Owner's Manual

# **Model 2032**

# 32K Static RAM Module



California Computer Systems

# CCS MODEL 2032 32K STATIC RAM MODULE OWNER'S MANUAL

# COPYRIGHT 1980

CALIFORNIA COMPUTER SYSTEMS 250 CARIBBEAN DRIVE SUNNYVALE CA 94086

MANUAL NO. 89000-02032

## TABLE OF CONTENTS

.

FEATURES		ii	L
CHAPTER		SETTING THE 2032 JUMPERS1.1SETTING THE MEMORY GROUP ADDRESSES1-71.2SETTING THE BANK BYTE1-21.3SETTING THE BANK PORT ADDRESS1-21.4SETTING MEMORY GROUP BANK-INDEPENDENCE1-31.5SETTING THE RESET JUMPER1-31.6SETTING THE PHANTOM JUMPER1-31.7SETTING THE WAIT JUMPER1-41.8EXAMPLES OF JUMPER SELECTION1-5	223334
CHAPTER		THEORY OF OPERATION2-12.1MEMORY2-22.2MEMORY ADDRESSING2-22.3BANK SELECTION2-22.4BANK-INDEPENDENCE2-32.5DATA BUFFERS2-12.6WAIT STATES2-12.7RESET2-1	22344
		TESTING AND TROUBLESHOOTING THE 20323.1 FRONT PANEL QUICK CHECKOUT3-13.2 DIAGNOSTIC TEST OVERVIEW3-23.3 PREPARING DRIVER ROUTINES3-43.4 SETTING UP FOR THE TEST3-53.5 LOADING THE DIAGNOSTIC3-53.6 RUNNING THE DIAGNOSTIC3-53.7 ERROR PRINTOUT INTERPRETATION3-63.8 SAMPLE MEMORY DIAGNOSTIC RUN3-103.9 MEMORY DIAGNOSTIC LISTING3-10	3455580
CHAPTER	4	TECHNICAL INFORMATION4.1SCHEMATIC/LOGIC DIAGRAM4.2ASSEMBLY COMPONENT LAYOUT4.3PARTS LIST4.4CONTROL ROM TRUTH TABLE4.5ADDRESS/CHIP TABLE4.62032 BUS CONNECTOR PINOUT	3 4 6 7

APPENDIX A LIMITED WARRANTY

#### FEATURES

Uses Popular 2114 Static RAMs Available with 200, 300, or 450 nsec RAMs Berg Jumpers Used for Selectable Features 8K Memory Blocks Individually Addressable to Any 8K Boundary Bank Selection by Bank Port and Bank Byte 8K Blocks Individually Bank-Enabled LEDs Indicate Board Active and Bank Active States Wait State Jumper Phantom Line Capability Optional Board-Enabling on Reset Operates on +8 Volts Fully Buffered Meets IEEE Proposed S-100 Signal Standards Diagnostic Software Included FR-4 Epoxy PC Board Solder-Masked on Both Sides Silk Screen of Part Numbers and Reference Designations

#### CHAPTER 1

#### SETTING THE 2032 JUMPERS

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

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

#### 1.1 SETTING THE MEMORY GROUP ADDRESSES

In order to provide maximum flexibility in the location of the 2032's memory groups within a bank, CCS has made the addresses of the four memory groups jumper-selectable. The jumper-set address of a memory group is compared with the high-order address lines A13-A15, and if the address matches, the group is selected. The Group Address (GRP ADDR) jumpers are in the upper left corner of the board (with the connector pins at the bottom). Set the jumpers of each group to the three high-order binary digits that specify the multiple of 8K at which you wish to locate the group. For example, the addresses of the block between 16K and 24K are 4000h-5FFFh, so you would locate a group in that block by setting its jumpers to 010. Since a memory group's base address must be a multiple of 8K, an easy way to calculate the jumper settings is to divide the base address by 8K. You can then set the jumpers to the binary equivalent of the result.

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

#### 1.2 SETTING THE BANK BYTE

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

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

#### 1.3 SETTING THE BANK PORT ADDRESS

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

#### SETTING THE 2032 JUMPERS

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

#### 1.4 SETTING MEMORY GROUP BANK-INDEPENDENCE

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

#### 1.5 SETTING THE RESET JUMPER

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

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

#### 1.6 SETTING THE PHANTOM JUMPER

The Phantom jumper is in the lower right corner of the board. Setting the jumper to B allows a device that generates a PHANTOM signal to overlay portions of the 2032 memory. For example, CCS peripheral control boards generate Phantom signals when certain ROM locations are addressed; these locations contain code to drive the peripherals. If an identically-addressed location exists on the 2032 board, the Phantom signal will block the output from the board of the contents of that location. This allows you to access the rest of the memory locations within the 8K block that contains the overlayed portion. Without Phantom capability the 2032 would not be able to locate a memory group in that block because the 2032 and the peripheral control board would both put data on the bus when a shared location was addressed.

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

#### 1.7 SETTING THE WAIT JUMPER

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

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

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

#### 1.8 EXAMPLES OF JUMPER SETTINGS

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



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



#### CHAPTER 2

#### THEORY OF OPERATION

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

#### 2.1 MEMORY

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

#### 2.2 MEMORY ADDRESSING

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

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

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

#### 2.3 BANK SELECTION

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

You assign the 2032 to a bank by jumper-setting the bank port address and the bank byte. The 2032 compares AO-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 The BANK PORT ADDRESS line inputs to the output low. If the conditions of the BANK PORT ADDRESS Control ROM. line and the other Control ROM inputs are right (see Section 4.4), BANK CLK will pulse low, clocking the Bank Enable flip-flop when it rises to high again. In the meantime the bank byte becomes present on DIO-7 and is inverted. Setting a Bank Byte Select jumper to 1 connects the corresponding inverted DATA IN line to the BANK DATA line. Thus a low signal on an inverted DATA IN line, indicating a 1 in the bank byte, will pull BANK DATA low if the corresponding Bank Byte Select jumper is set to 1.

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

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

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

#### 2.4 BANK-INDEPENDENCE

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

2.5 DATA BUFFERS

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

#### 2.6 WAIT STATES

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

#### 2.7 RESET

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

#### CHAPTER 3

#### TESTING AND TROUBLESHOOTING THE 2032

3.1 FRONT PANEL QUICK CHECKOUT

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

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



The priority feature will cause Group A to be selected. Set the Front Panel Adress Switches AO-A15 to the off position (0000H). Examine that address. Set the Data Switches D1-D7 to the OFF position and D0 to the ON position (01H). Deposit (write) into memory and compare the Data readout with the switch settings. Now switch D0 to OFF and D1 to ON, deposit into memory again, and compare the result with the switch settings. Continue the pattern of one Data Switch ON and the rest OFF until all data bits have been checked. If any data does not match the switch settings, isolate the malfunction with a logic probe or voltmeter before continuing.

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



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

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



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

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



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

#### 3.2 DIAGNOSTIC TEST OVERVIEW:

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

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

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

#### xx yyyy zz

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

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

#### WWWW XX YYYY ZZ

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

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

#### 3.3 PREPARING DRIVER ROUTINES

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

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

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

CONST: Console status. This routine reads the console input status. If data is not available, then the accumulator is set to 0 and the status flags must match. If data is pending, then a -1 (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.

#### TESTING AND TROUBLESHOOTING THE 2032

3.4 SETTING UP FOR THE TEST:

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



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

#### 3.5 LOADING THE DIAGNOSTIC:

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

#### 3.6 RUNNING THE DIAGNOSTIC

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

DIAGNOSTIC:

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

#### DIAGNOSTIC: WALKING BIT TEST BLOCK SIZE:

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

The computer will now ask for

#### BASE ADDRESS:

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

#### TEST DONE DIAGNOSTIC:

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

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

502 84 502 04

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

#### TESTING AND TROUBLESHOOTING THE 2032

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

#### TABLE 3.1

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

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

the second second second

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

#### 3.7 ERROR PRINTOUT INTERPRETATION:

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

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

# TESTING AND TROUBLESHOOTING THE 2032

.

ERROR CONDITION	PROBABLE CAUSE	SUSPECT PARTS
Bad data=OFFH, all groups	a) bank select b) board select	U5, U6, U85 U3, U5, U85
Random data or all O data, all groups	bad write control	U5, U83, U85
OFFH data, one group only	a) group A select b) group B select c) group C select d) group D select	U2, U3, U9 U2, U3, U42 U1, U3, U60 U1, U3, U70
One address line hung (printout: good data, bad address)	address buffers	U81 (A0-6, A15) U82 (A7-14)
One data line hung a) hung 0 (good address, bad data=0)	grounded data line	u83, u84
b) hung 1 (good address, bad data=1)		U83, U84 U83, U84, memory chips
Soft errors (random addresses and data,	a) memory chip access time	Try setting Wait jumper to B and
non-repeatable)	b) heat-sensitive parts	rerunning tests. Treat as a hard error and replace suspect parts.
Hard memory errors	bad memory chip	See Table 3.1 to identify chip.

# TABLE 3.2

٦

•

١

#### 3.8 SAMPLE MEMORY DIAGNOSTIC RUN:

DIAGNOSTIC: WALKING BIT TEST Typed in W BLOCK SIZE: 30 BASE ADDRESS: 300 BAD BASE ADDRESS: BASE ADDRESS: 400 TEST DONE DIAGNOSTIC: WALKING BIT TEST New test BLOCK SIZE: 400 BASE ADDRESS: 400 TEST DONE DIAGNOSTIC: WALKING BIT TEST BLOCK SIZE: 1000 **BASE ADDRESS: 400** TEST DONE DIAGNOSTIC: WALKING BIT TEST BLOCK SIZE: 1800 BASE ADDRESS: 400 TEST DONE DIAGNOSTIC: ???? DIAGNOSTIC: WALKING BIT TEST BLOCK SIZE: 579 BASE ADDRESS: 400 TEST DONE DIAGNOSTIC: CONTINUOUS BURNIN BLOCK SIZE: 3765 BASE ADDRESS: 3D3 **00 ERRORS** TEST DONE DIAGNOSTIC: CONTINUOUS BURNIN BLOCK SIZE: 3ABC **BASE ADDRESS: 3EF 00 ERRORS** TEST DONE **DIAGNOSTIC:** 

Block may be any size

Base address must be multiple of 1K (400H)

Equal block size, base address

Larger block size test

Typed in 1

Odd block size

Typed in C No parameter restrictions

Up to OFFH (255D) errors shown

1 0000			TITLE	'2114 ME	MORY DIAGNOSTIC	VER 1.1'
2 0000		:				
3 0000		;				
4 0000		; Consol	le innut.	output s	upport routines	
5 0000		, 0011301	re rubao,	output s		
6 0000		) Milliona				77
					ighly-matured, w	
7 0000					l's monitor. Th	
8 0000	•				to converse wit	
9 0000					microprocessor	
10 0000					are the accumul	
11 0000					ng active para	
12 0000				a routi		ack is used
13 0000					t space must be	
14 0000						pointer is
15 0000	. •				al place on exit	t unless an
16 0000				letected		rameters are
17 0000		; return	ned on th	ne stack.	In the latter	r case, the
18 0000		; stack	is offse	et by 2 t	imes the request	ted number of
19 0000		; parame	eters ar	nd will	be set right	after these
20 0000		; parame			off the stack.	
21 0000						
22 0000		Regist	ter use d	conforms	to ICOM and	CP/M defined
23 0000						d in the C
24 0000					ta is expected	
25 0000					ines require CP	
26 0000					ines as conta	
27 0000					or CI and CO a	
28 0000			ent ROM.	program,	or er and co a	S III UNE ICON
29 0000		; nesiu	ent nom.		· · ·	
	000+	j.	ROU	0.4.17	. AGOTT 14 8-	- 4
30 0000	A 0 0 0	LF	EQU	OAH	; ASCII line fe	
31 0000	000D	CR	EQU	ODH	; ASCII carriag	
32 0000	0040	CNTL	EQU	40H	; ASCII Cntl of	fset
33 0000	0040	STACK	EQU	40H		
34 0000		• • •				
35 0000		;	·		•	
36 0000		;		•	· · · · · · · · · · · · · · · · · · ·	
37 0000	0040		ORG	40H		
38 004 <u>0</u>		;				
39 0040	C38F03	• .	JMP	INIT		
40 0043		;				
41 0043	0100		ORG	0100H		
42 0100		;	÷			
43 0100		; SYSTE	M LINKAG	ES		
44 0100						
45 0100	C003	CONIN	EQU	0C003H		
46 0100	C006	CONOUT	EQU	0С006Н		
47 0100	C373	CONST	EQU	0C373H		
48 0100	C000	USER	EQU	осооон		
49 0100		;				
50 0100	C38F03	7	JMP	INIT		
51 0103		CONI:	JMP	CONIN		
52 0106		CONO:	JMP	CONOUT		
53 0109		CST:	JMP	CONST		
54 010C		ERR:	JMP	USER		
54 010C			UNI	U D D R		
25 0101		;				

; Routine BLK prints one blank on the current ; console device. 56 010F 57 010F ; ; Entry parameters: None . Return 58 010F 59 010F ; Return parameters: None ; Stack usage: 4 byt 60 010F 61 010F 62 010F 4 bytes 

 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 68 0115 68 0115 70 0115 69 0115 60 0115 60 0115 60 0115 60 0115 60 0115 60 0115 60 0115 71 0115 72 0115 ; Entry parameter: 4 bit binary number in 73 0115; Exit parameter:lower half of accumula74 0115; Stack usage:0 bytes75 0115; lower half of accumulator 760115E60FCONV:ANIOFH; Clear high bits770117C690ADI90H; Insert partial ASCII78011927DAA; Zone79011ACE40ACI40H; Insert rest of ASCII80011C27DAA; Zone 81 011D C9 RET ; ; Routine CRLF prints an ASCII carriage return and ; line feed (in that order) on the console. It ; follows these with 4 blanks to create a left ; margin. 82 011E 83 011E 84 011E 85 011E 86 011E 0, 011E 88 011E 89 011E 90 011E 91 011E 92 011T ; Entry parameter: None ; Exit parameter: None ; Stack Usage: 8 by1 8 bytes 

 90 011E
 ;

 91 011E
 ;

 92 011E E5
 CRLF: PUSH
 H
 ; Save (H,L)

 93 011F 212701
 LXI
 H,CRMSG; Get message address

 94 0122 CDAE01
 CALL
 PRTWAA; Print message

 95 0125 E1
 G
 POP
 H
 ; Restore (HL)

