Z—SCAN 8000

Zilog

User Manual

August 1981

ADVANCE COPY

Zilog

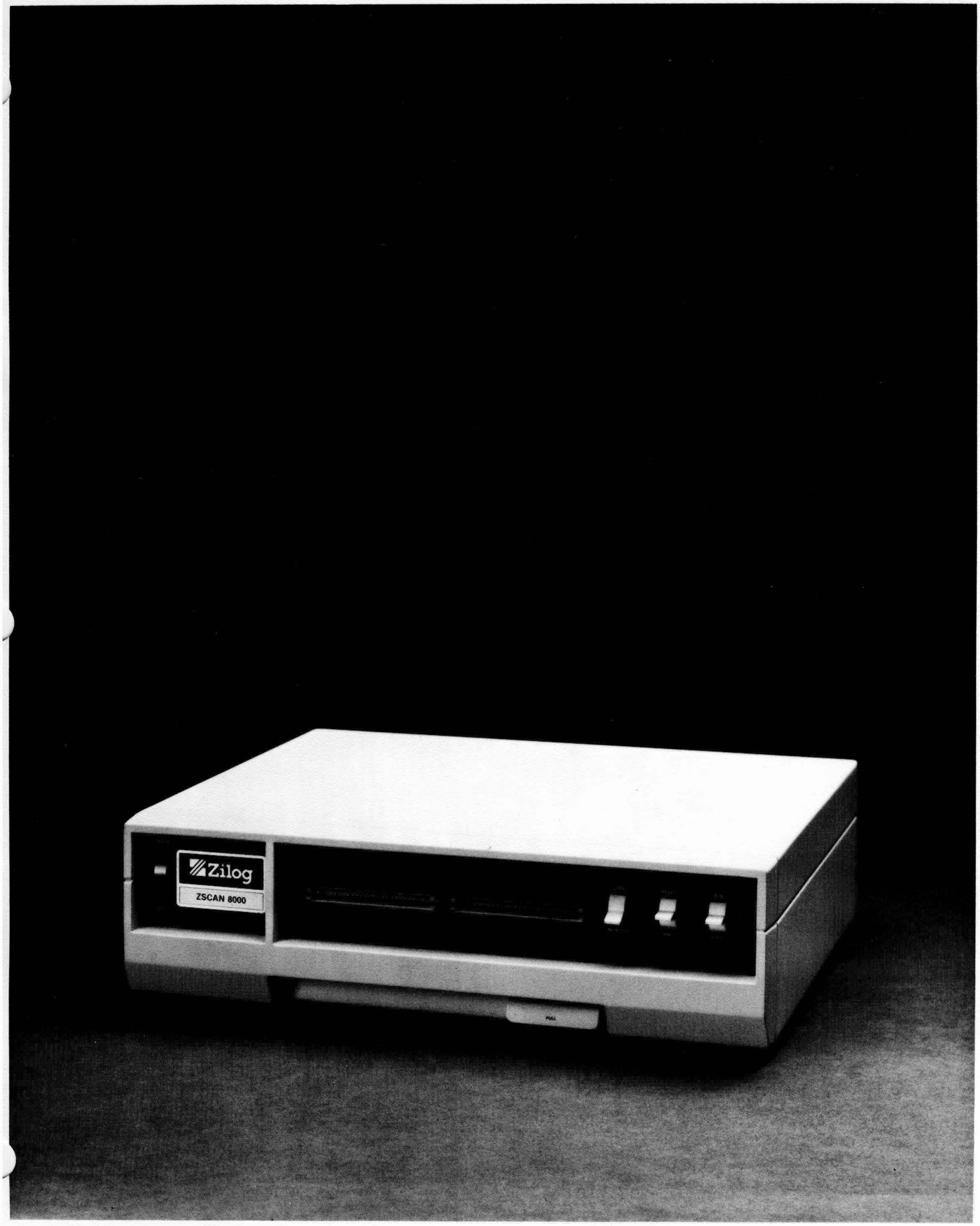
This is a technical manual. The information contained herein is subject to change.

Z-SCAN 8000 EMULATOR

USER'S GUIDE

This manual applies to serial numbers

3000 and up


Zilog
ZSCAN 8000


PDS 80
PRODUCT DEVELOPMENT SYSTEM
Zilog
Zilog

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS
(Continued)

LIST OF ILLUSTRATIONS
(Continued)

LIST OF ILLUSTRATIONS
(Continued)

LIST OF TABLES

LIST OF TABLES
(Continued)

SECTION ONE

HOW TO USE THIS MANUAL

Z-SCAN 8000, a Zilog Stand-Alone Circuit Analyzer, is a free-standing peripheral unit used for the design, debugging, and testing of equipment based on Zilog's Z8001 and Z8002 16-bit microprocessor. Because different users of the Z-SCAN 8000 have varying backgrounds or needs, various approaches to using this manual are suggested.

Writing conventions used throughout this manual:

1. The complement of a signal is specified by a dash (-) following the signal name, e.g., ABC- specifies that ABC is active low ($\overline{ABC}$).

2. The capital letter in each screen and/or command name indicates the key used to access each CRT terminal display.

3. Underscores used within text represent exactly how these items appear on the Z-SCAN terminal screen displays.

Important Notations: Three levels of notation are used throughout this manual to bring attention to specific information and safety precautions. The three levels of notation and their purposes are:

NOTE	Calls attention to specific points of technical or procedural importance.
CAUTION	Calls attention to conditions that could cause possible damage to equipment or facilities.
WARNING	Calls attention to conditions that could cause serious injury to personnel.

Suggested Reading Sequences

A. **All users:**

1. Read Section 3, Unpacking, Installation and Checkout, for special instructions and safety procedures.

2. Continue to those items below that are applicable to your situation and system configuration.

B. **New Users Unfamiliar with Z-SCAN:**

1. Read Section 2, General Description for an overview and glimpse of the features and capabilities of Z-SCAN.

2. Read Section 3 and perform the installation and checkout procedures applicable to your system configuration. All users should work from Sections 3.1 through 3.6. Users with a host system, see items E and F.

3. Read Section 4, Z-SCAN Monitor Tutorial and then work through the appropriate tutorial for your system configuration. The Z8002 and Z8001 tutorials are designed to accomplish two major tasks:

   a. Familiarize the user with the functions and capabilities of Z-SCAN.

   b. Assure that all Z-SCAN functions operate properly.

4. Read Section 6, Monitor Software Description, for more detailed information on commands and displays.

5. Continue to those items below that are best suited to your needs.

C. **Users Familiar with Z-SCAN:**

1. Review Section 4 and work through the appropriate tutorial for your system configuration.

2. See Section 6, Monitor Software Description for more detailed information on commands and displays.

3. Continue to any of the following items that best suit your needs.

D. **Users With a Target System:**

1. Complete all items from either B or C above.

2. Read Section 5 for the target system connections and supporting information that describes how the Z-SCAN 8000 interacts with the target during debugging.

