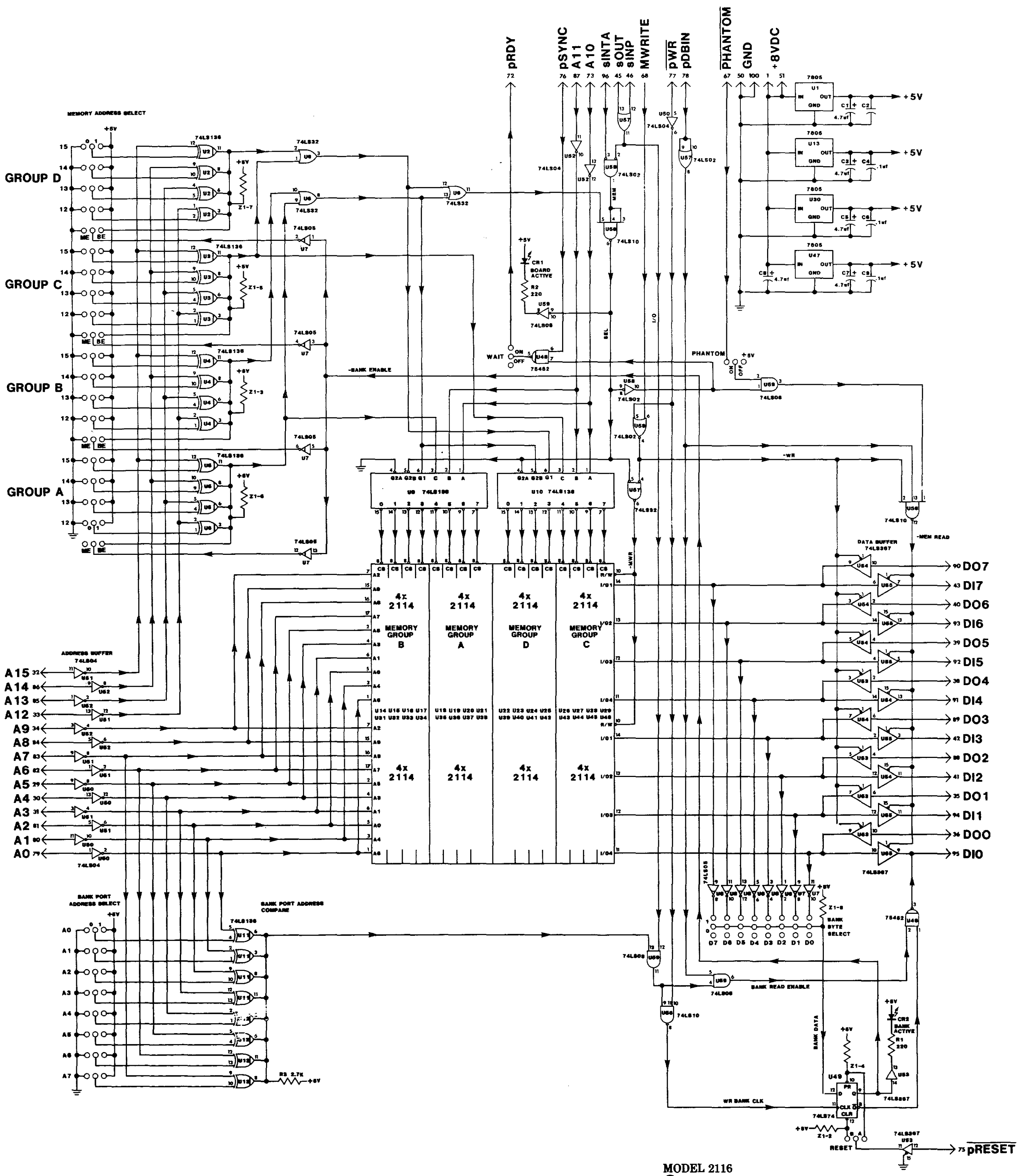
3.7 RESET

The Reset jumper allows you to choose whether or not the 2116 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 operate normally. The -pRESET line can be jumper-set 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 Preset low, the flip-flop is set (Q high), 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 (Q low), and the bank-dependent memory groups are enabled.