 96 0126 C9
 RET

 98 0127 0D0A20A0 CRMSG: DB CR,LF, ', '+80H 99 012B Routine DEPRT prints the contents of the (DE) register pair as a 4-digit hexadecimal number on the console. 100 012B 101 012B 102 012B 103 012B 104 012B 105 012B ; Entry parameter: (DE) = 4 digit hex number to be printed on console. ; Exit parameter: ; Stack usage: 106 012B None 107 012B 108 012B 10 bytes 109 012B CD1E01DEPRT: CALLCRLF; Print a CR, LF110 012E: Alternate entry point if no CR, LF wan 110 012E ; Alternate entry point if no CR, LF wanted

# TESTING AND TROUBLESHOOTING

111012E 7A 12 012F CD3301DEFPA: NOVA,D: Get high order byte1130132 7BHOVA,E : Get low order byte1140133: Alternate entry point to print (A) as two hex1150133: Alternate entry point to print (A) as two hex1160133 F5HEX2: PUSHPSW1170134 0FRRC : Kove high order nibble1180135 0FRRC : to lower half of (A)1190136 0FRRC : to lower half of (A)1200137 0FRRC : to lower half of (A)1210138 0GSC01CALL1220136 0FRRC : do console1230137: Alternate entry point to print low order nibble1240130: Alternate entry point to print low order nibble1250132: Alternate entry point to print (A) as sumed1260142: Conton encle1270142: Entry parameter:1300142: that the console is in a full duplex mode.1310142: Entry parameter:1310142: Stack usage:1320142: Stack usage:1330142: Alternate entry point to print (A)1340142: Alternate entry point to Print (A)1350142: Stack usage:1360145: Alternate entry point to print (A)1370142: Stack usage:1380145: Alternate entry point to print (A)1390145: Alternate entry point for BLK routine </th <th></th> <th></th> <th></th> <th></th>				
112 012F CD3301 113 0132 7B MOV A,E ; Get low order byte 114 0133 ; digits 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 ; to lower high order nibble 120 0137 0F RRC ; del low nibble back 123 0130 ; Alternate entry point to print low order nibble 125 0136 CD1501 HEX1: CALL HEX1 ; Print the nibble 125 0137 CJ1501 HEX1: CALL CONV ; Convert to ASCII 126 0137 C34501 JMP ECH1 ; Go print it 127 0142 ; routine ECH0 reads one character from the calling 129 0142 ; routine and then echoes it back. It is assumed 130 0142 ; that the console is in a full duplex mode. 131 0142 ; 132 0142 ; Extr parameter: Mone 134 0142 ; 137 0142 CD0301 ECH0: CALL CONI ; Read a character 136 0142 ; 137 0142 CD0301 ECH0: CALL CONI ; Read a character 136 0142 ; 137 0142 CD0301 ECH0: CALL CONI ; Read a character 138 0145 ; Alternate entry point to print (A) 139 0145 C5 ECH1: PUSH B ; Save (BC) 140 0146 E677 ANI TFH ; Strip off parity bit 141 0148 4F MOV C,A ; Put character into (C) 145 0149 ; Alternate entry point for BLK routine 143 0149 CD0601 ECR2: CALL CONO ; Output it 144 0142 C11 POF B ; Restore (BC) 145 0148 ; 150 0148 ; 151 0148 ; 150 0148 ; 151 0148 ; 151 0148 ; 152 0148 CD1801 HLPRT; CALL CRF ; Print a (CR, LF) 155 0148 CD1801 HLPRT; CALL CRF ; Print a (CR, LF) 155 0148 CD1801 HLPRT; CALL CRF ; Print a (CR, LF) 156 0151 ; Alternate entry point if no CR, LF wanted 157 0151 EB HLPRA: XCBE ; Wang (HL), (DE) 158 0152 CD2801 CALL DEPAA ; Go print (DE) 159 0155 CB CD1 CALL DEPAA ; Go print (DE) 150 0156 C9 RET 164 0157 ; Routine PCHK reads a character from the console 163 0157 ; Routi	111 012E 7A	DEPRA: MOV	A,D	: Get high order byte
113 0132 7B MOV A,E ; Get low order byte 114 0133 ; Alternate entry point to print (A) as two hex 115 0133 ; Alternate entry point to print (A) as two hex 116 0133 F5 HEZ: PUSH PSW ; Save low order byte 117 0134 OF RRC ; to lower half of (A) 119 0136 OF RRC ; to lower half of (A) 119 0136 OF RRC 120 0137 OF RRC 121 0138 CD3CO1 CALL HEX1 ; Print the nibble 122 0138 F1 POP PSW ; Get low nibble back 123 013C ; Alternate entry point to print low order nibble 124 013C (JATERNE CHI ; Go print it 127 0132 CD1501 HEX1: CALL CONV ; Convert to ASCII 126 0137 C34501 JHF ECH0 reads one character from the calling 129 0142 ; routine and then echoes it back. It is assumed 130 0142 ; that the console is in a full duplex mode. 131 0142 ; Entry parameter: None 133 0142 ; Entry parameter: None 134 0142 ; Stack usage: A bytes 135 0142 ; Alternate entry point to print (A) 139 0145 C5 ECH0: CALL CONI ; Read a character 136 0142 ; Alternate entry point to print (A) 139 0145 C5 ECH0: CALL CONI ; Read a character 130 0142 ; Alternate entry point to print (A) 130 0145 C5 ECH0: CALL CONI ; Put character into (C) 142 0149 ; Alternate entry point to print (A) 139 0145 C5 ECH1: PUSH B ; Save (EC) 140 0146 E67F ANI 7FH ; Strip off parity bit 141 0148 4F ANI 7FH ; Strip off parity bit 144 0142 C1 POP B ; Restore (BC) 145 0148 ; register as 4 hexadecimal digits on the console. 150 0148 ; text parameter: Mone 151 0148 ; Entry parameter: Mone 152 0148 ; Entry parameter: Mone 153 0148 ; Entry parameter: Mone 154 0148 ; 154 0148 ; Entry parameter: Mone 155 0148 C1 RET 155 0148 C1 RET 156 0151 ; Alternate entry point for ELK routine 157 0151 EB HLPRT; CALL CREF ; Print a (CR, LF) 156 0151 ; Alternate entry point for 0CR, LF wanted 157 0151 EB HLPRT; CALL DEPRA ; Go print (DE) 156 0151 ; Alternate entry point for 0CR, LF wanted 157 0151 EB HLPRT; CALL DEPRA ; Go print (DE) 156 0151 ; Routine PCHK reads a character from the console 159 0155 CP RET 150 0156 C9 RET 150 0156 C9 RET 150 0157 ; Routine PCH	112 012F CD3301	CAL		
<pre>114 0133     ; Alternate entry point to print (A) as two hex 115 0133     r5, digits 116 0133 F5     HEX2: PUSH PSW ; Save low order byte 117 0134 0F     RRC ; to lower high order nibble 118 0135 0F     RRC 120 0137 0F     RRC 121 0138 CD3C01     CALL HEX1 ; Print the nibble 122 0138 F1     POP PSW ; Get low nibble back 123 013C     ; Alternate entry point to print low order nibble 125 013C CD1501     HEX1: CALL CONV ; Convert to ASCII 126 0137 C34501     j routine ECHO reads one character from the calling 129 0142     ; routine and then echoes it back. It is assumed 130 0142     ; that the console is in a full duplex mode. 131 0142     ; Extry parameter: None 134 0142     ; Stack usage: 4 bytes 136 0142     ; Alternate entry point to print (A) 139 0145 C5     BCH1: PUSH B     ; Save (BC) 140 0146 E677     ANI 7FH ; Strip off parity bit 141 0148 4F     MOV C,A     ; Put character into (C) 145 0149     ; Alternate entry point to BLK routine 143 0149     ; Alternate entry point to print (A) 140 0142     ; 141 0148 4F     MOV C,A     ; Put character into (C) 145 0140     ; Alternate entry point to BLK routine 150 0142     ; Alternate entry point to print (A) 139 0145 C5     BCH1: PUSH B     ; Save (BC) 144 0142     ; 140 0146 E677     ANI 7FH ; Strip off parity bit 141 0148 4F     MOV C,A     ; Put character into (C) 145 0140     ; Alternate entry point to be printed 150 0142     ; Alternate entry point to be printed 150 0142     ; Alternate entry point to be printed 150 0142     ; Alternate entry point to print (A) 139 0145 C5     BCH1: PUSH B     ; Save (BC) 145 0140     ; Alternate entry point to print (A) 150 0142     ; Alternate entry point to print (A) 150 0145     ; Alternate entry point to print (A) 150 0145     ; Alternate entry point for BLK routine 151 0148     ; 164 0148     ; 175 0140     ; Alternate entry point it on CR,LF wanted 152 0148     ; Stack usage: 10 bytes 154 0148     ; 155 0148     ; Stack usage: 10 bytes 154 0148     ; 150 0151     ; Alternate entry point if no CR,LF wanted 152 0145</pre>				•
1150133; digits1160133 F5HEX2: PUSHPSW; Save low order byte11701340FRRC; Hove high order nibble11801350FRRC; to lower half of (A)11901360FRRC;1210138CONTCALLHEX1; Print the nibble1220138F1POPPSW; Get low nibble back123013C: Alternate entry point to print low order nibble124013C: Alternate entry point is; Go print it125013CCD1501HEX1: CALLCONV; Convert to ASCII1260137C34501JMFECH1; Go print it1270142; Foutine ECH0 reads one character from the calling1290142; that the console is in a full duplex mode.1310142; Entry parameter:None1320142; Entry parameter:None1330142; Stack usage:4 bytes1360142;< Alternate entry point to print (A)			•	
1160133F5HEX2:PUSHFSW; Save low order byte11701340FRRC; Move high order nibble11901360FRRC12001370FRRC121013605001CALLHEX1; Print the nibble1220138C13001CALLHEX1; Get low nibble back123013C; Alternate entry point to print low order nibble124013C; on console125013CCD1501JMPECH11260137C34501JMPECH1; Go print it1270142; Foutine ECH0 reads one character from the calling1290142; that the console is in a full duplex mode.1310142; Entry parameter:None1320142; Entry parameter:None1330142; Stack usage:4 bytes1360142; Alternate entry point to print (A)1370142; Alternate entry point for BLK routine136147MOVCA1370142; Alternate entry point for BLK routine1380145; Alternate entry point for BLK routine1390145CSECH2: CALL1400146EGT21410146; Register as 4 hexadecimal digits on the console.1430149; Alternate entry point for BLK routine1440146; Register as 4 hexadecimal digits on the console.1490148;			entry point	to print (A) as two hex
11701340FRRC; Hove high order nibble11801350FRRC; to lower half of (A)11901360FRRC12001370FRRC1210138C32(01)CALLHEX1 ; Print the nibble1220138F1POPPSW124013C; Alternate entry point to print low order nibble125013Cif Alternate entry point is go print it1260137CD1501HEX1:1260137CD1501HEX1:1270142; Foutine ECH0 reads one character from the calling1290142; routine and then echoes it back. It is assumed1300142; that the console is in a full duplex mode.1310142;1330142; Exit parameter:1340142; Exit parameter:1350142; Stack usage:1360142; Alternate entry point to print (A)1390142; Alternate entry point to print (A)1390142; Alternate entry point for BLK routine1400146ECH0:14101484F142MOVC,A1430149CD0601144HOVC,A145i Rest14601481471048148i Rest149i Alternate entry point if no CA, LF watted1500148151i Alternate entry point if no CA, LF watted </td <td></td> <td>· •</td> <td>•</td> <td></td>		· •	•	
11801350FRRC; to lower half of (A)11901360FRRC12001370FRRC1210138C3LLHEX1; Print the nibble122013BF1POPPSW; Get low nibble back123013C; Alternate entry point to print low order nibble124013C; on console125013C(ALLCALL126013FG345011270142; Routine ECHO reads one character from the calling1280142; that the console is in a full duplex mode.130142; that the console is in a full duplex mode.1310142; Entry parameter:None1330142; Exit parameter:None1340142; Stack usage:4 bytes1360142; Alternate entry point for print (A)1390142; Alternate entry point for BLK routine1410148f130144f141148f1410148f1410148f142f14301491440140145f145f146f147148149144144144145144145145146147148149				; Save low order byte
11801350FRRC; to lower half of (A)11901360FRRC12101370FRRC1210138CD3C01CALLHEX1 ; Print the nibble1220138F1POPPSW; Get low nibble back123013C; Alternate entry point to print low order nibble124013C; Alternate entry point to print low order nibble125013CCD1501HEX1:CALL126013FC34501JMPECH1 ; Go print it1270142; Routine ECH0 reads one character from the calling1280142; that the console is in a full duplex mode.1310142; that the console is in a full duplex mode.1310142; Exit parameter:None1330142; Stack usage:4 bytes1360142; Stack usage:4 bytes1360142; Alternate entry point for print (A)1390142; Alternate entry point for ELK routine140145; Alternate entry point for ELK routine1410148FHOVC.A1430149CD0601ECH2:CALL1440140ECH2; Stare (BC)145: Alternate entry point for ELK routine1440146;145: Alternate entry point for ELK routine146: Exit parameter:HOP1470148;1481491490148<	117 0134 OF.	RRC		; Move high order nibble
1190136OFRRC1200138CD3C01CALLHEX1; Print the nibble1210138CD3C01CALLHEX1; Print the nibble1220138F1POPPSW; Get low nibble back123013C; Alternate entry point to print low order nibble124013C; on console125013CGALLCONV; Convert to ASCII126013FC34501JMPECH1; Go print it1270142; RoutineECH0 reads one character from the calling1290142; routine and then echoes it back.It is assumed1310142; Entry parameter:None1330142; Entry parameter:(A)1340142; Stack usage:4 bytes1360142; Alternate entry point to print (A)1370142CD0301ECH0:1360142; Alternate entry point for print (A)1390145C5ECH1:1300145(AL CONO)1310144C11410146ECH2:142(AL CONO); Output it1430149(BC1440140ECH2:1450149(AL CONO)1461471470148147O148148149149C1140C1140C11410140142(AL CONO) </td <td>118 0135 OF</td> <td>RRC</td> <td></td> <td>: to lower half of (A)</td>	118 0135 OF	RRC		: to lower half of (A)
12001370FRRC1210138CD3C01CALLHEX1 ; Print the nibble122013BF1POPPSW ; Get low nibble back123013C; Alternate entry point to print low order nibble124013C; on console125013CCD3501HEX1:126013FCJ3501JMP1270142;1280142; Routine ECH0 reads one character from the calling1290142; routine and then echoes it back. It is assumed1300142; Exit parameter:None1310142; Exit parameter:None1320142; Exit parameter:None1330142; Stack usage:4 bytes1360142; Stack usage:4 bytes1360142; Alternate entry point to print (A)1390145C5ECH1: PUSH B1390145C5ECH1: PUSH B1300142; Alternate entry point for plt. K routine1310144f1410148f1330142; Stack usage:1340144f140144150ECH1: PUSH B137142138014513901451410148130f142f1410148142f1430149144f145f146 </td <td>119 0136 OF</td> <td>RRC</td> <td></td> <td></td>	119 0136 OF	RRC		
1210138CD3C01CALLHEX1; Print the nibble1220138F1POPPSW; Get low nibble back123013C; Alternate entry point to print low order nibble124013C; on console125013CGONSOLE126013FC345011270142;1280142;1290142;1200142;1210142;12211421230142124;124;125014212611421270142128014212901421201142121;1211142122;1230142124;124;1250142126;12701421280142129;1201142121;121:1221:123101421242:125:126:127:126:127:128:129:120142::::::::::::::::: <t< td=""><td></td><td></td><td></td><td></td></t<>				