E. **Users with Z-SCAN and a Host System:**

1. Complete all items from either B or C above.

2. If your Z-SCAN configuration includes a target system, complete item 2 of D above.

3. Complete Section 3.7 which describes how to connect a host system to Z-SCAN and initialize Monitor mode and Transparent mode.

4. Read Section 4, Z-SCAN Monitor Tutorials, and then work through the appropriate tutorial for your system configuration. Section 4.6 of the Z8002 tutorial and Section 4.8 of the Z8001 tutorial describe the downloading facility used by a host system.

5. If you have a non-Zilog host system, see F below.

F. **Users With a Non-Zilog Host System**

Z-SCAN is able to communicate with any system that supports asynchronous serial communication.

1. Complete all items from either B or C above.

2. If your Z-SCAN system configuration includes a target system, complete item 2 of D above.

3. Complete section 3.7 which describes how to connect a host system and initialize Monitor mode and Transparent mode.

4. If your host already supports Z-SCAN's downloading facility, you can familiarize yourself with its operation (Section 4.6 of the Z8002 tutorial and Section 4.8 of the Z8001 tutorial).

5. If your host does not support downloading, Section 7, Interface to Non-Zilog Host, contains the information needed to write the required host resident utility program.

SECTION TWO

GENERAL DESCRIPTION

2.1 INTRODUCTION

Z-SCAN 8000, the Zilog Stand-Alone In-Circuit Analyzer, is a free-standing peripheral unit that emulates Zilog's Z8001 and Z8002 16-bit microprocessors. These are MOS/LSI, register-oriented CPUs designed for general-purpose applications. Features of the Z8000 CPUs include:

- Available as 40-pin nonsegmented version (Z8002) or 48-pin segmented version (Z8001)
- 110 instruction types
- Eight addressing modes
- 64-kilobyte (Z8002) or 8-megabyte (Z8001) addressing capability in each of the six address spaces
- Synchronous and asynchronous interrupts
- Automatic dynamic RAM refresh
- Single +5 V dc power supply
- Single-phase clock
- Advanced instructions, such as hardware signed multiplication and signed division, for both 16-bit and 32-bit words

The Z8000 CPU Technical Manual (document #00-2010-C) gives a detailed description of Z8000 CPU architecture and applications.

The Z-SCAN 8000 can operate with Zilog's family of development hosts by interfacing to the host and a CRT terminal via two RS-232C serial ports. Because it employs a standard serial interface, Z-SCAN can be used with any software host system that runs a cross assembler or cross compiler capable of generating Z8000 code. Z-SCAN supports the development and testing of this code. Z-SCAN communicates with a host system through a standard serial format that requires only simple download and upload utilities in the host system. For PROM-based target systems, Z-SCAN can operate as a stand-alone unit with a CRT terminal because the monitor and debug software are EPROM resident.

In keeping with Zilog's design philosophy of separating a development system into two identifiable units (the software host and an emulation peripheral), the Z-SCAN fits into three environments.

- As a peripheral to Zilog's Z-LAB 8000, PDS 8000 and ZDS-1 series of development systems, the Z-SCAN 8000 completes the Zilog development support package for the Z8001 and Z8002 microprocessors:

- As a peripheral to any development host capable of compiling or assembling Z8000 code, the Z-SCAN 8000 offers a low-cost emulation capability, which precludes substantial reinvestment in a software host system.

- As a stand-alone in-circuit emulator operating with a CRT terminal, the Z-SCAN 8000 provides simple testing and debugging capability for PROM-based target systems.

2.2 SYSTEM FEATURES

User Interface. Instead of a line-oriented interface, the Z-SCAN 8000 incorporates a two-dimensional, screen-oriented user interface, which makes it easy to use. A choice of screen erase, line erase and cursor addressing sequences is provided, allowing most popular terminals to be used with Z-SCAN.

The object of the user interface is to provide a screen format with a menu-like approach, which directs the user through the operations of the emulator. At all times, the Z-SCAN 8000 displays information about system parameters, system resources, current execution, and error messages. This feature keeps the user informed of the status of the debug process. When the system is turned on, a bootstrap routine sets the baud rate and produces a display, which informs the user of the unit's configuration and requests the user to define set-up parameters. A menu of display choices offers the user various system capabilities:

- The Memory_io screen display shows the various memory and I/O manipulation commands that give the user access to the target system.

- The Resources screen display presents the full range of arguments applicable to target system emulation.

- The Execution screen display lists all the commands and parameters necessary for emulation.

- The Trace subscreen presents a disassembled listing of each instruction executed during an emulation.

Execution of specific Z-SCAN monitor commands is always possible, and information on other relevant system parameters and resources is always displayed. This highly interactive user interface makes it possible to use the Z-SCAN 8000 without frequent reference to the operating manual.

Shadow Memory. Although the Z-SCAN system uses a single CPU for both monitor and emulation functions, no restrictions are placed on the size of the target system memory. This is because the entire Z-SCAN monitor resides in Z-SCAN PROM (shadow) memory and therefore does not use the target system memory space or addresses.

Hardware Trigger. The Z-SCAN 8000 offers the versatility of setting breakpoints in either or both of two fields: the address/data field and the control/status field. A pass counter can be set from 0 to a maximum of 255 counts to allow multiple-pass triggering. In addition, Z-SCAN 8000 can also be set to break on instruction fetches only (single-step execution) or, by using a pass counter, can be set for a maximum of 250 counts to allow triggering on multiple instruction fetches (multi-step execution).

The breakpoint logic has two operating modes. In the first, the address/data field and the control/status field must simultaneously match the programmed breakpoint condition to terminate an emulation. The second mode allows either an address/data match or a control/status match to terminate an emulation. This mode can be used to terminate emulation when either of two conditions is detected during an emulation, for example, execution of an instruction at a particular address or acknowledgement of an interrupt. This feature, when combined with the multi-step mechanism, allows a break to be programmed on either of two target bus conditions or, if those conditions do not occur, after a set number of instruction fetches.

Mappable Memory. The Z-SCAN 8000 offers a 4K word (8192 bytes) block of high-speed static RAM. This block simulates a target system memory block, which typically is ROM. No wait states are required at 4 MHz. This block is mappable anywhere in the Z8000 address space and can be specified to respond to any combination of normal code, normal data, normal stack, system code, system data or system stack accesses. Mapping must be done on 4K word boundaries only, and the entire block can be protected against illegal emulation memory writes, causing the emulation to either terminate or continue, depending on user options. When a break results from a write protect violation, an error message appears on the CRT display informing the user of an illegal write.