CHAPTER 4

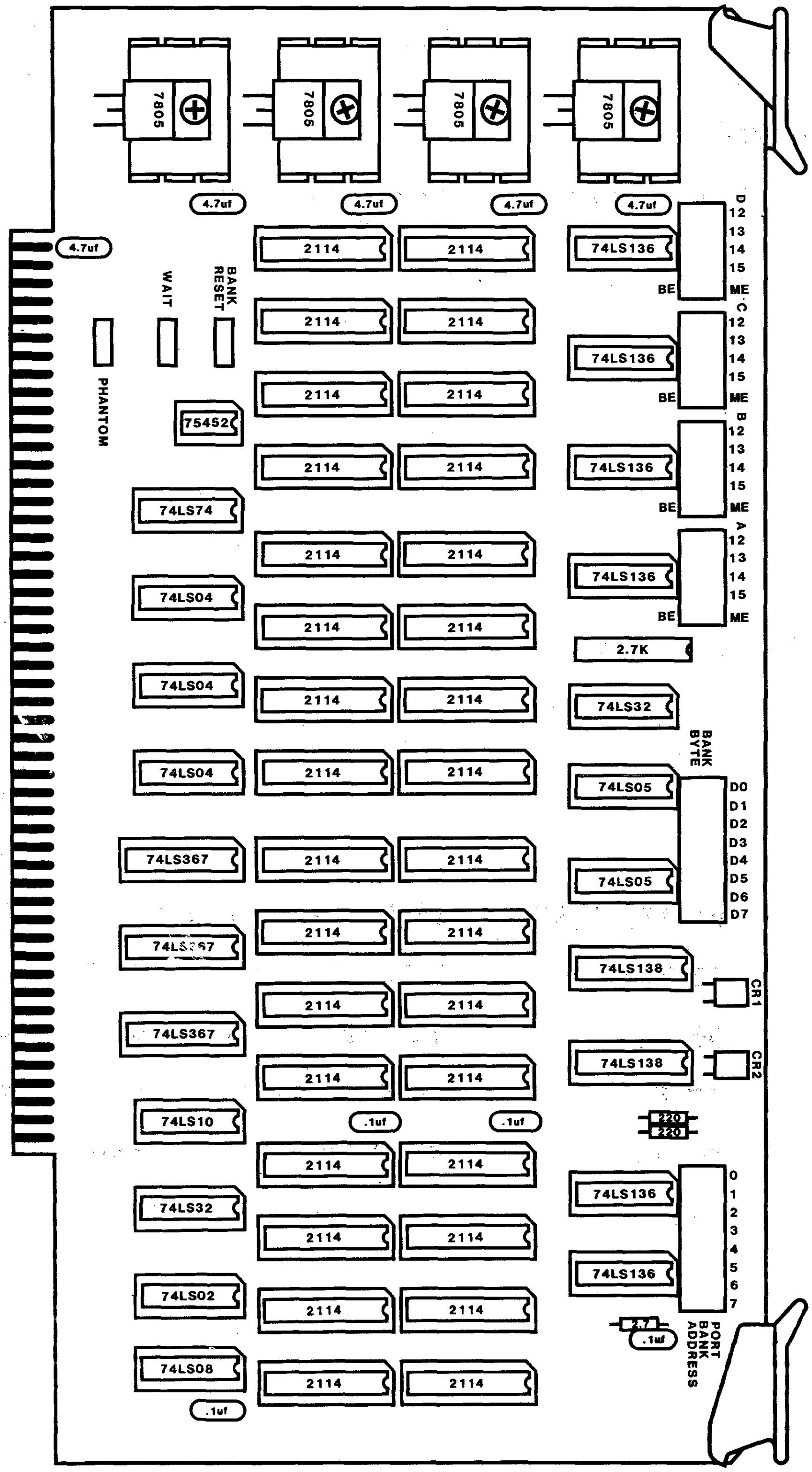
TECHNICAL INFORMATION

4.1 SCHEMATIC/LOGIC DIAGRAM



MODEL 2116
©CCS 1980
16K(4K×4) INDEPENDENT
BANK SELECTABLE
STATIC RAM BOARD

4.2 ASSEMBLY COMPONENT LAYOUT



4.3 PARTS LIST

QTY	REF	DESCRIPTION	CCS PART #
CAPACITORS			
5	C1,3,5,7,8	Tantalum, 4.7uf, 35 vdc, 20%	42084-54756
4	C2,4,6,9	Ceramic, .1uf, 50 vdc, 20%	42142-21046
RESISTORS			
2	R1,2	220 ohm, 1/4 w, 5%	40002-02215
1	R3	2.7K ohm, 1/4w, 5%	40002-02725
1	Z1	Resistor Network, SIP 2.7K ohm x 7	40930-72726
INTEGRATED CIRCUITS			
32	U14-29,31-46	MOS 2114 1Kx4 Static RAMS	31900-21142 (200nsec) or -21143 (300nsec) or -21144 (450nsec)
4	U1,13,30,47	7805 +5v regulator	32000-07805
6	U2-5,11,12	74LS136 quad ex-OR:OC	30000-00136
1	U6	74LS32 quad 2-in OR	30000-00032
2	U7,8	74LS05 hex inverter:OC	30000-00005
2	U9,10	74LS138 octal decoder	30000-00138
1	U48	75452 dual NAND: OC	30300-00452
1	U49	74LS74 dual D flip-flop	30000-00074
3	U50-52	74LS04 hex inverter	30000-00004
3	U53-55	74LS367 hex bus driver	30000-00367

QTY	REF	DESCRIPTION	CCS PART #
1	U56	74LS10 tri 3-in NAND	30000-00010
2	U57,58	74LS02 quad 2-in NOR	30000-00002
1	U59	74LS08 quad 2-in AND	30000-00008
IC SOCKETS			
1		IC Socket, 8 PIN	58102-00080
17		IC Socket, 14 PIN	58102-00140
5		IC Socket, 16 PIN	58102-00160
32		IC Socket, 18 PIN	58102-00180
MISCELLANEOUS			
39		Header Strip, 1x3	56004-01003
39		Berg Jumper	56200-00001
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	02016-00003
2		Extractor, PCB Non-locking	60100-00000
2		Roll Pin Extractor Mounting	60100-00001
1		Owner's Manual	89000-02116

4.4 ADDRESS/CHIP TABLE

2116 ADDRESS/CHIP TABLE

		X000-X3FF	X400-X7FF	X800-XBFF	XC00-XFFF
A	HIGH	U18	U19	U20	U21
A	LOW	U35	U36	U37	U38
B	HIGH	U14	U15	U16	U17
B	LOW	U31	U32	U33	U34
C	HIGH	U26	U27	U28	U29
C	LOW	U43	U44	U45	U46
D	HIGH	U22	U23	U24	U25
D	LOW	U39	U40	U41	U42

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.