122013B F1POPFSW; Get low nibble back123013C; Alternate entry point to print low order nibble124013C; on console125013C CD1501HEX1: CALLCONV; Convert to ASCII126013F C34501JMPECH1; Go print it1270142;Routine ECH0 reads one character from the calling1280142; that the console is in a full duplex mode.1310142; that the console is in a full duplex mode.1310142; Exit parameter:None1330142; Exit parameter:None1340142; Stack usage:4 bytes1360142;Stack usage:4 bytes1360142;Stack usage:4 bytes1360142;Alternate entry point to print (A)1390145C5ECH1: FUSH B; Stark usage:1390145C5ECH2: CALLCONI1400146ECT7ANI7FH1410148HPMOVC,A1430149; Alternate entry point for ELK routine1430149; Alternate entry point for ELK routine1440146ECH2: CALLCONO142; Stack usage:1 be printed1430148;1440146ECH2: CALL144144145: Alternate entry point for ELK routine146014E147HP </td <td></td> <td></td> <td></td> <td>. Daint the mibble</td>				. Daint the mibble
123013C; Alternate entry point to print low order nibble124013C; on console125013C CD1501HEX1: CALLCONV; Convert to ASCII126013F C34501JMPECH1; Go print it1270142;Routine ECH0 reads one character from the calling1290142; routine and then echoes it back. It is assumed1300142; that the console is in a full duplex mode.1310142; Entry parameter:None1330142; Exit parameter:None1340142; Stack usage:4 bytes1350142; Stack usage:4 bytes1360142;Stack usage:4 bytes1370142CD0301ECH0: CALLCONI ; Read a character1380145; Alternate entry point to print (A)1390145C5ECH1: PUSH B; Save (BC)1400146EG7FMNI TFH; Strip off parity bit1410148for BLK routinefor BLK routine1430149CD0601ECH2: CALLCONO ; Output it1440140C14FOP B; Restore (BC)1450148; register as 4 hexadecimal digits on the console.1460148; register as 4 hexadecimal digits on the console.1470148; Exit parameter:None1500148; Exit parameter:None1510148; Exit parameter:None1520148 <td></td> <td></td> <td></td> <td>•</td>				•
124013C; on console125013CCD1501HEX1:CALLCONV; Convert to ASCII126013FC34501JMPECH1; Go print it1280142;Routine ECH0 reads one character from the calling1290142;routine and then echoes it back. It is assumed1300142;that the console is in a full duplex mode.1310142;Entry parameter:None1320142;Entry parameter:None1330142;Ext parameter:(A) = Character read from1340142;Stack usage:4 bytes1350142;Stack usage:4 bytes1360142;Stack usage:4 bytes1370142CD0301ECH0:CALLCONI1380145;Alternate entry point to print (A)1390145C5ECH1:PUSHB1300146E67FANI7FH1400146E67FANI7FH1410146ECH2:CALLCONO; Output it142POPB; Restore (BC)1430149; Alternate entry point for BLK routine1430149GD601ECH2:CALL144014CC1POP150014E;Rest164014E;Entry parameter:(HL)145014D; Exit parameter: <td></td> <td></td> <td></td> <td></td>				
125013CCD1501HEI1:CALLCONV; Convert to ASCII126013FC34501JMPECH1; Go print it1270142;Noutine ECHO reads one character from the calling1280142; routine and then echoes it back. It is assumed1300142; routine and then echoes it back. It is assumed1310142; that the console is in a full duplex mode.1310142; Entry parameter:None1320142; Entry parameter:None1330142; Stack usage:4 bytes1350142; Stack usage:4 bytes1360142;Stack usage:4 bytes1360142;Stack usage:4 bytes1360145: Alternate entry point to print (A)1390145: Alternate entry point for BLK routine1400146ECH2:CALLCONO1410148ECH2CALLCONO1420149; Alternate entry point for BLK routine1430149CD0601ECH2:CALL144014C:register as 4 hexadecimal digits on the console.1450146;:if parameter:146014E;:if parameter:146014E;:if parameter:147014E;:if parameter:148014E;:if parameter:149::::<	-			to print low order nibble
126013FC34501JMPECH1; Go print it1270142;Routine ECH0 reads one character from the calling1280142;routine and then echoes it back. It is assumed1300142;that the console is in a full duplex mode.1310142;that the console is in a full duplex mode.1310142;Entry parameter:None1330142;Entry parameter:(A) = Character read from1340142;Stack usage:4 bytes1350142;Stack usage:4 bytes1360142;iternate entry point to print (A)1370142(BCH0:CALLCONI1380145C5ECH1:PUSH B1390145C5ECH1:PUSH B1300145C5ECH1:PUSH B1310142fAlternate entry point to print (A)1390145C5ECH1:PUSH B1410148AFMOVC,A1420149;Alternate entry point for ELK routine1430149CD601ECH2:CALL1440148;register as 4 hexadecimal digits on the console.1450140;register as 4 hexadecimal digits on the console.1450148;Entry parameter:(HL) = 4 hex digit number1500148;Entry parameter:Mone1500148;	124 013C	; on consol	e	
125013FC34501JMPECH1; Go print it1270142; Routine ECH0 reads one character from the calling1280142; Routine and then echoes it back. It is assumed1300142; that the console is in a full duplex mode.1310142; Entry parameter:None1320142; Entry parameter:None1340142; Exit parameter:(A) = Character read from1340142; Stack usage:4 bytes1360142;it console keyboard1360142; Stack usage:4 bytes1360145; Alternate entry point to print (A)1390145C5ECH1: PUSH B1390145C5ECH2: CALL1400146E67FANI1410148AFMOV1420149; Alternate entry point for ELK routine1430149CD6001ECH2: CALL1440140;1450140;146014E;147014E; Routine HLPRT prints the contents of the (HL)148014E; register as 4 hexadecimal digits on the console.149014E;150014E; Exit parameter:161014E;177014E; Exit parameter:184014E;195014E; Exit parameter:1960151; Alternate entry point if no CR, LF1960152	125 013C CD1501	HEX1: CAL	L CONV	: Convert to ASCII
<pre>127 0142 128 0142 128 0142 129 0142 130 0142 131 0142 131 0142 131 0142 132 0142 133 0142 133 0142 134 0142 135 0142 135 0142 135 0142 135 0142 137 0142 137 0142 138 0145 138 0145 138 0145 139 0145 130 0145 140 0145 150 0145 150 0145 150 0145 150 0145 151 0145 150 0145 151 0145 150 0145 151 0145 150 0145 151 0145 150 0145 151 014 151 014 151 014 151 014 151 014 151 014 151 014 151 014 151 014 151 014 151 014 151 014 151 014 151 014 151</pre>	126 013F C34501	JMP	ECH1	•
1280142; Routine ECHO reads one character from the calling1290142; routine and then echoes it back. It is assumed1300142; that the console is in a full duplex mode.1310142; Entry parameter: None1320142; Entry parameter: (A) = Character read from1340142; Exit parameter: (A) = Character read from1340142; Stack usage: 4 bytes1360142; Stack usage: 4 bytes1360142; Stack usage: 4 bytes1370142CD03011360145; Alternate entry point to print (A)1390145C51400146ECHO: CALL1390145C51400146ECH2: CALL14101484FMOVC,A; Put character into (C)1420149; Alternate entry point for BLK routine1430149CD0601ECH2: CALL1440140C11450140; Restore (BC)146014E; register as 4 hexadecimal digits on the console.1470148; Entry parameter: (HL) = 4 hex digit number1510148; Entry parameter: None1520142; Stack usage: 10 bytes1540142; Entry parameter: (HL) = 4 hex digit number1550142; Entry parameter: None1550145; Alternate entry point if no CR,LF wanted1560151; Alternate entry point if no CR,LF wanted15		•		, p
1290142; routine and then echoes it back. It is assumed1300142; that the console is in a full duplex mode.1310142; that the console is in a full duplex mode.1320142; Entry parameter:None1330142; Exit parameter:(A) = Character read from1340142; the console keyboard1350142; Stack usage:4 bytes1360142; Stack usage:4 bytes1370142CD0301ECHO: CALLCONI ; Read a character1380145C5ECH1:PUSHB1390145C5ECH1:PUSHB1400146EGTFANITFH ; Strip off parity bit14101484FMOVC.A; Put character into (C)1420149; Alternate entry point for BLK routine:1430149CD0601ECH2:CALLCONO ; Output it144014CC1POPB; Restore (BC)1450149; register as 4 hexadecimal digits on the console.:146014E;istit parameter:(HL) = 4 hex digit number151014E; Exit parameter:None152014E; Exit parameter:None153014E; Exit parameter:None153014E; Exit parameter:None153014E; Exit parameter:None153014E; Exit parameter:None <td< td=""><td></td><td>· Routine F</td><td>CHO réade on</td><td>a observator from the calling</td></td<>		· Routine F	CHO réade on	a observator from the calling
1300142; that the console is in a full duplex mode.1310142;1320142; Entry parameter:None1330142; Exit parameter:(A) = Character read from1340142;the console keyboard1350142; Stack usage:4 bytes1360142;stack usage:4 bytes1360142;Stack usage:4 bytes1370142CD0301ECH0:CALLCONI1390145C5ECH1:PUSHB1390145C5ECH1:PUSHB1390145C5ECH1:PUSHB1400146E67FANI7FH; Stave (BC)1400146E67FANI7FH; Fut parameter into (C)1420149; Alternate entry point for BLK routine1430149CD0601ECH2:CALLCONO144014CC1POPB; Restore (BC)144014C1POPRET144144014E;register as 4 hexadecimal digits on the console.149014E;Entry parameter:(HL) = 4 hex digit number151014E;to be printed152014E;Stack usage:10 bytes154014E;Alternate entry point if no CR.LF wanted155014E;Stack usage:10 bytes154014E <td< td=""><td></td><td></td><td></td><td></td></td<>				
1310142;Entry parameter:None1320142;Entry parameter:None1330142;Exit parameter:(A) = Character read from1340142;the console keyboard1350142;Stack usage:4 bytes1360142;Stack usage:4 bytes1370142CD0301ECH0:CALLCONI1380145;Alternate entry point to print (A)1390145C5ECH1:PUSH1390145C5ECH1:PUSH1400146E67FANI7FH1400146E67FANI7FH14101484FMOVC,A1420149;Alternate entry point for BLK routine1430149CD0601ECH2:CALL144014CC1POPB1450149;ReT146014E;register as 4 hexadecimal digits on the console.147014E;Entry parameter:(HL)148014E;Exit parameter:None150014E;Exit parameter:None151014E;Stack usage:10 bytes154014E;Stack usage:10 bytes155014ECD1E01HLPRT:CALLCRLF1560151;Alternate entry point if no CR,LF wanted1570155	÷			
1320142; Entry parameter:None1330142; Exit parameter:(A) = Character read from1340142;the console keyboard1350142; Stack usage:4 bytes1360142;stack usage:4 bytes1370142CD0301ECH0: CALLCONI; Read a character1380145; Alternate entry point to print (A)1390145C5ECH1:PUSHB; Save (BC)1400146E67FANI7FH; Fut character into (C)1420149; Alternate entry point for BLK routine1430149C0601ECH2:CALLCONO; Output it1440148HFMOVC,A; Put character into (C)1420149; Alternate entry point for BLK routineEXT1440146C1POPB; Restore (BC)1450140C9RETit be printed146014E;register as 4 hexadecimal digits on the console.149014E;Exit parameter:HL146014E;it be printed150014E; Stack usage:10 bytes154014E;Stack usage:10 bytes155014E; Stack usage:10 bytes154014E;Alternate entry point if no CR,LF wanted157014E; Alternate entry point if no CR,LF wanted1580152CD2E01<	-	; that the	console is i	n a full duplex mode.
1330142; Exit parameter:(A) = Character read from1340142;the console keyboard1350142; Stack usage:4 bytes1360142;stack usage:4 bytes1370142(A)ECHO:CALLCONI1380145:Alternate entry point to print (A)1390145C5ECH1:FUSH1300146E67FANI7FH1400146E67FMOVC,A14001484FMOVC,A1400148ECH2:CALLCONO1420149;Alternate entry point for BLK routine1430149CD0601ECH2:CALL1440142C1POPB1450149CD0601ECH2:1440142C1For antice1450149CD0601ECH2:146014E;147014E;146014E150014E;150014E;151014E152014E153014E154014E155014E15601511580152158015215801521580152159RET16001571640157164015716401571640157164	-	;		
1340142;the console keyboard1350142;Stack usage:4 bytes1360142;1370142CD0301ECH0: CALL CONI ; Read a character1380145;Alternate entry point to print (A)1390145C5ECH1: PUSH B ; Save (BC)1400146E67FANI 7FH ; Strip off parity bit14101484FMOV C,A ; Put character into (C)1420149;Alternate entry point for BLK routine1430149CD0601ECH2: CALL CONO ; Output it1440140C1POP B ; Restore (BC)1450149C9RET146014E;147014E; Routine HLPRT prints the contents of the (HL)148014E; register as 4 hexadecimal digits on the console.149014E;150014E; Entry parameter:151014E; Exit parameter:153014E; Stack usage:154014E;155014E; Alternate entry point if no CR,LF wanted1570151; Alternate entry point if no CR,LF wanted1580152CD2E011600156C91610157;1620157164015716401571640157164015716401571640157165; comma, or carriage return). <td< td=""><td></td><td></td><td></td><td></td></td<>				
1340142;the console keyboard1350142;Stack usage:4 bytes1360142;1370142CD0301ECH0: CALL CONI ; Read a character1380145;Alternate entry point to print (A)1390145C5ECH1: PUSH B ; Save (BC)1400146E67FANI 7FH ; Strip off parity bit14101484FMOV C,A ; Put character into (C)1420149;Alternate entry point for BLK routine1430149CD0601ECH2: CALL CONO ; Output it1440140C1POP B ; Restore (BC)1450149C9RET146014E;147014E; Routine HLPRT prints the contents of the (HL)148014E; register as 4 hexadecimal digits on the console.149014E;150014E; Exit parameter:151014E; Exit parameter:153014E; Stack usage:154014E;155014E; Alternate entry point if no CR,LF wanted157151EB1580152CD2E011600156C916401571640157164015716401571640157164015716401571640157164015716401571650157164		; Exit para	meter:	(A) = Character read from
1350142; Stack usage:4 bytes1360142;1370142CD0301ECH0: CALL CONI ; Read a character1380145; Alternate entry point to print (A)1390145C5ECH1: PUSH B1400146E67FANI 7FH ; Strip off parity bit1410148AFMOV C,A ; Put character into (C)1420149; Alternate entry point for BLK routine1430149CD0601ECH2: CALL CONO ; Output it1440140C1POP B ; Restore (BC)145014D C9RET146014E;147014E; Routine HLPRT prints the contents of the (HL)148014E;150014E; Entry parameter: (HL) = 4 hex digit number151014E; Exit parameter: None152014E; Stack usage:10 bytes154014E;155014E; Alternate entry point if no CR,LF wanted1570151 EBHLPR1: CALL CRLF ; Print a (CR,LF)1560152CD2E011580152CD2E011600156C91610157;162015716401571650157164015716401571640157164015716401571640157165015716401571650157164 <td>134 0142</td> <td>;</td> <td>•</td> <td></td>	134 0142	;	•	