Memory Peek. The Z-SCAN 8000 has a software feature that displays the contents of three 4-word areas of target memory. The display is updated every time an emulation terminates and it supplements the information displayed in the register contents. The three areas displayed, can be at any address in any memory space (system code, normal data, etc.). In addition, the Trace screen displays the data at the top of normal and system stacks.

Wait States. Under software control, Z-SCAN can insert zero to eight wait states in each bus transaction.

2.3 Z-SCAN 8000 SPECIFICATIONS

Processors:	Nonsegmented 40-pin Z8002 CPU or segmented 48-pin Z8001 CPU
Clock Rate:	3.3 MHz (internal), 500 kHz to 4.0 MHz (external)
I/O:	Two RS-232C serial ports for terminal and host
CRT Terminal:	A choice of various CRT terminals (see Table 6.6)
Baud Rate:	Automatically adjusted from 50 to 19.2K per baud set on the terminal
Mappable Memory:	4096 x 16 static RAM (no wait states at 4 MHz when Z-SCAN is operating using an external user clock)
Breakpoint:	Address, data, control, address and control, data and control, instruction fetch; or a combination of instruction fetch and any field argument. The address field on the Z8001 may be offset, segment or segment and offset.
Emulator Input Loading	Reset (RESET-), data strobe, (DS-), non-maskable interrupt (NMI-), vectored interrupt (VI-), nonvectored interrupt (NVI-), segment trap (SEGT-): one low-power Schottky transistor-to-transistor logic (LS-TTL) load plus 10k pullup plus 30 pF max. All others: one LS-TTL load plus 30pF max.

Emulator Output Drive:	Driven by LS-TTL buffer with 33 ohm series termination
Cables:	
Z8002 Emulator Cable	18 inches
Z8001 Emulator Cable	18 inches
Terminal Cable:	48 inches
Front Panel:	TARGET/MONITOR toggle switch, RESET and NMI momentary switches, POWER rocker switch with indicator. 40-pin connector, 3M type 3495 (Z8002) 48-pin connector, 3M type 3496 (Z8001)
Rear Panel:	BNC connector for pulse output, standard LS-TTL level 2x25 pin connectors, 3M type 3483 (terminal and host) 3-pin power connector 1-1/4 in. fuseholder, screwdriver-release type 115/220 voltage selection slide switch
Power:	180-264V ac or 90-130V ac switch selectable; 47-63 Hz; 60 VA max.
Fuse:	1-1/4 in. antisurge, 3 A (120 V), 1.5A (220 V)
Dimensions:	4 in. (10.2 cm) (H) x 17.5 in. (44.5 cm) (W) x 14.5 in. (36.8 cm) (D)
Environmental:	Operates at 10°C to 50°C: relative humidity 10% to 90%
Unit Weight:	25 pounds
Shipping Weight:	30 pounds

2.4 ORDERING INFORMATION

05-0103-01	Z-SCAN 8000 Emulator (Supports Z8001 and Z8002 Emulation and Control)

2.5 RECOMMENDED SYSTEMS FOR USE WITH Z-SCAN 8000

Model	Description	Prerequisite
ZDS-1/25, 1/40	Zilog Development Systems, Floppy-based	Z8000 Software Development Package
PDS 8000, Models 10, 15, 30, 35	Zilog Product Development Systems, Floppy and Hard Disk	Z8000 Software Development Package
Z-LAB 8000	Development Station	None

SECTION THREE

UNPACKING, INSTALLATION AND CHECKOUT

3.1 INTRODUCTION

This section contains instructions for unpacking, installing, checking, and verifying a Z-SCAN unit. The latter part of the section describes the set-up and initialization of a link between the Z-SCAN and a host system. A trouble shooting guide is also included to help users. Refer to Section Five for the target system connection and verification procedures.

All Z-SCAN 8000 shipments include factory-selected 6MHz Z8001A CPU and Z8002A CPU components. The performance is impaired if standard speed 4MHz parts are used. Although the Z-SCAN 8000 can be operated using either the Z8001A CPU or the Z8002A CPU, the original design permitted only one CPU to be installed on the PC Board at a time. As a convenience upgrade, Z-SCAN boards have been updated to allow both CPU's (Z8001 and Z8002) to be resident at the same time. Section 3.9 lists the procedures necessary to change CPU's, especially helpful to those user's with the early Z-SCAN design. For those user's with both CPU's resident on the PC board, refer to Section 3.10 which gives information on rearranging jumpers to access either the Z8001 or the Z8002 CPU.

The following procedures should be followed only if there is no indication of damage. If any damage is detected, installation of the equipment should be immediately suspended and a Zilog field service representative should be contacted.

Three levels of important notations appear within this section to bring attention to specific information and safety precautions. These notations and their purposes are:

NOTE Calls attention to specific points of technical or procedural importance.

CAUTION Calls attention to conditions that could cause possible damage to equipment or facilities.

WARNING Calls attention to conditions that could cause serious injury to personnel.

3.2 UNPACKING

Every Z-SCAN 8000 system is fully inspected and tested before shipment to ensure that it meets specifications. All equipment is packaged for safe transit under normal freight-handling conditions and should arrive ready to be installed. Before unpacking the system, inspect the shipping container for signs of possible damage to the unit during transit. If shipping damage is suspected, claims with the freight carrier should be filed immediately.

To unpack the system, the following steps should be taken:

1. Open top end of box and remove packing.

2. Lift system out of the carton and remove polyethylene plastic covering.

3. Locate the packing list, cables, and accessories and check to see that all items on the packing list are accounted for.

4. Replace all packing in the shipping container and store the container until the unit has been checked out and is considered operational, or if possible, keep all packing material for future use (see Section 3.2.1).

5. Inspect the Z-SCAN unit for external damage, such as dents, broken switches or loose connections. Any sound of loose items inside the cabinet is evidence of damage. **If damage is evident or suspected, make no further attempt to operate the system.**

   If there is no damage a unit checkout followed by a system setup and checkout (see sections 3.5 and 3.6) should be performed <u>without</u> the target or host system.

3.2.1 Reshipment or Relocation

The packing material has been specially tested to protect Z-SCAN for shipment, therefore the packing boxes and materials should be retained after unpacking.

--NOTE--

Refer to Section 5.2.4, Care of the Emulator Cable, before disconnecting or relocating the Z-SCAN system.

Repack the Z-SCAN equipment in the original packing material for reshipment.

3.3 Z-SCAN 8000 POWER CONNECTION

Connecting power to a Z-SCAN 8000 is the same regardless of whether the system will be used as a Z8001 or a Z8002 system. **Special consideration must be given, however, to the power cord used for Z-SCAN, especially if the system is to be used in countries outside United States.** See Section 3.3.1 for information regarding Z-SCAN's power cord for U.S. usage, or Section 3.3.2 for information on Z-SCAN's special power cord to be used in all foreign countries.