136014213701421380145138014513901451400146140014614001461400146140014614001461410148142014914201491430149144014214501491440142145014914601421470142148014214901421440142145014014501401450140146014214701421480142149014214901421490142150014215101421510142152014215301421550142155014215501421550142155014215501421550142155014215501421550142155014215501421560151157158158152159155159152150152150154155154155154155154 <td< td=""><td>135 0142</td><td>: Stack usa</td><td>ge:</td><td></td></td<>	135 0142	: Stack usa	ge:	
1370142CD0301ECHO:CALLCONI; Read a character1380145; Alternate entry point to print (A)1390145C5ECH1:PUSHB; Save (BC)1400146E67FANI7FH; Strip off parity bit14101484FMOVC,A; Put character into (C)1420149; Alternate entry point for BLK routine1430149CD0601ECH2:CALLCONO1440140C1POPB; Restore (BC)1450140C9RET146014E;register as 4 hexadecimal digits on the console.149014E;register as 4 hexadecimal digits on the console.149014E;Entry parameter:(HL) = 4 hex digit number151014E;Exit parameter:None152014E;Exit parameter:None153014E;Stack usage:10 bytes154014E;Alternate entry point if no CR,LF wanted1570151HLPRT:CALLCRLF1580152CD2E01CALLDEPRA1590155EBKCHG; Unswap (HL), (DE)1640157;RET1640157;ReT1640157; comma, or carriage return).If so, a zero is				
1380145; Alternate entry point to print (A)1390145C51400146E67F14101484F14101484F1420149; Alternate entry point for BLK routine1430149CD06011440142C11450149146014E147014E148014E149contact144014C145014E146014E147014E148contact149contact146014E1471014E148contact149contact144014E150contact150contact151014E152contact153contact154contact155contact1560151158contact1580152159contact1590155154contact155contact156contact157contact158contact159contact159contact159contact150contact155contact156contact157contact158contact159contact150contact150contact151<		ECHO. CAL	I CONT	· Read a obanactor
1390145C5ECH1:PUSHB; Sare (BC)1400146E67FANI7FH; Strip off parity bit14101484FMOVC,A; Put character into (C)1420149; Alternate entry point for BLK routine1430149CD0601ECH2:CALLCONO144014CC1POPB; Restore (BC)145014DC9RET146014E;register as 4 hexadecimal digits on the console.149014E;register as 4 hexadecimal digits on the console.149014E;Entry parameter:(HL) = 4 hex digit number150014E;Entry parameter:None152014E;Stack usage:10 bytes154014E;Stack usage:10 bytes155014ECD1E01HLPRT:CALLCRLF1560151;Alternate entry point if no CR,LF wanted1570151EBHLPRA:XCHG1580152CD2E01CALLDEPRA1600156C9RET1610157;1620157;1630157;1640157;1640157;1640157;1640157;1640157;1650157; Routine PCHK reads a character from the console1640157<				
1400146E67FANI7FH; Strip off parity bit14101484FMOVC,A; Put character into (C)1420149; Alternate entry point for BLK routine1430149CD0601ECH2:CALLCONO144014CC1POPB; Restore (BC)145014DC9RET146014E;register as 4 hexadecimal digits on the console.149014E;register as 4 hexadecimal digits on the console.149014E;Entry parameter:(HL) = 4 hex digit number150014E;Exit parameter:None151014E;Stack usage:10 bytes154014E;Stack usage:10 bytes155014ECD2E01HLPRT:CALLCRLF1580152CD2E01CALLDEPRA; Go print (DE)1590155EBXCHG; Unswap (HL), (DE)1600156C9RET1640157;insweicher it is a valid delimiter (space, icomma, or carriage return).If so, a zero is				-
141 0148 4FMOVC,A; Put character into (C)142 0149; Alternate entry point for BLK routine143 0149 CD0601ECH2: CALLCONO; Output it144 014C C1POPB; Restore (BC)145 014D C9RET146 014E;147 014E; Routine HLPRT prints the contents of the (HL)148 014E; register as 4 hexadecimal digits on the console.149 014E;141 014E; Entry parameter:142 014E;151 014E; Exit parameter:152 014E; Stack usage:153 014E; Stack usage:154 014E;155 014E CD1E01HLPRT: CALL157 0151 EBHLPRA: XCHG158 0152 CD2E01CALL159 0155 EBXCHG160 0156 C9RET164 0157;162 0157; Routine PCHK reads a character from the console163 0157; and checks whether it is a valid delimiter (space,164 0157; comma, or carriage return).164 0157;				•
142 0149; Alternate entry point for BLK routine143 0149 CD0601ECH2: CALL CONO ; Output it144 014C C1POP B ; Restore (BC)145 014D C9RET146 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 number151 014E; Exit parameter: None152 014E; Stack usage: 10 bytes154 014E;155 014E CD1E01HLPRT: CALL CRLF ; Print a (CR,LF)156 0151; Alternate entry point if no CR,LF wanted157 0151 EBHLPRA: ICHG ; Swap (HL), (DE)158 0152 CD2E01CALL DEPRA ; Go print (DE)160 0156 C9RET164 0157;162 0157; Routine PCHK reads a character from the console163 0157; and checks whether it is a valid delimiter (space,164 0157; comma, or carriage return). If so, a zero is			•	
143 0149 CD0601ECH2:CALLCONO; Output it144 014C C1POPB; Restore (BC)145 014D C9RET146 014E;147 014E; Routine HLPRT prints the contents of the (HL)148 014E; register as 4 hexadecimal digits on the console.149 014E;140 014E; register as 4 hexadecimal digits on the console.149 014E;150 014E; Entry parameter:151 014E; Exit parameter:152 014E; Exit parameter:153 014E; Stack usage:154 014E;155 014E CD1E01156 0151; Alternate entry point if no CR,LF wanted157 0151 EBHLPRA:158 0152 CD2E01CALL159 0155 EBXCHG160 0156 C9RET161 0157;162 0157; Routine PCHK reads a character from the console163 0157; and checks whether it is a valid delimiter (space,164 0157; comma, or carriage return).164 0157; comma, or carriage return).				; Put character into (C)
143 0149 CD0601ECH2: CALLCONO; Output it144 014C C1POPB; Restore (BC)145 014D C9RET146 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:150 014E; Entry parameter:151 014E; Exit parameter:152 014E; Exit parameter:153 014E; Stack usage:154 014E; Alternate entry point if no CR, LF wanted157 0151EB158 0152 CD2E01CALL158 0152 CD2E01CALL160 0157; Routine PCHK reads a character from the console164 0157; comma, or carriage return).164 0157; comma, or carriage return).	142 0149	; Alternate	entry point	for BLK routine
144014C C1POPB; Restore (BC)145014D C9RET146014E;147014E; Routine HLPRT prints the contents of the (HL)148014E; register as 4 hexadecimal digits on the console.149014E;150014E; Entry parameter:151014E; Exit parameter:152014E; Exit parameter:153014E; Stack usage:154014E;155014E; Stack usage:155014E; Alternate entry point if no CR,LF wanted1570151; Alternate entry point if no CR,LF wanted1570155EBKCHG1590155EB1600156C91640157 <td>143 0149 CD0601</td> <td></td> <td></td> <td></td>	143 0149 CD0601			
145014DC9RET146014E;147014E;148014E;149014E;149014E;149014E;150014E;151014E;152014E;153014E;154014E155014E;155014E;1560151;156015115801521580152158015215901551600156161CALL16101571620157164 <td>144 014C C1</td> <td></td> <td></td> <td></td>	144 014C C1			
146014E;147014E;148014E;149014E;150014E;150014E;151014E;152014E;153014E;154014E;155014E;155014E;155014E;155014E;155014E;155014E;155014E;1560151;1560151;1580152CD2E011580152CD2E011600156C9161015716201571640157165015716501571660157<				, nebudie (10)
147 014E; Routine HLPRT prints the contents of the (HL)148 014E; register as 4 hexadecimal digits on the console.149 014E;149 014E;150 014E; Entry parameter:150 014E; Entry parameter:151 014E;152 014E; Exit parameter:153 014E; Exit parameter:153 014E; Exit parameter:154 014E;155 014E CD1E01HLPRT: CALL CRLF ; Print a (CR,LF)156 0151; Alternate entry point if no CR,LF wanted157 0151 EBHLPRA: XCHG158 0152 CD2E01CALL DEPRA160 0156 C9RET164 0157;162 0157; Routine PCHK reads a character from the console163 0157; and checks whether it is a valid delimiter (space,164 0157; comma, or carriage return).165157164 0157; comma, or carriage return).				
148014E; register as 4 hexadecimal digits on the console.149014E;149014E;150014E;151014E;152014E;152014E;153014E;154014E;155014E;155014EcRLF1560151;1560151;1570151EB1580152CD2E011590155EB1600156C91610157;1620157163015716401571651651661651671651681651691651640157<				
149014E;150014E;151014E;151014E;152014E;152014E;153014E;153014E;154014E;155014ECRLF155014ECRLF1560151;1560151;1560152CD2E011580152CD2E011590155EB1600156C91610157;162015716301571640157164015716401571640157164015716401571640157164015716401571640157164015716401571640157164015716401571640157164015716401571640157165015716401571650157165015716501571650157165015716501571650157165015716501571650157165015716501571650157 <trr< td=""><td></td><td></td><td></td><td></td></trr<>				
150014E; Entry parameter:(HL) = 4 hex digit number151014E; Exit parameter:to be printed152014E; Exit parameter:None153014E; Stack usage:10 bytes154014E;155014E CD1E01HLPRT: CALL CRLF ; Print a (CR,LF)1560151; Alternate entry point if no CR,LF wanted1570151 EBHLPRA: XCHG; Swap (HL), (DE)1580152 CD2E01CALL DEPRA; Go print (DE)1590155 EBXCHG; Unswap (HL), (DE)1600156 C9RET1640157;Routine PCHK reads a character from the console1630157; and checks whether it is a valid delimiter (space,1640157; comma, or carriage return).If so, a zero is		; register	as 4 hexadeci	Imal digits on the console.
151 014E;to be printed152 014E; Exit parameter:None153 014E; Stack usage:10 bytes154 014E;155 014E CD1E01HLPRT: CALL CRLF ; Print a (CR,LF)156 0151; Alternate entry point if no CR,LF wanted157 0151 EBHLPRA: XCHG158 0152 CD2E01CALL DEPRA159 0155 EBXCHG160 0156 C9RET161 0157;162 0157; Routine PCHK reads a character from the console163 0157; comma, or carriage return).164 0157;		;		
152 014E; Exit parameter:None153 014E; Stack usage:10 bytes154 014E;155 014E CD1E01HLPRT: CALL CRLF ; Print a (CR,LF)156 0151; Alternate entry point if no CR,LF wanted157 0151 EBHLPRA: XCHG158 0152 CD2E01CALL DEPRA159 0155 EBXCHG160 0156 C9RET161 0157;162 0157; Routine PCHK reads a character from the console163 0157; and checks whether it is a valid delimiter (space,164 0157; comma, or carriage return).164 0157;		; Entry par	ameter:	(HL) = 4 hex digit number
152 014E; Exit parameter:None153 014E; Stack usage:10 bytes154 014E;155 014E CD1E01HLPRT: CALL CRLF ; Print a (CR,LF)156 0151; Alternate entry point if no CR,LF wanted157 0151 EBHLPRA: XCHG158 0152 CD2E01CALL DEPRA159 0155 EBXCHG160 0156 C9RET161 0157;162 0157; Routine PCHK reads a character from the console163 0157; comma, or carriage return).164 0157; comma, or carriage return).	151 014E	;		
153 014E; Stack usage:10 bytes154 014E;155 014E CD1E01HLPRT: CALL CRLF ; Print a (CR,LF)156 0151; Alternate entry point if no CR,LF wanted157 0151 EBHLPRA: XCHG ; Swap (HL), (DE)158 0152 CD2E01CALL DEPRA ; Go print (DE)159 0155 EBXCHG ; Unswap (HL), (DE)160 0156 C9RET164 0157;164 0157; Routine PCHK reads a character from the console164 0157; comma, or carriage return).164 0157; comma, or carriage return).	152 014E	: Exit para	meter:	-
154 014E;155 014E CD1E01HLPRT: CALL CRLF ; Print a (CR,LF)156 0151; Alternate entry point if no CR,LF wanted157 0151 EBHLPRA: XCHG ; Swap (HL), (DE)158 0152 CD2E01CALL DEPRA ; Go print (DE)159 0155 EBXCHG ; Unswap (HL), (DE)160 0156 C9RET161 0157;162 0157; Routine PCHK reads a character from the console163 0157; and checks whether it is a valid delimiter (space,164 0157; comma, or carriage return).164 0157; comma, or carriage return).	153 014E	• •		
155 014E CD1E01HLPRT: CALLCRLF; Print a (CR,LF)156 0151; Alternate entry point if no CR,LF wanted157 0151 EBHLPRA: XCHG; Swap (HL), (DE)158 0152 CD2E01CALLDEPRA; Go print (DE)159 0155 EBXCHG; Unswap (HL), (DE)160 0156 C9RET;164 0157;Routine PCHK reads a character from the console163 0157; and checks whether it is a valid delimiter (space,164 0157; comma, or carriage return).If so, a zero is			B	
156 0151; Alternate entry point if no CR,LF wanted157 0151 EBHLPRA: XCHG; Swap (HL), (DE)158 0152 CD2E01CALL DEPRA; Go print (DE)159 0155 EBXCHG; Unswap (HL), (DE)160 0156 C9RET164 0157;163 0157; Routine PCHK reads a character from the console164 0157; comma, or carriage return).164 0157; comma, or carriage return).	-			· Detat - (OD tD)
157 0151 EBHLPRA: XCHG; Swap (HL), (DE)158 0152 CD2E01CALL DEPRA; Go print (DE)159 0155 EBXCHG; Unswap (HL), (DE)160 0156 C9RET164 0157;163 0157; Routine PCHK reads a character from the console164 0157; comma, or carriage return).164 0157; comma, or carriage return).				; Print a (CR,LF)
1580152CD2E01CALLDEPRA <th; (de)<="" go="" print="" th="">1590155EBXCHG; Unswap (HL), (DE)1600156C9RET1640157;1620157; RoutinePCHK1630157; and checks whether it is a valid delimiter (space,1640157; comma, or carriage return).If so, a zero is</th;>	-		entry point	
159 0155 EBICHG <th; (de)<="" (hl),="" th="" unswap="">160 0156 C9RET164 0157;162 0157; Routine PCHK reads a character from the console163 0157; and checks whether it is a valid delimiter (space,164 0157; comma, or carriage return).</th;>				
160 0156 C9RET164 0157;162 0157; Routine PCHK reads a character from the console163 0157; and checks whether it is a valid delimiter (space,164 0157; comma, or carriage return).				
160 0156 C9RET161 0157;162 0157;163 0157; Routine PCHK reads a character from the console164 0157; and checks whether it is a valid delimiter (space,; comma, or carriage return).If so, a zero is				; Unswap (HL), (DE)
164 0157;162 0157; Routine PCHK reads a character from the console163 0157; and checks whether it is a valid delimiter (space,164 0157; comma, or carriage return).	160 0156 C9	RET		· · · · · ·
162 0157; Routine PCHK reads a character from the console163 0157; and checks whether it is a valid delimiter (space,164 0157; comma, or carriage return).		;		
163 0157 ; and checks whether it is a valid delimiter (space, 164 0157 ; comma, or carriage return). If so, a zero is		: Routine I	CHK reads	a character from the console
164 0157 ; comma, or carriage return). If so, a zero is				
returned in the status flags. If the character is		, comma, or	carriage ret	urny. II so, a zero is
		, returned 1	n the status	itags. If the character is

and the second second

; a carriage return, the carry bit is set also. If 166 0157 ; it is not a delimiter, a non-zero, no-carry 167 0157 ; indication is required. 168 0157 169 0157 170 0157 ; Entry parameters: None ; Exit Parameters: 171 0157 See description above. ; Stack usage: 172 0157 6 bytes 173 0157 174 0157 CD4201 PCHK: CALL ECHO ;Read a character ; Alternate entry point if CHAR already in (A) 175 015A 1 1 176 015A FE20 PCH2: CPI ; Check for a blank ; Return if (SO) 177 015C C8 RZ 178 015D FE2C CPI ; Check for a comma 179 015F C8 RZ ; Return if (SO) 180 0160 FEOD CPI 'M'-CNTL 181 0162 ; Check for a CAR RET ; Set the carry flag 182 0162 37 STC 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 space or 190 0166 ; comma. If a carriage return is entered, PRM stops 191 0166 ; reading values and returns to the caller. Only ; the last 4 characters of a string are saved; to 192 0166 193 0166 ; correct an error, type until the last four 194 0166 ; characters are correct. The caller may retrieve 195 0166 ; the values by popping them from the stack. 196 0166 ; last-entered character first. 197 0166 198 0166 ; Entry parameter: (C) = number of expected 199 0166 parameters (C) Parameters on stack: 200 0166 ; Exit parameters: 201 0166 If a bad value was entered, ; '????' is printed and 202 0166 ; control transferred to a 203 0166 ; 204 0166 user provided error handler. ; 205 0166 The stack pointer value is ; 206 0166 indeterminate and needs ; 207 0166 to be reset ; Stack usage: 208 0166 4 + 2 = (C) bytes 209 + 0166 ; 210 0166 ; Alternate entry point if only one parameter is ; desired. 211 0166 PARM1: MVI 212 0166 0E01 C,1 213 0168 ; Normal entry point ; Set (HL) = 0214 0168 210000 PRM: LXI Η.Ο ECHO ; Get a character 215 016B CD4201 CALL PRA: ; Save input character 216 016E 47 PRB: MOV B,A ; Check it and CVB 217 016F CD9901 CALL NIBBL 218 0172 DA7E01 PRC JC ; Not hex, see if delim ; Multiply (HL) by 16 219 0175 29 DAD H 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		OHI	INA	, do get next endiacter
	;			
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	U.	; Yes, done
			0000	• •
234 0189 C2C401	PRD:	JNZ	QPRT	; Take error exit if not 0
235 018C OD		DCR	С	; All in?
236 018D C26801		JNZ	PRM	; No, go get another
237 0190 C9		RET		
238 0191	:			
239 0191	• Alter	nate en	try poin	t if only one parameter
240 0191				racter already in (A).
	PRF:	MVI	C,1	lacter arready in (R).
241 0191 0E01	PAF:		•	
242 0193 210000		LXI	Н,О	; Set up (HL)
243 0196 C36E01		JMP	PRB	; Go get rest of parameter
244 0199	;			
245 0199				rips the ASCII zone off a
246 0199	chara	acter in	the (A)	register and verifies that it
247 0199	is a	wolid h	an diait	Te as the binews welve is
		varra no	EX GIGIC	. II SO. LIE DINARV VALUE IS
	· retu	ned to	the low	. If so, the binary value is er half of the A register: the
248 0199	: retu	rned to	the low	er half of the A register; the
248 0199 249 0199	; retur ; upper	rned to half is	the low s set to	er half of the A register; the zero. If not, the carry flag
248 0199 249 0199 250 0199	; retur ; upper	rned to half is	the low s set to	er half of the A register; the
248 0199 249 0199 250 0199 251 0199	; retur ; upper ; is se ;	rned to r half i: et and co	the low s set to ontrol r	er half of the A register; the zero. If not, the carry flag eturned to the caller.
248 0199 249 0199 250 0199 251 0199 252 0199	; retur ; upper ; is se ; ; Entr:	rned to r half is et and co y Parame	the low s set to ontrol r ter:	er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR
248 0199 249 0199 250 0199 251 0199	; retur ; upper ; is se ; ; Entr ; Exit	rned to r half is et and co y Paramet paramet	the low s set to ontrol r ter: ers:	er half of the A register; the zero. If not, the carry flag eturned to the caller.
248 0199 249 0199 250 0199 251 0199 252 0199	; retur ; upper ; is se ; ; Entr ; Exit	rned to r half is et and co y Parame	the low s set to ontrol r ter: ers:	er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 254 0199	; retur ; upper ; is se ; ; Entr ; Exit	rned to r half is et and co y Paramet paramet	the low s set to ontrol r ter: ers:	er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 254 0199 255 0199	; retur ; upper ; is se ; ; Entr; ; Exit ; Stack	rned to r half is et and co y Parame paramet k usage:	the low s set to ontrol r ter: ers:	er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 254 0199 255 0199 255 0199 256 0199 D630	; retur ; upper ; is se ; ; Entr ; Exit	rned to r half is et and co y Parame paramet k usage: SUI	the low s set to ontrol r ter: ers: '0'	er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 254 0199 255 0199 255 0199 256 0199 D630 257 019B D8	; retur ; upper ; is se ; ; Entr; ; Exit ; Stack	rned to r half is et and co y Parame paramet k usage: SUI RC	the low s set to ontrol r ter: ers: "30H '0' = 54, Mc	er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9	; retur ; upper ; is se ; ; Entr; ; Exit ; Stack	rned to r half is et and co y Paramet paramet k usage: SUI RC ADI ==	the low s set to ontrol r ter: ers: '0'	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ;'; Strip off (AF) zone</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8	; retur ; upper ; is se ; ; Entr; ; Exit ; Stack	rned to r half is et and co y Paramet paramet k usage: SUI RC ADI = RC	the low s set to ontrol r ter: ers: "0" = 64, HC sc '0'-'G	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 20 260 019F C606	; retur ; upper ; is se ; ; Entr; ; Exit ; Stack	rned to rhalf is et and co y Paramet paramet k usage: SUI RC ADI = RC ADI	the low s set to ontrol r ter: ers: "0" "0" "0" "G 6	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 0 260 019F C606 261 01A1 F2A701	; retur ; upper ; is se ; ; Entr; ; Exit ; Stack	rned to rhalf is et and co y Paramet paramet k usage: SUI RC ADI FC ADI JP	the low s set to ontrol r ter: ers: "30H '0' "50H '0' "50H '0' "50H '0' "50H '0' "50H '0' "50H '0' "50H '0' Soft '0' '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' '0' '' Soft '0' Soft ' Soft ' Soft '0' Soft ' Soft ' Soft ' '0' ' Soft ' ' ' Soft ' ' ' '	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF)</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 20 260 019F C606	; retur ; upper ; is se ; ; Entr; ; Exit ; Stack	rned to r half is et and co y Paramet paramet k usage: SUI RC ADI = RC ADI	the low s set to ontrol r ter: ers: "0" "0" "0" "G 6	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 0 260 019F C606 261 01A1 F2A701	; retur ; upper ; is se ; ; Entr; ; Exit ; Stack	rned to rhalf is et and co y Paramet paramet k usage: SUI RC ADI FC ADI JP	the low s set to ontrol r ter: ers: "30H '0' "50H '0' "50H '0' "50H '0' "50H '0' "50H '0' "50H '0' "50H '0' Soft '0' '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' Soft '0' '0' '' Soft '0' Soft ' Soft ' Soft '0' Soft ' Soft ' Soft ' '0' ' Soft ' ' ' Soft ' ' ' '	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF)</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 254 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 259 019E D8 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8	; retur ; upper ; is se ; Entry ; Exit ; Stack ; NIBBL:	rned to rhalf is t and co y Paramet paramet k usage: SUI RC ADI FC ADI JP ADI RC	the low s set to ontrol r ter: ers: "0' "b4" HC "0'-'G 6 NIO 7	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 0 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A	; retur ; upper ; is se ; ; Entr; ; Exit ; Stack	rned to rhalf is t and co y Paramet paramet k usage: SUI RC ADI =: RC ADI =: ADI JP ADI RC ADI RC ADI	the low s set to ontrol r ter: ers: "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "50H	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return ; Adjust binary value</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 259 019E D8 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A 265 01A9 B7	; retur ; upper ; is se ; Entry ; Exit ; Stack ; NIBBL:	rned to rhalf is t and co y Paramet paramet k usage: SUI RC ADI RC ADI JP ADI RC ADI RC ADI QP ADI RC	the low s set to ontrol r ter: ers: "0' "b4" HC "0'-'G 6 NIO 7	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A 265 01A9 B7 266 01AA C9	; retur ; upper ; is se ; Entry ; Exit ; Stach ; NIBBL: NIO:	rned to rhalf is t and co y Paramet paramet k usage: SUI RC ADI =: RC ADI =: ADI JP ADI RC ADI RC ADI	the low s set to ontrol r ter: ers: "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "50H	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return ; Adjust binary value</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 259 019E D8 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A 265 01A9 B7 266 01AA C9 267 01AB	; retur ; upper ; is se ; Entry ; Exit ; Stach ; NIBBL: NIO: ;	rned to rhalf is et and co y Parame paramet k usage: SUI RC ADI == RC ADI JP ADI RC ADI RC ADI RC ADI RC ADI RC ADI RC ADI RC ADI RC ADI RC ADI RC ADI RC ADI RC ADI RC	the low s set to ontrol r ter: ers: "30H "0"-"G 6 NIO 7 10 A	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return ; Adjust binary value ; Reset carry bit</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 254 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 20 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A 265 01A9 B7 266 01AA C9 267 01AB 268 01AB	; retur ; upper ; is se ; Entry ; Exit ; Stack ; NIBBL: NIBBL:	rned to rhalf is t and co y Parame paramet k usage: SUI RC ADI = RC ADI JP ADI RC RC ADI RC RC ADI RC RC ADI RC RC ADI RC RC RC ADI RC RC RC ADI RC RC RC RC RC RC RC RC RC RC RC RC RC	the low s set to ontrol r ter: ers: """"""""""""""""""""""""""""""""""""	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return ; Adjust binary value ; Reset carry bit</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A 265 01A9 B7 266 01AA C9 267 01AB 268 01AB 269 01AB	; retur ; upper ; is se ; Entry ; Exit ; Stack ; NIBBL: NIBBL:	rned to half is t and co y Paramet paramet k usage: SUI RC ADI = RC ADI JP ADI RC ADI RC ADI RC ADI RC ADI RC ADI RC ADI RC ADI RC ADI SUI RC ADI RC ADI RC ADI RC ADI RC ADI SUI RC ADI RC ADI SUI SUI RC ADI SUI RC ADI SUI SUI RC ADI SUI SUI SUI SUI SUI RC ADI SUI SUI SUI SUI SUI SUI SUI SUI SUI SU	the low s set to ontrol r ter: ers: "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "0" "50H "50" "50"	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return ; Adjust binary value ; Reset carry bit</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 0 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A 265 01A9 B7 266 01AA C9 267 01AB 268 01AB 269 01AB 269 01AB	; retur ; upper ; is se ; Entry ; Exit ; Stack ; NIBBL: NIBBL: NIC: ; Rout; ; Conso ; LF r	rned to half is t and co y Paramet paramet k usage: SUI RC ADI I RC ADI JP ADI RC ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI BC ADI JP ADI RC ADI JP ADI RC ADI BC ADI JP ADI RC ADI BC ADI JP ADI RC ADI BC BC ADI BC BC ADI BC BC ADI BC BC BC BC BC BC BC BC BC BC BC BC BC	the low s set to ontrol r ter: ers: "0"-"G 6 NIO 7 10 A D prints Dependin printed	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return ; Adjust binary value ; Reset carry bit</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A 265 01A9 B7 266 01AA C9 267 01AB 268 01AB 269 01AB	; retur ; upper ; is se ; Entr; ; Exit ; Stach ; NIBBL: NIBBL: NIC: ; ; Rout: ; conso ; LF m ; messa	rned to half is t and co y Paramet paramet k usage: SUI RC ADI I RC ADI JP ADI RC ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI BC BC ADI BC ADI BC BC ADI BC ADI BC ADI BC ADI BC ADI BC ADI BC BC ADI BC BC BC ADI BC BC BC BC BC BC BC BC BC BC BC BC BC	the low s set to ontrol r ter: ers: "0"-"G 6 NIO 7 10 A D prints Dependin printed delimit	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return ; Adjust binary value ; Reset carry bit a character string on the g on the entry point, a CR and first. Three forms of ers are accepted: Bit 7=1 in</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 0 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A 265 01A9 B7 266 01AA C9 267 01AB 268 01AB 269 01AB 269 01AB	; retur ; upper ; is se ; Entr; ; Exit ; Stach ; NIBBL: NIBBL: NIC: ; ; Rout: ; conso ; LF m ; messa	rned to half is t and co y Paramet paramet k usage: SUI RC ADI I RC ADI JP ADI RC ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI JP ADI RC ADI BC BC ADI BC ADI BC BC ADI BC ADI BC ADI BC ADI BC ADI BC ADI BC BC ADI BC BC BC ADI BC BC BC BC BC BC BC BC BC BC BC BC BC	the low s set to ontrol r ter: ers: "0"-"G 6 NIO 7 10 A D prints Dependin printed delimit	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return ; Adjust binary value ; Reset carry bit a character string on the g on the entry point, a CR and first. Three forms of ers are accepted: Bit 7=1 in</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 259 019E D8 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A 265 01A9 B7 266 01AA C9 267 01AB 268 01AB 269 01AB 270 01AB 270 01AB 271 01AB	; retur ; upper ; is se ; Entry ; Exit ; Stach ; NIBBL: NIBBL: NIC: ; ; Rout; ; conse ; LF r ; messa ; last	rned to half is t and co y Parame paramet k usage: SUI RC ADI = RC ADI JP ADI RC RC ADI RC RC ADI RC RC ADI RC RC ADI RC RC RC RC RC RC RC RC RC RC RC RC RC	the low s set to ontrol r ter: ers: """"""""""""""""""""""""""""""""""""	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return ; Adjust binary value ; Reset carry bit a character string on the g on the entry point, a CR and first. Three forms of ers are accepted: Bit 7=1 in output; ASCII ETX (CNTRL C)</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 254 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 20 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A 265 01A9 B7 266 01AA C9 267 01AB 268 01AB 269 01AB 270 01AB 271 01AB 271 01AB 273 01AB	; retur ; upper ; is se ; Entry ; Exit ; Stach ; NIBBL: NIBBL: NIC: ; ; Rout; ; conso ; LF m ; messa ; last ; follo	rned to half is t and co y Parame paramet k usage: SUI RC ADI = RC ADI JP ADI RC RC ADI RC RC ADI RC RC RC RC RC RC RC RC RC RC RC RC RC	the low s set to ontrol r ter: ers: """"""""""""""""""""""""""""""""""""	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return ; Adjust binary value ; Reset carry bit a character string on the g on the entry point, a CR and first. Three forms of ers are accepted: Bit 7=1 in output; ASCII ETX (CNTRL C) character; or a user-specified</pre>
248 0199 249 0199 250 0199 251 0199 252 0199 253 0199 253 0199 255 0199 255 0199 256 0199 D630 257 019B D8 258 019C C6E9 259 019E D8 259 019E D8 260 019F C606 261 01A1 F2A701 262 01A4 C607 263 01A6 D8 264 01A7 C60A 265 01A9 B7 266 01AA C9 267 01AB 268 01AB 269 01AB 270 01AB 270 01AB 271 01AB	; retur ; upper ; is se ; Entry ; Exit ; Stack ; NIBBL: NIBBL: NIC: ; ; Rout; ; conso ; LF m ; messa ; last ; follo ; delim	rned to half is t and co y Parame paramet k usage: SUI RC ADI I RC ADI I RC ADI RC RC ADI RC RC ADI RC RC ADI RC RC ADI RC RC ADI RC RC RC RC RC RC RC RC RC RC RC RC RC	the low s set to ontrol r ter: ers: "0' "6" NIO 7 10 A D prints Dependin printed delimit er to be he last llowing	<pre>er half of the A register; the zero. If not, the carry flag eturned to the caller. (A) = ASCII CHAR See description above None ; Strip off 0-9 Zone ; Invalid value RET ; Strip off (AF) zone ; Invalid value RET ; Sort out in-between values ; Jump if (AF) ; Insure it is 0-9 ; wasn't: Return ; Adjust binary value ; Reset carry bit a character string on the g on the entry point, a CR and first. Three forms of ers are accepted: Bit 7=1 in output; ASCII ETX (CNTRL C)</pre>