--WARNINGS--

a. **The Z-SCAN 8000 must be operated with a three-wire grounded power system. Do not use a two-wire power system, as this can damage the Z-SCAN unit and poses a safety hazard to operators and maintenance personnel.**

b. **The top cover of the unit must not be removed while the unit is connected to a power receptacle. Hazardous voltages exist around the power transformer, the fuse and the power switch. These areas are shown in Figures 3-1 and 3-2.**

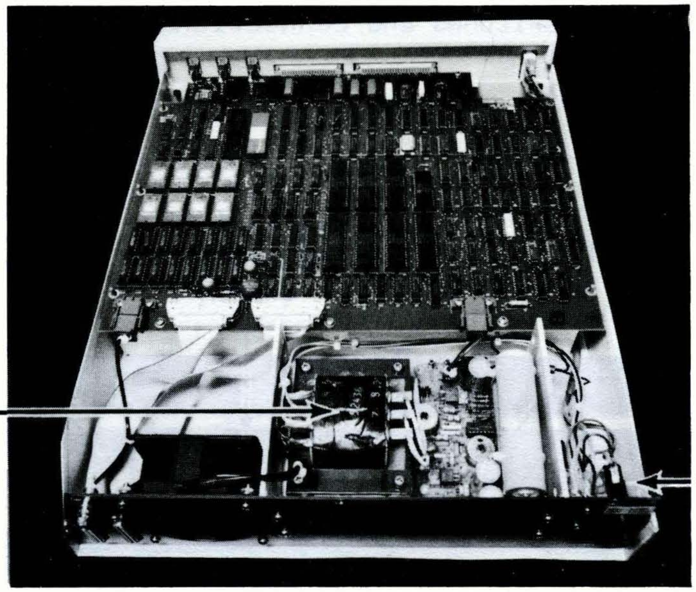


Figure 3-1. High Voltage Areas - Rear

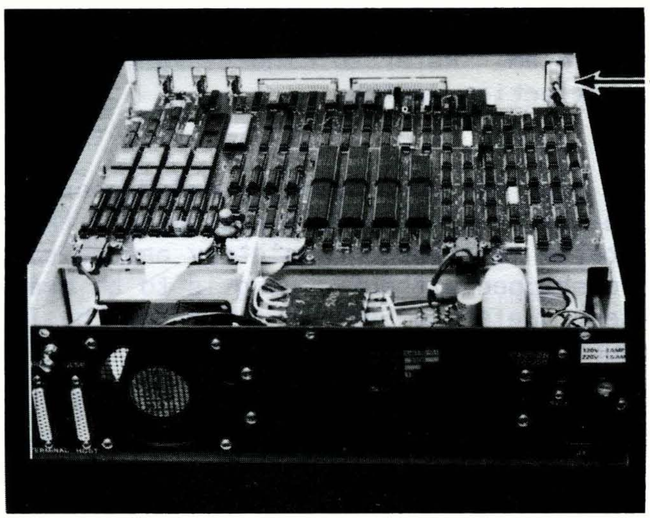


Figure 3-2. High Voltage Areas - Front

Power Up Sequence

1. Turn the red power switch on the Z-SCAN front panel to the OFF position. Figure 3-3 illustrates the switches and connections on the Z-SCAN front panel.

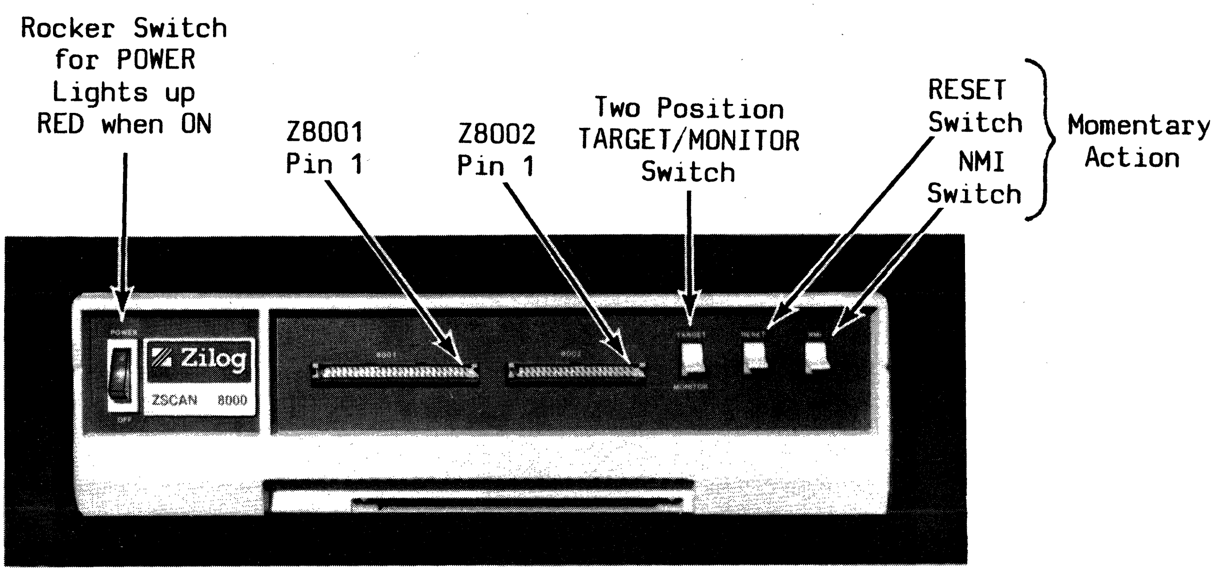


Figure 3-3. **Z-SCAN Front Panel**

2. Locate the 115/220 voltage selection switch on the Z-SCAN rear panel and verify that it is set correctly for the electrical power source to be connected to the Z-SCAN system. Refer to Figure 3-4 for the approximate location of the voltage selection switch.

3. Locate the correct power cord for your country, and discard the other one. Two power cords are included in Z-SCAN 8000 shipments: the power cord to be used in the United States is shipped completely assembled (see Section 3.3.1); the power cord shipped without a plug is to be used in countries outside the United States. Refer to Section 3.3.2 for the instructions to attach a three-wire grounded power plug to this cord.

3.3.1 U.S. Power Cord

The U.S. power cord meets the U.S. National Electrial and Manufacturing regulations and is suitable for use in the United States only. This power cord is identified by the molded plug attached to the cord and is ready to be connected to the Z-SCAN unit. Figure 3-4 identifies the location for the power connection on Z-SCAN's rear panel.

--WARNING--

> **Do not use the U.S. power cord in any other country as it could damage the Z-SCAN system and pose a safety hazard to operators and maintenance personnel.**

Should Z-SCAN's power cord for U.S. usage at any time require a new plug, be sure that the new plug is a three-wire plug and that it is properly grounded.