276 01AB	; on en	itry to H	PRTA.	
277 01AB	;			
278 01AB	; Entry	Paramet	cers:	
279 01AB 280 01AB	;			(B) = ETX delimiter (See
281 01AB	; . Evit	Paramete		description above.)
282 01AB	-	usage:		None - (HL) is altered 12 bytes MAX
283 01AB	; DUACE	usage.		12 Dyles MAX
	; Entry	point f	for CR.LF	(will not work with user
285 01AB	; defin	ed ETX d	lelimiter	·).
286 01AB CD1E01		CALL	CRLF	
287 01AE	; Entry	point f	for No. C	CR,LF and a bit 7 or ASCII
	; ETX D			
289 01AE C5 290 01AF 0603	PRTWA:	PUSH	B	; Save (BC) ; Get an ASCII ETX
291 01B1 CDB601		CALL	8,3 🖗 DDTA	; Get an ASCII ETX ; Print message
292 01B4 <b>C5C</b>		POP	R	; Frint message ; Restore (BC)
293 01B5 C9		RET	<b>Д</b> ,	, nescore (bc)
294 01B6	;			
295 01B6		point f	or user	defined ETX delimiter
296 01 <b>B6 78</b>	PRTA:	MOV		; Put ETX in A
297 01B7 4E		MOV		; Get next character
298 01B8 B9		CMP	С	; EOM?
299 01B9 C8 300 01BA CD0601		RZ	0010	; Yes, done
301 01BD 79				; No, output it ; Retrieve CHAR
302 01BE 23		TNY	н,С	; Point to next CHAR
303 01BF B7		ORA	A	; See if bit 7 is set
304 01C0 F2B601		JP	PRTA	; No, continue
305 01C3 C9		RET		
306 01C4	;			
307 01C4	; Routi	ne QPRT	prints "	????" and transfers control
308 01C4				r- recovery routine. (SP) is
309 01C4 310 01C4	; indet	erminate	e on exit	•
311 01C4 21CD01	, QPRT:	LXI	H OMSG	; Message address
			PRTWA	; Print it
313 01CA C30C01		JMP	ERR	; Go to error recovery
314 01CD	;			
315 01CD 3F3F3FBF	QMSG:	DB	'???','	?'+80H
316 01D1	;			
317 01D1	;			
318 01D1	; Hardw			can be divided into 3 stages:
319 01D1 320 01D1	;		ault dete ault isol	
321 01D1	· ·		ault corr	
322 01D1	, These			te the first stage only. See
323 01D1				guidelines for the second
324 01D1				cond step is completed, fault
325 01D1				no trouble.
326 01D1	;			
327 01D1	;			
328 01D1	; SUBRO	UTINES H	FOR THE M	MEMORY DIAGNOSTICS
329 01D1 330 01D1	; . LThan	a had me	mony oci	lis datastad this sautist
	, when	a vau me	amon'y del	l is detected, this routine

•				
331 01D1	; is call	led to pi	rint the '	bad address, bad data, test
332 01D1				(in that order). With this
333 01D1				isolation process can be
334 01D1	; conduct			
	; conduc	veu.		
335 01D1	;			
336 01D1 CD2B01	ADPRT:	CALL	DEPRT ;	Print bad address
337 01D4 CD0F01	4	CALL	BLK ;	Print a blank
338 01D7 78	1	MOV	A,B ;	Get a bad data
339 01D8 C3E001			ADPRB	
340 01DB		0111		
-	,			· · · · · · · · · · · · · · · · · · ·
341 01DB	•		y point w	hen bad address is
342 01DB	; meanin			•
343 01DB F5	ADPRA:	PUSH	PSW	
344 01DC CD1E01	• (	CALL	CRLF ;	Do a (CR,LF)
345 01DF F1			PSW	
346 01E0 CD3301				Drint had data
			•	Print bad data
347 01E3 CDOF01			BLK	
348 01E6 CD0F01		CALL		
349 01E9 CD5101		CALL	HLPRA ;	Print test address
350 01EC CD0F01		CALL	BLK	
351 01EF 79				Get test data
352 01F0 C33301				Print it
		JHF	ПБАС ;	
353 01F3	;			· · · · · · · · · · · · · · · · · · ·
354 01F3				console status to see if a
355 01F3	; charact	ter has l	been type	d in. If so, it checks to
356 01F3	; see if	f it is	s an ASCI	I ETX (CNTRL C). If so, it
357 01F3	: types a	an "ABOR"	r" message	e and returns control to
358 01F3	the ca	lling rou	utine.	
359 01F3	•			
360 01F3 CD0901	, BREAK:	CALL		Changeton weiting?
				Character waiting? No, return
361 01F6 C8				NO. PELUTD
		RZ		
362 01F7 CD0301	!	CALL	CONI ;	Yes, get it
362 01F7 CD0301 363 01FA FE03	!	CALL		
	!	CALL	CONI ; 'C'-CNTL	
363 01FA FE03 364 01FC		CALL CPI	CONI ; 'C'-CNTL ;	Yes, get it See if Cntl C
363 01FA FE03 364 01FC 365 01FC C0		CALL CPI RNZ	CONI ; 'C'-CNTL ; ;	Yes, get it See if Cntl C No, return
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702		CALL CPI RNZ LXI	CONI ; 'C'-CNTL ; H,ABMSG ;	Yes, get it See if Cntl C No, return Print out the
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01	. :	CALL CPI RNZ LXI CALL	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00	. :	CALL CPI RNZ LXI CALL	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK-	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206		CALL CPI RNZ LXI CALL LXI	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK-	Yes, get it See if Cntl C No, return Print out the 'ABORT' message
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00		CALL CPI RNZ LXI CALL	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206		CALL CPI RNZ LXI CALL LXI	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207	; ;	CALL CPI RNZ LXI CALL LXI RET	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52	; Abmsg:	CALL CPI RNZ LXI CALL LXI RET	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52 020B D4	; Abmsg:	CALL CPI RNZ LXI CALL LXI RET DB	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52 020B D4 373 020C	; Abmsg:	CALL CPI RNZ LXI CALL LXI RET DB	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; 'ABOR','T	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52 020B D4 373 020C 374 020C	ABMSG:	CALL CPI RNZ LXI CALL LXI RET DB	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; 'ABOR','T reads in	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H the desired test block size
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52 020B D4 373 020C 374 020C 375 020C	ABMSG:	CALL CPI RNZ LXI CALL LXI RET DB e PARM n lock bas	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; 'ABOR','T reads in ; se addre	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52 020B D4 373 020C 374 020C	ABMSG:	CALL CPI RNZ LXI CALL LXI RET DB e PARM n lock bas	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; 'ABOR','T reads in ; se addre	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H the desired test block size
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52 020B D4 373 020C 374 020C 375 020C	ABMSG:	CALL CPI RNZ LXI CALL LXI RET DB e PARM n lock bas	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; 'ABOR','T reads in ; se addre	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H the desired test block size
363       01FA       FE03         364       01FC         365       01FC       CO         366       01FD       210702         367       0200       CDAB01         368       0203       313E00         369       0206       C9         371       0207       41424F52         020B       D4         373       020C         374       020C         376       020C         377       020C	ABMSG: Routing and b pushed	CALL CPI RNZ LXI CALL LXI RET DB e PARM n lock bas onto the	CONI ; 'C'-CNTL H,ABMSG ; PRTWD ; SP,STACK- ; 'ABOR','T reads in se addre e stack.	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H the desired test block size ss. Both parameters are
363       01FA       FE03         364       01FC         365       01FC       CO         366       01FD       210702         367       0200       CDAB01         368       0203       313E00         369       0206       G         370       0206       C9         371       0207       41424F52         020B       D4         373       020C         374       020C         375       020C         376       020C         377       020C         378       020C	ABMSG: ; Routine; and b; ; pushed ; PARM:	CALL CPI RNZ LXI CALL LXI RET DB e PARM n lock bas onto the CALL	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; 'ABOR','T reads in ; se addres e stack. PRTWA ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H the desired test block size ss. Both parameters are Print celler's name
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52 020B D4 373 020C 374 020C 375 020C 376 020C 377 020C 378 020C CDAE01 379 020F 212402	ABMSG: Routine and b pushed FARM:	CALL CPI RNZ LXI CALL LXI RET DB e PARM n lock bas onto the CALL LXI	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; 'ABOR','T reads in se addrese stack. PRTWA ; H,BZMSG ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H the desired test block size ss. Both parameters are
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52 020B D4 373 020C 374 020C 375 020C 376 020C 376 020C 377 020C 378 020C CDAE01 379 020F 212402 380 0212 CDAB01	ABMSG: Routine and b pushed FARM:	CALL CPI RNZ LXI CALL LXI RET DB OB OB CALL CALL CALL CALL	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; 'ABOR','T reads in ; se addres e stack. PRTWA ; H,BZMSG ; PRTWD	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H the desired test block size ss. Both parameters are Print caller's name Print BLOCK SIZE message
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52 020B D4 373 020C 374 020C 375 020C 376 020C 376 020C 377 020C 378 020C CDAE01 379 020F 212402 380 0212 CDAB01 381 0215 CD6601	ABMSG: Routine and b pushed PARM:	CALL CPI RNZ LXI CALL LXI RET DB e PARM r lock bas onto the CALL CALL CALL CALL CALL	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; SP,STACK- ; 'ABOR','T reads in ; reads in ; reads in ; se addres e stack. PRTWA ; H,BZMSG ; PRTWD PARM1 ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H the desired test block size ss. Both parameters are Print celler's name Print BLOCK SIZE message Get block size
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52 020B D4 373 020C 374 020C 375 020C 376 020C 376 020C 377 020C 378 020C CDAE01 379 020F 212402 380 0212 CDAB01 381 0215 CD6601 382 0218 E1	ABMSG: Routine; and b pushed PARM:	CALL CPI RNZ LXI CALL LXI RET DB e PARM n lock bas onto the CALL LXI CALL CALL POP	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; SP,STACK- ; 'ABOR','T reads in ; reads in ; reads in ; se addres e stack. PRTWA ; H,BZMSG ; PRTWD PARM1 ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H the desired test block size ss. Both parameters are Print caller's name Print BLOCK SIZE message
363       01FA       FE03         364       01FC         365       01FC       CO         366       01FD       210702         367       0200       CDAB01         368       0203       313E00         369       0206       C9         371       0207       41424F52         020B       D4         373       020C         374       020C         376       020C         377       020C         378       020C         380       0212         CDAB01       381         381       0215         CD6601       382         383       0219         83       0219	ABMSG: Routine; and b pushed PARM:	CALL CPI RNZ LXI CALL LXI RET DB e PARM r lock bas onto the CALL CALL CALL CALL CALL	CONI ; 'C'-CNTL ; H,ABMSG ; PRTWD ; SP,STACK- ; SP,STACK- ; 'ABOR','T reads in ; reads in ; reads in ; se addres e stack. PRTWA ; H,BZMSG ; PRTWD PARM1 ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H the desired test block size ss. Both parameters are Print celler's name Print BLOCK SIZE message Get block size
363 01FA FE03 364 01FC 365 01FC C0 366 01FD 210702 367 0200 CDAB01 368 0203 313E00 369 0206 370 0206 C9 371 0207 372 0207 41424F52 020B D4 373 020C 374 020C 375 020C 376 020C 376 020C 377 020C 378 020C CDAE01 379 020F 212402 380 0212 CDAB01 381 0215 CD6601 382 0218 E1	ABMSG: ; Routine ; and b ; pushed ; PARM:	CALL CPI RNZ LXI CALL LXI RET DB e PARM n lock bas onto the CALL LXI CALL CALL CALL CALL POP XTHL	CONI ; 'C'-CNTL H,ABMSG ; PRTWD ; SP,STACK- ; 'ABOR','T reads in se addres e stack. PRTWA ; H,BZMSG ; PRTWD PARM1 ; H ;	Yes, get it See if Cntl C No, return Print out the 'ABORT' message 2 Reset the stack Return to exec '+80H the desired test block size ss. Both parameters are Print celler's name Print BLOCK SIZE message Get block size