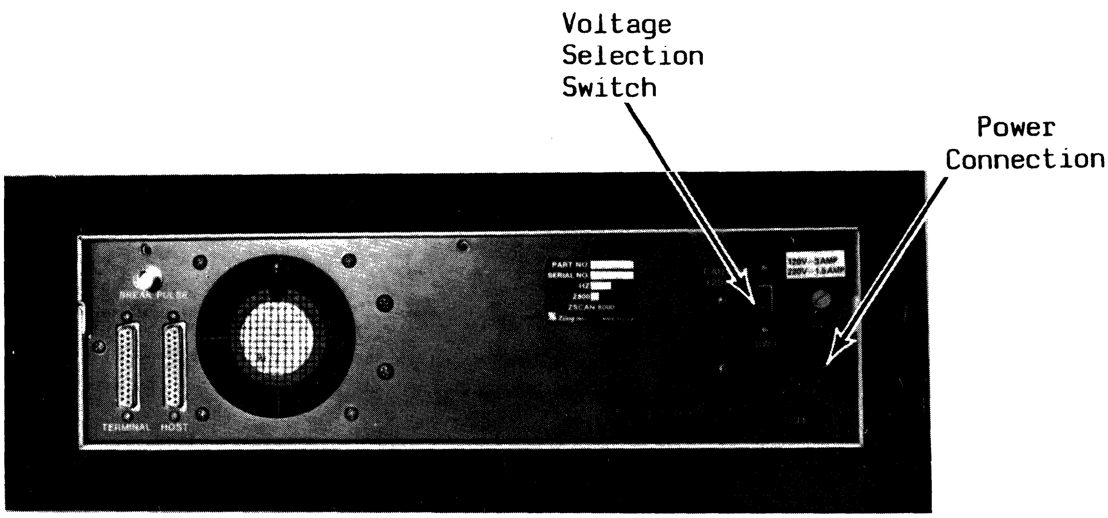


Figure 3-4. Z-SCAN Rear Panel

3.3.2 Power Cord for Other Countries

--WARNING--

The Z-SCAN unit must be safety grounded (earthed).

The power cord for countries outside the United States is shipped without a plug attached. Use the following instructions to connect a plug:

**Connecting a Three-wire Plug
for
Usage In All Countries Except
the United States**

The wires in European power cable (mains lead) are colored in accordance with the following code:

- Green and Yellow - Earth (safety grounded)
- Blue - Neutral
- Brown - Live

Since these colors might not correspond with the colored markings identifying the terminals in your plug, connect as indicated in Table 3-1.

Table 3-1. Cable Leads to Connector Terminal Interconnections

Cable Lead (Wire Color)	Use	Connector Terminals: Marking or Color	
1. Green and yellow	Ground (Earth)	E or Earth	Green or Yellow and Green
2. Blue	Neutral	N	Black
3. Brown	Live	L	Red

3.4 Z-SCAN UNIT OPERATIONAL CHECK

It is important that following the unpacking and installation of the proper power cord to the Z-SCAN unit, all users perform the following initial check of the Z-SCAN unit. **These procedures determine if your Z-SCAN unit is operational and must be done before connecting the CRT terminal to Z-SCAN or before changing the CPU to the Z8002 CPU.**

--NOTE--

If during the unpacking the Z-SCAN unit appeared undamaged and the power cord installation has been successfully completed, this procedure must be attempted before requesting repair service.

1. Set the voltage selection switch (rear panel of Z-SCAN) to the proper setting for your facility. The two settings on Z-SCAN's voltage selection switch are 115 and 230. See Figure 3-4, Z-SCAN Rear Panel.

2. Power up Z-SCAN by pushing the red rocker-type switch on the front left panel to the POWER position. This turns the electrical power on to the Z-SCAN unit. See Figure 3-3, Z-SCAN Front Panel.

3. Make sure that the red indicator in Z-SCAN's power switch is illuminated.

4. Make sure that the Z-SCAN cooling fan is running; do not block the fan exhaust area with cables, books, prints, etc.

5. Place the front panel TARGET/MONITOR switch in the MONITOR position. This is a two-position switch; the "up" position is used for Target mode operations, and the "down" position is used for Monitor mode operations.

6. Verify that the two other front-panel switches (to the right of the TARGET/MONITOR switch) toggle correctly. These toggle switches are the RESET switch and the NMI switch. Push up on each switch to toggle it. When released the switches should return to their original positions.

7. Turn the power switch to the OFF position.

3.5 Z-SCAN 8000 AND CRT TERMINAL CONNECTION

Three ribbon cables are included with Z-SCAN as follows:

- o A 25-pin terminal cable, 1-1/4" (3.2 cm) wide with identical 25-pin connectors at each end
- o A 48-pin emulator cable for the Z8001, 2-1/2" (6.5 cm) wide
- o A 40-pin emulator cable for the Z8002, 2" (5.2 cm) wide

1. Select the terminal cable.
2. Connect one end of the terminal cable to the Z-SCAN rear panel RS-232 connector labeled "terminal." Figure 3-5 illustrates the connecting areas for the CRT terminal and Z-SCAN 8000 (stand-alone configuration).

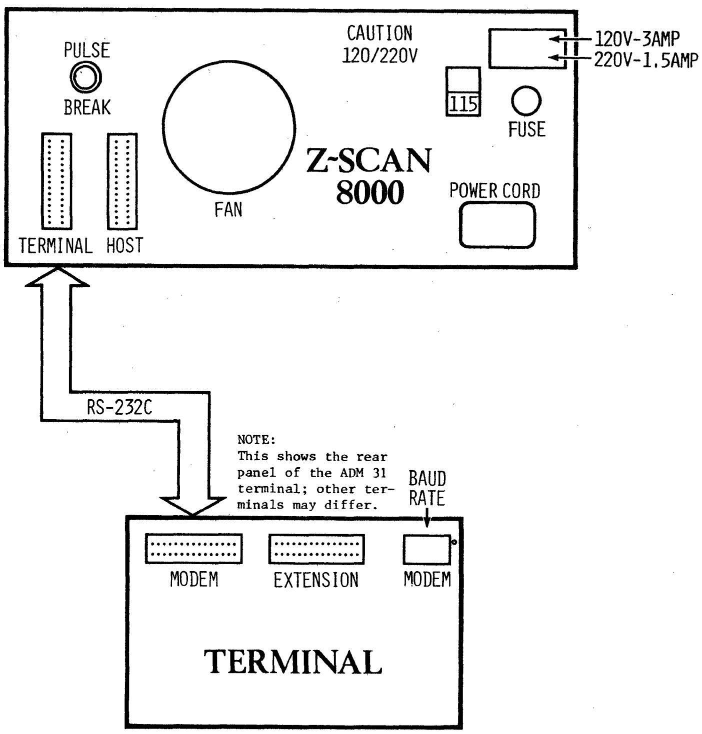


Figure 3-5. Z-SCAN Unit and Terminal Connections (Stand-Alone Configuration)

3. Connect the other end of the terminal cable to the main port connector on the rear panel of the CRT terminal. This socket is labeled "MODEM" on most types of terminals.