385	021B	213002	PARMA:	LXI	H.BAMSG	; Print BASE ADDRESS
		CDAB01		CALL	ספייעה	; message
		C36601				
	0224	030001		JHF	PARMI	; Get it and return
			;	~ ~		
309		424C4F43	BZMSG:	DB	'BLOCK S	SIZE:',' '+80H
		4B205349				
	022C	5A453AA0				
390			BAMSG:	DB	'BASE'	
		-	ADMSG:			S:',' '+80H
		52455353	ADIIOU.		ADDIES	5. , +00n
	023C	3440				
392	023E		;			
393	023E		; Routir	he MADT r	erforms a	a "Walking Bit" test on both
394	023E					lines of a 2114 pair at the
	023E					
	023E					zeros all cells in the
	-					n ensures that they are all
	023E		; zero.	It te	ests eac	h 1K section separately.
	023E		; Detect	ted erro	ors are l	ogged on the console as they
399	023E		; occur.			
400	023E		;	-		
401	J23E		· Tho h	and addm	an uhon	asked for, must be on a 1K
	023E		• hours	ase adure	is when	asked for, must be on a fr
	023E					be rejected and another
			; addres	ss asked	for.	
	023E		;			
	023E		; The of	perator (	ean abort	, the test at any time by
406	023E					) should too many errors be
407	023E					he test to complete will
408	0236		: ongun	eunabe a	t cich a	
	023E 023E		; ensure	e adequat	ce data f	or thorough fault isolation.
409	023E		;	•		or thorough fault isolation.
409 410	023E 023E		; ; Withou	ut error:	s, this d	or thorough fault isolation. Hiagnostic tests a 1K cell in
409 410 411	023E 023E 023E		; ; Withou	ut error:		or thorough fault isolation. Hiagnostic tests a 1K cell in
409 410 411 412	023E 023E 023E 023E		; ; Withou	ut error:	s, this d	or thorough fault isolation. Hiagnostic tests a 1K cell in
409 410 411 412 413	023E 023E 023E 023E 023E	217F02	; ; Withou	ut error: ximately	s, this d 2 second	for thorough fault isolation. Hiagnostic tests a 1K cell in Is.
409 410 411 412 413	023E 023E 023E 023E 023E		; ; Withou ; appro: ; MADT:	ut error: ximately LXI	s, this d 2 second H,WBMSG	For thorough fault isolation. Hiagnostic tests a 1K cell in Hs. ; Sign on
409 410 411 412 413 413	023E 023E 023E 023E 023E 023E 0241	CD0C02	; ; Withou ; appro: ; MADT:	ut errors ximately LXI CALL	s, this d 2 second H,WBMSG PARM	<pre>or thorough fault isolation. liagnostic tests a 1K cell in ls. ; Sign on ; Get parameters</pre>
409 410 411 412 413 414 415	023E 023E 023E 023E 023E 0241 0244	CD0C02 E1	; ; Withou ; appro: ; MADT: MADTA:	ut error: ximately LXI CALL POP	s, this d 2 second H,WBMSG PARM H	<pre>or thorough fault isolation. liagnostic tests a 1K cell in ls. ; Sign on ; Get parameters ; Retrieve BASE ADDRESS</pre>
409 410 411 412 413 414 415 416	023E 023E 023E 023E 023E 023E 0241 0244 0245	CD0C02 E1 D1	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP	s, this d 2 second H,WBMSG PARM H D	<pre>For thorough fault isolation. Hiagnostic tests a 1K cell in Is. ; Sign on ; Get parameters ; Retrieve BASE ADDRESS ; Retrieve BLOCK SIZE</pre>
409 410 411 412 413 414 415 416 417	023E 023E 023E 023E 023E 023E 0241 0244 0245 0246	CD0C02 E1 D1 7C	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV	s, this d 2 second H,WBMSG PARM H D A,H	<pre>or thorough fault isolation. liagnostic tests a 1K cell in ls. ; Sign on ; Get parameters ; Retrieve BASE ADDRESS</pre>
409 410 411 412 413 414 415 416 417 418	023E 023E 023E 023E 023E 0241 0244 0245 0246 0247	CD0C02 E1 D1 7C E603	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI	s, this d 2 second H,WBMSG PARM H D A,H 3	<pre>For thorough fault isolation. Hiagnostic tests a 1K cell in Is. ; Sign on ; Get parameters ; Retrieve BASE ADDRESS ; Retrieve BLOCK SIZE</pre>
409 410 411 412 413 414 415 416 417 418 419	023E 023E 023E 023E 023E 0241 0244 0245 0246 0247 0249	CD0C02 E1 D1 7C E603 B5	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA	s, this d 2 second H,WBMSG PARM H D A,H	<pre>For thorough fault isolation. Hiagnostic tests a 1K cell in Is. ; Sign on ; Get parameters ; Retrieve BASE ADDRESS ; Retrieve BLOCK SIZE</pre>
409 410 411 412 413 414 415 416 417 418 419	023E 023E 023E 023E 023E 0241 0244 0245 0246 0247 0249	CD0C02 E1 D1 7C E603	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI	s, this d 2 second H,WBMSG PARM H D A,H 3 L	<pre>For thorough fault isolation. Hiagnostic tests a 1K cell in Is. ; Sign on ; Get parameters ; Retrieve BASE ADDRESS ; Retrieve BLOCK SIZE</pre>
409 410 411 412 413 415 415 416 417 418 419 420	023E 023E 023E 023E 023E 0241 0244 0245 0246 0247 0249	CD0C02 E1 D1 7C E603 B5 CA6002	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB	<pre>For thorough fault isolation. liagnostic tests a 1K cell in ls. ; Sign on ; Get parameters ; Retrieve BASE ADDRESS ; Retrieve BLOCK SIZE ; Test for 1K boundary ; OK, jump</pre>
409 410 411 412 413 414 415 416 417 418 420 421	023E 023E 023E 023E 023E 0241 0244 0245 0246 0245 0246 0247 0249 024A 024D	CD0C02 E1 D1 7C E603 B5 CA6002 D5	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D	<pre>For thorough fault isolation. Hiagnostic tests a 1K cell in Is. ; Sign on ; Get parameters ; Retrieve BASE ADDRESS ; Retrieve BLOCK SIZE ; Test for 1K boundary ; OK, jump ; Save block size</pre>
409 410 411 412 413 414 415 416 417 418 420 421 422	023E 023E 023E 023E 023E 0241 0244 0245 0246 0247 0249 024A 024D 024E	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG	<pre>For thorough fault isolation. liagnostic tests a 1K cell in ls. ; Sign on ; Get parameters ; Retrieve BASE ADDRESS ; Retrieve BLOCK SIZE ; Test for 1K boundary ; OK, jump</pre>
409 410 411 412 413 414 415 416 417 418 420 420 422 423	023E 023E 023E 023E 023E 0241 0244 0245 0244 0245 0246 0247 0249 024A 024D 024E 0251	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD	<pre>For thorough fault isolation. Hiagnostic tests a 1K cell in Is. ; Sign on ; Get parameters ; Retrieve BASE ADDRESS ; Retrieve BLOCK SIZE ; Test for 1K boundary ; OK, jump ; Save block size</pre>
409 410 411 412 413 415 415 415 415 417 418 420 422 422 424	023E 023E 023E 023E 023E 0241 0244 0245 0245 0246 0247 0249 024A 024D 024E 0251 0254	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG	<pre>For thorough fault isolation. Hiagnostic tests a 1K cell in Is. ; Sign on ; Get parameters ; Retrieve BASE ADDRESS ; Retrieve BLOCK SIZE ; Test for 1K boundary ; OK, jump ; Save block size</pre>
409 410 411 412 413 415 415 415 415 415 416 422 422 422 422 422 422 422 425	023E 023E 023E 023E 023E 0241 0244 0245 0246 0247 0249 0244 0249 024A 024D 024E 0251 0257	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI CALL	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWA	<pre>Por thorough fault isolation. Provide the standard state of the standard state of the state</pre>
409 410 411 412 413 415 415 415 415 415 422 422 422 422 422 422 422 422 422 42	023E 023E 023E 023E 023E 0241 0244 0245 0246 0247 0246 0247 0249 0244 0245 0244 0251 0254 0257 025A	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01 CD1B02	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWA PARMA	<pre>For thorough fault isolation. Final Sign on Final Get parameters Final Fi</pre>
409 410 411 412 413 415 415 415 415 415 422 422 422 422 422 422 422 422 422 42	023E 023E 023E 023E 023E 0241 0244 0245 0246 0247 0246 0247 0249 0244 0245 0244 0251 0254 0257 025A	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI CALL	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWA PARMA	<pre>Por thorough fault isolation. Provide the standard state of the standard state of the state</pre>
409 410 411 412 413 415 415 415 417 419 422 422 422 422 422 422 422 422 422 42	023E 023E 023E 023E 023E 0241 0244 0245 0246 0247 0246 0247 0249 0244 0245 0244 0251 0254 0257 025A	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01 CD1B02	; ; Withou ; appro: ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI CALL CALL	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWA PARMA	<pre>For thorough fault isolation. Final Sign on Final Get parameters Final Fi</pre>
409 410 411 412 413 415 415 415 415 415 415 422 422 422 422 422 422 422 422 422 42	023E 023E 023E 023E 023E 0241 0244 0245 0244 0245 0246 0247 0249 0244 0249 0244 0245 0244 0251 0254 0257 0250 0260	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01 CD1B02 C34402	; Withou ; approx ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI CALL LXI CALL JMP	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWA PARMA MADTA	<pre>Por thorough fault isolation. Provide the set of t</pre>
409 411 412 411 412 415 415 415 415 415 415 415 415 415 415	023E 023E 023E 023E 023E 0241 0244 0245 0244 0245 0246 0247 0249 0244 0245 0244 0245 0244 0245 0251 0257 0255 0250 0260 0260	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01 CD1B02 C34402 CD9902	; Withou ; appros ; MADT: MADTA: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI CALL JMP CALL	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWA PARMA MADTA ZTBK	<pre>Por thorough fault isolation. Provide the standard state of the standard state of the state</pre>
409 411 412 411 412 415 415 415 415 415 415 422 23 45 67 890	023E 023E 023E 023E 023E 0241 0245 0245 0246 0247 0249 0244 0245 0244 0245 0244 0245 0244 0245 0257 0257 0255 0250 0260 0263	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01 CD1B02 C34402 CD9902 D5	; Withou ; approx ; MADT: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI CALL LXI CALL JMP CALL PUSH	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWA PARMA MADTA ZTBK D	<pre>Por thorough fault isolation. Provide the standard state of the standard state of the state</pre>
409 4112 4112 4112 4115 4115 41112 4115 4112 4122 412	023E 023E 023E 023E 023E 0241 0244 0245 0244 0245 0246 0247 0244 0245 0244 0245 0257 0257 0250 0260 0263 0264	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01 CD1B02 C34402 CD9902 D5 3E04	; Withou ; appros ; MADT: MADTA: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI CALL LXI CALL JMP CALL PUSH MVI	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWD H,BAMSG PRTWA PARMA MADTA ZTBK D A,4	<pre>Por thorough fault isolation. Prove thorough fault isolation. Prove the state of the state</pre>
409 4112 44112 444444444444444444444444444	023E 023E 023E 023E 023E 0241 0244 0245 0244 0245 0246 0247 0244 0245 0244 0245 0244 0245 0257 0255 0250 0260 0263 0266	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01 CD1B02 C34402 CD9902 D5 3E04 BA	; Withou ; appros ; MADT: MADTA: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI CALL LXI CALL JMP CALL PUSH MVI CMP	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWD H,BAMSG PRTWA PARMA MADTA ZTBK D A,4 D	<pre>Por thorough fault isolation. Provide the standard state of the standard state of the standard state st</pre>
409 4112 4112 4112 4112 4112 4112 4112 411	023E 023E 023E 023E 023E 0244 0245 0244 0245 0244 0245 0247 0244 0247 0244 0245 0244 0257 0257 0255 0260 0263 0266 0267	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01 CD1B02 C34402 CD9902 D5 3E04 BA F26B02	; Withou ; appros ; MADT: MADTA: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI CALL LXI CALL JMP CALL PUSH MVI CMP JP	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWA PARMA MADTA ZTBK D A,4 D MADTD	<pre>Por thorough fault isolation. Provide the state of t</pre>
400012345678901234 41123456789012345678901234 433334	023E 023E 023E 023E 023E 0241 0244 0245 0244 0245 0246 0247 0244 0244 0245 0244 0245 0254 0257 0255 0255 0260 0263 0266 0267 026A	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01 CD1B02 C34402 CD9902 D5 3E04 BA F26B02 57	; Withou ; appros ; MADT: MADTA: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI CALL LXI CALL JMP CALL PUSH MVI CMP	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWD H,BAMSG PRTWA PARMA MADTA ZTBK D A,4 D	<pre>Por thorough fault isolation. Provide the state of t</pre>
400012345678901234 41123456789012345678901234 433334	023E 023E 023E 023E 023E 0241 0244 0245 0244 0245 0246 0247 0244 0244 0245 0244 0245 0254 0257 0255 0255 0260 0263 0266 0267 026A	CD0C02 E1 D1 7C E603 B5 CA6002 D5 217B02 CDAB01 213002 CDAE01 CD1B02 C34402 CD9902 D5 3E04 BA F26B02	; Withou ; appros ; MADT: MADTA: MADTA:	ut errors ximately LXI CALL POP POP MOV ANI ORA JZ PUSH LXI CALL LXI CALL LXI CALL JMP CALL PUSH MVI CMP JP	s, this d 2 second H,WBMSG PARM H D A,H 3 L MADTB D H,BEMSG PRTWD H,BAMSG PRTWA PARMA MADTA ZTBK D A,4 D MADTD	<pre>Por thorough fault isolation. Provide the standard state of the standard state of the standard state st</pre>