--NOTE--

Other CRT terminal connectors for printers or auxillary equipment are not used by Z-SCAN.

4. Verify that the Z-SCAN 8000 and the CRT terminal are connected to a working power outlet. DO NOT TURN ON EITHER UNIT AT THIS TIME.

5. Ensure that the CRT terminal is set to a baud rate supported by the Z-SCAN. Table 3-2 lists supported speeds. Appendix A gives details of specific terminals. A baud rate of 9600 is recommended for most applications.

6. Power up the CRT terminal only. A cursor should appear at top left of the CRT display shortly after power up.

7. Depress the Cap Lock Key and ensure that the CRT terminal is in Cap Lock Mode (this is optional and not required for Z-SCAN systems operating under Monitor Version 3.1 or higher).

8. Turn the CRT terminal power switch to OFF.

3.6 Z-SCAN 8000 VERIFICATION

It is assumed that all previous procedures have been correctly performed up to this point. Begin the following procedures with both the Z-SCAN unit and the CRT terminal turned OFF.

A. CRT Terminal Power Up and Setting the Hardware Baud Rate.

1. Power up the CRT terminal connected to Z-SCAN. A cursor should appear at the top left of the CRT display about 10 seconds after power up.

2. Make sure that the cursor is displayed on the CRT terminal. If the cursor does not appear, refer to the Troubleshooting Guide at the end of this section.

3. Ensure that the terminal is set to a baud rate supported by the Z-SCAN. Table 3-2 lists supported speeds. Appendix A gives details of specific terminals. A baud rate of 9600 is recommended for most applications.

4. Make sure that the terminal is in CAP Lock mode (not required for Monitor version 3.1 or higher).

Table 3-2. ADM 31 Baud Rates Supported by Z-SCAN

Baud Rate	ADM 31 Switch Setting
75	1
110	2
134.5	3
150	4
300	5
600	6
1200	7
1800	8
2400	10
4800	12
9600	14

B. Z-SCAN Unit Power Up and Power Check

1. Power up Z-SCAN by turning the red rocker-type switch on the front left panel to the POWER position.

2. Make sure that red indicator in Z-SCAN's power switch is illuminated. This tells you that the Z-SCAN unit is turned on, and that power is being received from the power outlet.

3. Make sure that the Z-SCAN cooling fan is running and that the fan exhaust area is kept clear.

Power Up Sequence

1. Turn the red power switch on the Z-SCAN front panel to the OFF position. Figure 3-3 illustrates the switches and connections on the Z-SCAN front panel.

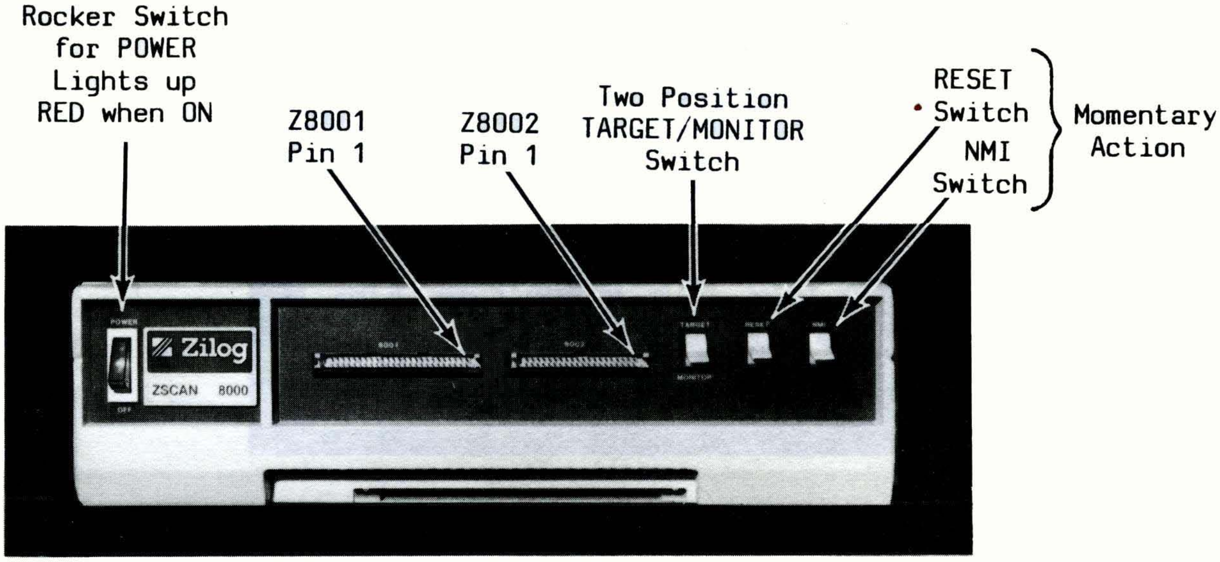


Figure 3-3. Z-SCAN Front Panel

2. Locate the 115/220 voltage selection switch on the Z-SCAN rear panel and verify that it is set correctly for the electrical power source to be connected to the Z-SCAN system. Refer to Figure 3-4 for the approximate location of the voltage selection switch.

3. Locate the correct power cord for your country, and discard the other one. Two power cords are included in Z-SCAN 8000 shipments: the power cord to be used in the United States is shipped completely assembled (see Section 3.3.1); the power cord shipped without a plug is to be used in countries outside the United States. Refer to Section 3.3.2 for the instructions to attach a three-wire grounded power plug to this cord.

Figure 4-1. Lear Siegler ADM 31 Keyboard Layout

An upgrade of Z-SCAN's Monitor (version 3.1) permits both upper and lower case terminal input to Z-SCAN.

NOTE

Z-SCAN Monitor Version 3.0 ignores lower case letters except in file names for Load/Send. The cap lock key on the Lear Siegler ADM31 must be illuminated. Monitor Version 3.1 and higher allow both upper and lower case letters for use on commands and the cap lock is no longer required.

Certain key symbols are set in boldface in the tutorial to aid in identifying error recovery points, (as described in Section 4.4.1). Enter these keys as you would normally.

Table 4-2 describes the four operations possible with the Z-SCAN front panel switches. Section 5.3 describes the effects of the TARGET/MONITOR, RESET and NMI switches.

3.7 Z-SCAN CONNECTION TO HOST SYSTEM

It is assumed that the user has successfully completed all of the previous installation and checkout procedures for the Z-SCAN 8000 unit and the CRT terminal connection.

The following step-by-step procedures specify how to **safely** connect a host system to the Z-SCAN system, and perform the basic initialization of the Monitor mode and Transparent mode operations.

If you have problems performing any of the following steps, refer to the Troubleshooting Guide at the end of this section.

A. Connect Host System to Z-SCAN.

1. Before connecting the host system to Z-SCAN, turn all units OFF: the Z-SCAN unit, the CRT terminal, and the host system.

2. Connect the host system terminal cable to the RS-232C connector on the rear panel of Z-SCAN that is marked "Host". Figure 3-8 illustrates the proper connections. Table 3-3 lists the socket to which the terminal cable should be attached on Zilog host systems.

Table 3-3. Zilog Host System Terminal Connectors

System Type	Socket Designation
PDS 8000 Model 5, 10, 15	J106
PDS 8000 Model 20, 25, 30	J106
ZDS 1/25	J108
ZDS 1/40	J106

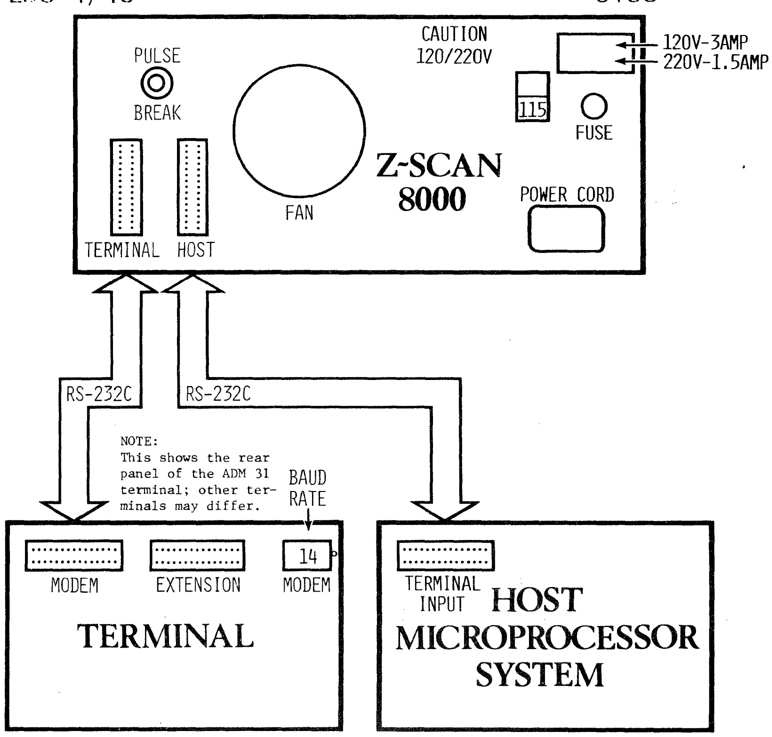


Figure 3-8. Z-SCAN/Terminal/Host Configuration

B. Initialize Monitor Mode, Terminal System Screen and System Screen.

1. Turn all units ON (CRT terminal, Z-SCAN and host system).

2. Toggle the RESET switch again and press RETURN once. The Terminal Selection screen appears.

3. Enter the appropriate terminal type selection number on the Terminal Selection screen and press RETURN once. The System screen appears on the terminal and Z-SCAN is now in Monitor mode.

   If an incorrect terminal type selection number for your type of terminal is entered on the Terminal Selection screen, it is necessary to redo step two which re-initializes the Terminal Selection screen and then enter a RETURN to initialize the System screen.

C. Select Parameters for Variable Fields on System Screen.

1. If the host operates at a baud rate which differs from that of the terminal enter RETURN to move the cursor into the first variable field (host baud rate) on this screen.

2. Use the SHIFT and > keys to select the appropriate baud rate for your host terminal. Refer to Table 3-4 for the host baud rates supported by Z-SCAN.

Table 3-4. Host Baud Rates Supported by Z-SCAN

19,200	1,200	134.5
9,600	600	110
4,800	300	75
2,400	200	50
1,800	150	

3. If you wish to change the default value of the status_to_target field, use the cursor down key to move the cursor from the host_baud rate to this variable field, then use the SHIFT and > keys repeatedly to select the parameter desired.

4. A RETURN must be entered to move the cursor from either of the two variable fields in the System screen to the System screen name (menu area).

NOTE

The cursor must be on the System screen name to either continue operations in Monitor mode or to change to Transparent mode.

D. **Change to Host (Transparent) Mode from Monitor Mode.**

1. To select Host (Transparent) mode, type H. The initialization message for the Host screen is the word HOST which appears in the upper left-hand corner of the screen. If the terminal and host baud rates differ, set the terminal baud rate to match that of the host, then enter RETURN.

   The Z-SCAN system is now in the Transparent mode and serves only as a link between the terminal and the host system.

   Entry of a RETURN should elicit the same response from the host, as would be expected if the terminal was directly connected to the host. Refer to the host's documentation for further details.

E. **Return to Monitor Mode from Host (Transparent) Mode.**

1. Press the terminal keyboard BREAK key.

2. If the baud rates selected for host and terminal differ, set the terminal baud rate to match that required by the Z-SCAN monitor, then enter RETURN. When the terminal baud rate is correct, the Z-SCAN returns to Monitor mode and displays the System screen.

3.8 TROUBLESHOOTING GUIDE

It may happen that the correct display does not appear on the terminal screen at the end of the verification procedure, or that the host system does not respond correctly. In most cases, this indicates a small oversight in connection or verification rather than a fault with the Z-SCAN or the terminal. Table 3-5 lists the most common symptoms of problems and their causes.

Table 3-5. Troubleshooting Guide

Symptom	Cause	Solution
1. Cursor does not appear and cap lock indicator does not illuminate when terminal switched on.	Terminal is not receiving power from the power outlet.	Use a live power outlet and check terminal power connection and switch.
2. Cursor does not appear when terminal switched on.	Brightness control incorrectly adjusted.	Adjust control (small adjacent to baud rate switch on ADM 31 rear panel).