436 026E E1 POP H ; Get remaining size 437 026F 7D MOV A,L ; Subtract tested size 438 0270 93 SUB Е 439 0271 6F MOV L.A 440 0272 70 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 ; Set new base address В 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 ; ; Routine ZTBK zeros and tests for a contiguous 453 0299 ; block of memory. On entry, the (DE) register must ; have the block size and the (HL) register must 454 0299 455 0299 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 С,О ; Write into the block 462 029D 71 ZTBKA: MOV M,C ; Next address 463 029E 23 INX H 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 PUSH 470 02A7 D5 D ; Save parameters 471 02A8 E5 PUSH H 472 02A9 7E ZTBKB: MOV ; Read a cell A, M 473 02AA B9 С ; Same as written? CMP CNZ 474 02AB C4DB01 ADPRA ; Log\_error if necessary 475 02AE CDF301 CALL BREAK ; See if abort wanted 476 02B1 23 ; Next address INX H 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 \$

486 487 488 489	02BB 02BB 02BB 02BB 02BB 02BB		; data ; After ; teste ; is l	bit of a a bit d for zem ogged am	ll addre: is wri ros. Whe s descr:	sse tte en ibe	ingle high bit through each es in a controlled manner. en, all other locations are an error is detected, it ed above. If excess errors y typing CNTRL C.
491 492 493 494	02BB 02BB 02BC 02BD 02BE	E5 23	WLKDA:	PUSH PUSH INX MVI	D H H C,11H	;;;	Save block size Save address Set A0
496 497 498 499	02C0 02C1 02C2 02C3	C5 71 E5 33	WLKC:	PUSH MOV PUSH INX	B M,C H SP	;;;;;	Set DO, D4 (2114) Save it Write byte into memory Save current address Adjust stack to
501 502 <sup>-</sup> 503	02C4 02C5 02C6 02C7 02C8	33 33 E1		INX INX INX POP PUSH	SP SP SP H H	;	find base address Retrieve base address Restore it
505 506 507 508	02C9 02CA 02CB 02CC 02CC	3B 3B 3B 3B	WLKB:	DCX DCX DCX DCX	SP SP SP SP	;	Readjust stack
510 511 512	02CE 02CF 02D0 02D1	47 A7 EB		MOV MOV ANA XCHG XTHL	A , M B , A A	;	Read byte Save byte in (B) Test data Get test address
515 516 517	02D5 02D8	C2DE02 CD1703 CCD101 C3E802		JNZ CALL CZ JMP	DNZT CHLDE ADPRT CONT	;;	Save loop control Non-zero data, jump Test addresses Bad cell Continue test
519 520 521 522	02DE 02DE 02DF 02E2	B9 C2E502 CD1703	; DNZT:	CMP JNZ CALL	C BADD Chlde	;	See if same as test data Jump if bad data Test addresses
524 525 526		E3 EB	BADD: CONT:	CNZ CALL BRI XTHL XCHG INX	ADPRT EAK H	;	See if abort wanted Unscramble registers Next address
528 529 530 531	02EE 02EF 02F0 02F1	1B 7B B2 C2CD02		DCX MOV ORA JNZ	D A,E D WLKB	;	Done on this cell? No, jump
533 534 535	02F4 02F5 02F6 02F7 02F8	C1 33 33		POP POP INX INX POP	H B SP SP D	;	Get test address Get data Get block size
537 538	02F9 02FA 02FB	D5 3B		PUSH DCX DCX	D SP SP	7	

540 O2FC 79		MOV	A,C	; Get data into (A)
541 O2FD 07		RLC		; Shift for next pattern
542 O2FE 4F		MOV	C,A	
543 02FF D2C002		JNC	WLKC	; Not done yet
544 0302 C1		POP	В	; Get base address
545 0303 D1		POP	D	; Get block size
546 0304 3600		MVI	Μ,Ο	Reset test cell
547 0306 7D	-	MOV	A,L	; Strip off base
548 0307 91		SUB	C C	; address
549 0308 6F		MOV	L,A	,
550 0309 7C		MOV	A,H	
551 030A 98		SBB	в. В	
552 030B 67				
		MOV	H , A	· On the member of damage bit
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	В	; Build next address
557 0312 D5		PUSH	D	; Save block size
558 0313 C5		PUSH	В	; Save base address
559 0314 C3BE02		JMP	WLKDA	; Go do it again
560 0317	;			
561 0317	; Compa	re (HL)	register	to (DE) register and set
562 0317	; flags	on resu	ilt.	-
563 0317	;			
564 0317 70	CHLDE:	MOV	A,H	
565 0318 92		SUB	D	
566 0319 CO		RNZ	-	
567 031A 7D	•	MOV	A,L	
568 031B 93		SUB	E E	
569 031C C9		RET	U	
570 031D		NG I		
571 031D	j Doubi	D D D N T N		
-				ously writes a sequence of
572 031D				a specified memory block and
573 031D	; reads	tnem b	ack for	comparison. If errors occur,
574 031D				e console. A running error
575 031D	; total			tained. The test may be
576 031D				e with a CNTRL C; the error
577 031D	; total	. at th	is time	will be displayed on the
578 031D				data steps from 1 to 255
579 031D	; decim	al, then	repeats	itself, always skipping 0.
580 031D	;			
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	Č,1	; Seed the data
587 0327 0600		MVI	B,0	; Initialize error count
588 0329 C5	BRNA:	PUSH	В,0	; Save data, error count
589 032A D5		PUSH	D	; Save block size
590 032B E5		PUSH	H	•
	DDND -			; 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 O
594 0331 OC		INR	С	; Set to 1

-

<b>595</b>	0332	23	BRNC:	INX	Н	:	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		
	0339			POP	Н	:	Get base address
	033A			POP	D	•	Get block size
	033B			POP	B		Get data seed, error cour
	033C			PUSH	D		Restore them
	033D			PUSH	H	,	
	033E		BRND:	MOV	A,M	•	Read data byte
	033F		2	CMP	C		Check it
		CA4703		JZ	BRNE	,	Skip if OK
	0343			INR	B	,	Error count
		CDDB01		CALL	ADPRA	,	Log the error
	0347		BRNE:	INR	C	1	-
		C24C03	DANE.	JNZ	BRNF	,	Change test data
	034B	-				;	Skip if not zero
	034C		DDNP.	INR	C	-	Reset to 1
	034C		BRNF:	INX	H		Next address
				DCX	D	;	Loop control
	034E			MOV	A,E		
	034F			ORA	D		
		C23E03		JNZ	BRND		
	0353			POP	Н		Reset base address
	0354			POP	D		and block size
		CD0901		CALL	CST		Time to quit
		CA2903		JZ	BRNA		No, do it again
		CD0301		CALL	CONI		Get character
	035E	FE03		CPI	'C'-CNTL		
	0360					;	ETX (Cntl C)?
		C22903		JNZ	BRNA	;	No, continue
626	0363	CD1E01		CALL	CRLF		
	0366			MOV	A,B	;	Error count
628	0367	CD3301		CALL	HEX2	;	Print it
629	036A	217003		LXI	H,ERMSG	;	Get error message addres
630	036D	C3AE01		JMP	PRTWA		Print it and return to E
			;			•	
		20455252		DB	' ERROR'		'S'+80H
		4F52D3				•	
633		434F4E54	CBMSG:	DB	'CONTINU	101	US BURNIN',' '+80H
		494E554F					,
		55532042					
		55524E49					
		4EA0					
634	0389	V LINO	•	••			
	0389		; • Routiu	DAS TNTT	and EXE		initialize the computer a
	0389						or a command. When a val
627	0389						ntrol is transferred to t
	0389			priate ro			
	0389		, approj	PLIADE IN			
		219002	; RETN:	1 <b>W</b> T	U TOMOC	•	Deist ITECT DAND!
		CDAB01	ABIN:		•	ÿ	Print 'TEST DONE'
	-		TNT	CALL	PRTWD	•	
		314000 GFE7		LXI			; Set stack point
		21AC03	EXEC:		•	;	Print diag message
044	0395	CDAB01		CALL	PRTWD		

.....

-	0398 039B	218903 E5		LXI PUSH	H , RETN H	; Set up return address
647	0390	CD0301		CALL	CONI	; Wait for command
	039F 03A1	FE43 CA1D03		CPI JZ	'C' BRNIN	; Continuous burn-in
650	03A4	FE57		CPI	1111	; Walking bit
-	-	CA3E02 C3C401		JZ JMP	MADT QPRT	
653	03AC	-	;		-	
654	-	44494147 4E4F5354	DIMSG:	DB	'DIAGNO	STIC:',' '+80H
	03B4	49433AA0				
655	03B8		;			
656	03B8	0000		END		

TOTAL ERRORS=00

# CHAPTER 4

# TECHNICAL INFORMATION

## 4.1 SCHEMATIC/LOGIC DIAGRAM



4-2





TECHNICAL INFORMATION

4.3 PARTS LIST

QTY	REFERENCE	DESCRIPTION	CCS PART #				
CAPACIT	ORS						
3	C5-7	Tantalum, 4.7uf,	42804-54756				
6	C1-4,8-9	35 vdc, 20% Ceramic, .1uf,	42142-21046				
RESISTO	DRS	50 vdc, 20%					
3	Z1-3	Network, SIP, 2.7K x 7	40930-72726				
INTEGRA	ATED CIRCUITS						
64	U10-41,43-58, 61-68,71-78	MOS 2114 1Kx4 Static RAMS	31900-21142 (200nsec) or -21143 (300nsec) or -21144 (450nsec)				
2	U59,69	LM323 +5v regulator	32000-03230				
2	U79,80	74LS136 quad ex-OR:OC	30000-00136				
2	U1,2	74LS20 dual 4-in NAND	30000-00020				
2	U7,8	74LS05 hex inverter:0C	30000-00005				
4	U9,42,60,70	74LS138 octal decoder	30000-00138				
2	U4,86	75453 dual 2-in OR: OC	30300-00453				
1	U6	74LS74 dual D flip-flop	30000-00074				
1	U3	74LSO8 quad 2-in AND	30000-00008				
1	U5	74LS139 2:4 decoders	30000-00139				
4	U81-84	74LS244 Tri buffer	30000-00367				
1	<b>U85</b>	ROM 5623 256x4	30900 <b>-</b> 05623				

4\_4

# TECHNICAL INFORMATION

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

CONTROL ROM INPUTS								ROM OUTPUTS					DECODER OUTPUTS					OPERATION	
×	-BOARD SEL	BNK PT ADDR	MWRITE	-pWR	GROUND	0/1	-PHANTOM	sMEMR	×	R/W DEC B	R/W DEC A	B/B DEC B	B/B DEC A	BANK CLK	-CSE	-RD ENABLE	-WR ENABLE	-PORT READ	
НЕХ	A 7	A 6	Α5	A 4	A 3	A2	A 1	AO	НЕХ	04	03	02	01	Y١	Y2	Yl	Y 2	<b>Y3</b>	
13	0	0	0	1	0	0	1	1	6	0	1	1	0	0	1	0	1	1	Bank-Independent Memory Read
22	0	0	1	0	0	0	1	0	Α	1	0	1	0	0	1	1	0	1	Bank-Independent Memory Write (CPU)
32	0	0	1	1	0	0	1	0	Α	1	0	1	0	0	1	1	0	1	Bank-Independent Memory Write (FP)
46	0	1	0.	0	0	1	1	0	9	1	0	0	1	٦	0	1	0	1	Write to Port, Memory Selected
53	0	1	0	1	0	0	1	1	6	0	1	1	0	0	1	0	1	1	Bank-Dependent Memory Read
56	0	1	0	1	0	1	1	0	C	1	1	0	0	1	1	1	1	0	Read from Port, Memory Selected
62	0	1	1	0	0	0	1	0	Α	1	0	1	0	0	1	1	0.	1	Bank-Dependent Memory Write (CPU)
72	0	1	1	1	0	0	1	0	Α	1	0	1	0	0	1	1	0	1	Bank-Dependent Memory Write (FP)
<b>C6</b>	1	1	0	0	0	1	1	0	9	1	0	0	1	1	0	1	0	1	Write to Port, No Memory Selected
D6	1	1	0	1	0	1	1	0	С	1	1	0	0	1	1	1	1	0	Read from Port, No Memory Selected
		ar	ny oti	her lo	ocatio	on			0	0	0	0	0	1	1	1	1	1	

4.4 CONTROL ROM TRUTH TABLE

# TECHNICAL INFORMATION

# 4.5 ADDRESS/CHIP TABLE

	NUBBLES COO	-FFF	800	-BFF	400	-7FF	000		
	LOW	HIGH	LOW	HIGH	LOW	HIGH	LOW	HIGH	
HIGH NIBBLE	ັບາວ	ווט	U12	U13	U14	U15	U16	U17	GRO
W+1	U18	U19	U20	U21	U22	U23	U24	U25	
x	U26	U27	U28	U29	U30	U31	U32	U33	- E
X+1	U34	U35	U36	U37	U38	U39	U40	U41	
Y	U43	U44	U45	U46	U47	U48	U49	U50	- 9
¥+1	U51	U52	U53	U54	U55	U56	U57	U58	•
z	U61	U62	U63	U64	U65	U66	U67	U68	-
Z+1	U71	U72	U73	U74	U75	U76	U77	U78	'

#### 2032 ADDRESS/CHIP TABLE

### 4.6 2032 BUS CONNECTOR PINOUT



TOP VIEW

#### APPENDIX A

#### LIMITED WARRANTY

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

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

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

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

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

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

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

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

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

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

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

CCS expressly disclaims any and all liability arising from the use and/or operation of its products sold in any and all applications not specifically recommended, tested, or by CCS, in writing. certified 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 warrantv 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

A-2

#### LIMITED WARRANTY

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 These warranties apply to the orginal enforceable. purchaser only, and do not run to successors, assigns, or AS TO ALL PERSONS OR subsequent purchasers or owners; ENTITIES OTHER THAN THE ORIGINAL PURCHASER, CCS MAKES NO WARRANTIES WHATSOEVER, EXPRESS OR IMPLIED OR STATUTORY. The "original purchaser" as used in this warranty shall be term 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.