Table 3-5. Troubleshooting Guide

Symptom	Cause	Solution
3. Terminal displays meaningless data as soon as it is switched on.	Z-SCAN was powered on before terminal and has interpreted switch-on of terminal as a baud rate synchronization signal.	Toggle Z-SCAN RESET switch, then enter RETURN. Check to see that the selected baud rate is supported by Z-SCAN (Table 3-4).
4. Terminal displays meaningless data after RETURN entered.	The value set on the baud rate switch at the rear of the terminal corresponds to a rate not supported by Z-SCAN.	Reset the baud rate switch at the rear of the terminal to a baud rate supported by Z-SCAN. Then toggle Z-SCAN RESET switch and enter RETURN.
5. No display after RESET and RETURN entered.	Z-SCAN not correctly connected to terminal.	Check that cable links modem socket on terminal to terminal socket on Z-SCAN. Then toggle the Z-SCAN RESET switch and enter RETURN again.
	Z-SCAN's internal clock source jumper is in "external" position. (All units are shipped with this jumper in the "internal" position.)	Refer to Section 5.2.1 for instructions on altering clock jumper position.
6. System screen not displayed correctly after entry of terminal selection number and RETURN.	Terminal selection number incorrect.	Repeat instructions of Section 3.6 using correct selection number.
	Terminal's option switch settings incorrect.	Refer to Appendix A and terminal documentation. Repeat instructions of Section 3.6 with correct settings.
7. Host system does not respond to characters entered after Z-SCAN has displayed "host" message or responds with garbage.	Host incorrectly connected or in need of initialization.	Check cable and refer to host system documentation.
	Host baud rate does not match Z-SCAN and terminal baud rate.	Set terminal baud rate (Table 3-1), toggle Z-SCAN RESET switch, then enter return H.

3.9 CHANGING THE CPU

The Z-SCAN 8000 can emulate either the Z8001 or the Z8002 CPU, depending on the CPU type installed in the unit. The following procedures are primarily applicable to the Z-SCAN units which permit only one CPU (either the Z8001 or Z8002) to be resident on the PC board at a time. The same monitor software PROMs and option jumper settings are used for either CPU type.

--Note--

Z-SCAN units with both CPU's resident on the PC board require special jumper placements to access either the Z8001 or the Z8002 CPU; refer to Section 3.10 for this information.

To change the CPU, proceed as follows:

1. Switch Z-SCAN power off by pushing the red power switch located on the front panel to the OFF position.

2. Remove the power cord from the socket on the rear of the unit.

--WARNING--

Failure to remove power from the unit prior to removal of the cover may result in exposure to hazardous voltages.

3. Remove the three screws and washers that secure the top cover of the unit, at the left, center and right of the rear panel, as shown in Figure 3-9. Store the screws and washers in a safe place.

4. Grasping the rear of the top cover, lift it upward and rearward to release the cover from the front panel (Figure 3-9).

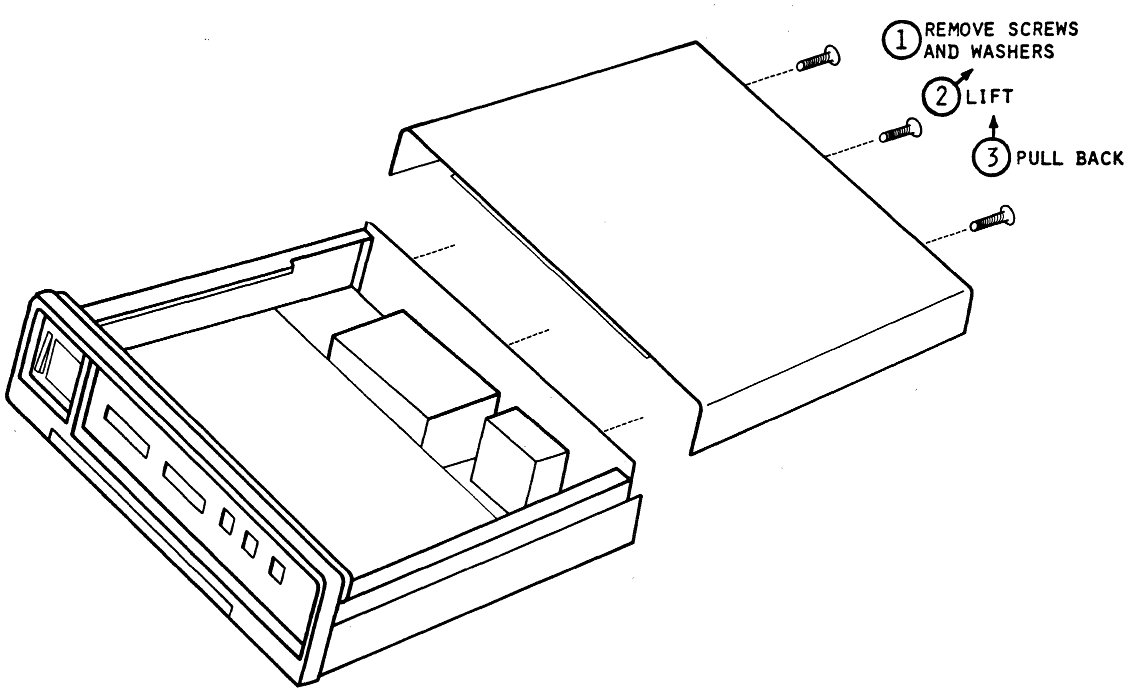


Figure 3-9. Top Cover Removal

5. Locate the two CPU sockets towards the front right of the Z-SCAN circuit board. Using a small screwdriver or IC removal tool, gently pry the installed CPU from its socket and place it in conductive foam for safekeeping.

--CAUTION--

> To avoid possible damage to NMOS components by static discharge, it is recommended that both the Z-SCAN chassis and the user are grounded through a high-impedance circuit while the CPU is changed. **Do not use the power cord to effect a ground connection.**

6. Install the alternative CPU in the correct socket. Care is required to avoid bending the pins. The 40-pin socket for the Z8002 is on the left, and the 48-pin socket for the Z8001 is on the right. The notch identifying pin 1 of the component must face towards the rear of the unit.

7. To replace the top cover, locate the front flange under the front bezel; insert the front edge of the top cover under the front bezel and swing the rear down. Make sure that the rear flange is inside the rear panel of the unit.

--WARNING--

> **Do not connect power to the unit until the top cover has been replaced and secured.**

8. Replace the screws and washers removed in step three.

9. Reconnect the power cord to the rear of the unit and verify correct system operation by following the procedure of Section 3.6.

3.10 SPECIAL JUMPER PLACEMENTS FOR DUAL RESIDENT CPU'S

As a convenience upgrade, Z-SCAN production PC boards have been updated to allow both CPU's (Z8001 and Z8002) to be resident at the same time. The new board is identified by the four sets of jumpers at the base of the Z8002 socket (U124). The jumpers are labeled as E17 through E28.

To change between active CPU's:

1. Press Z-SCAN's power switch to the OFF position. Power must be turned off before changing between active CPU's.

2. Locate the four sets of jumpers at the base of the Z8002 socket marked U124 (see Figure 3-10).

3. Move the four jumpers to the appropriate row as indicated in Table 3-6.

Table 3-6. Jumper Placements for Dual Resident CPU's

Z8002	Z8001
E 17-18	E 18-19
E 20-21	E 21-22
E 23-24	E 24-25
E 26-27	E 27-28

4. Change the front emulator cable to the correct one (refer to Section 5.2.2, Connection of the Emulator Cable).

5. Press Z-SCAN's power switch to ON.

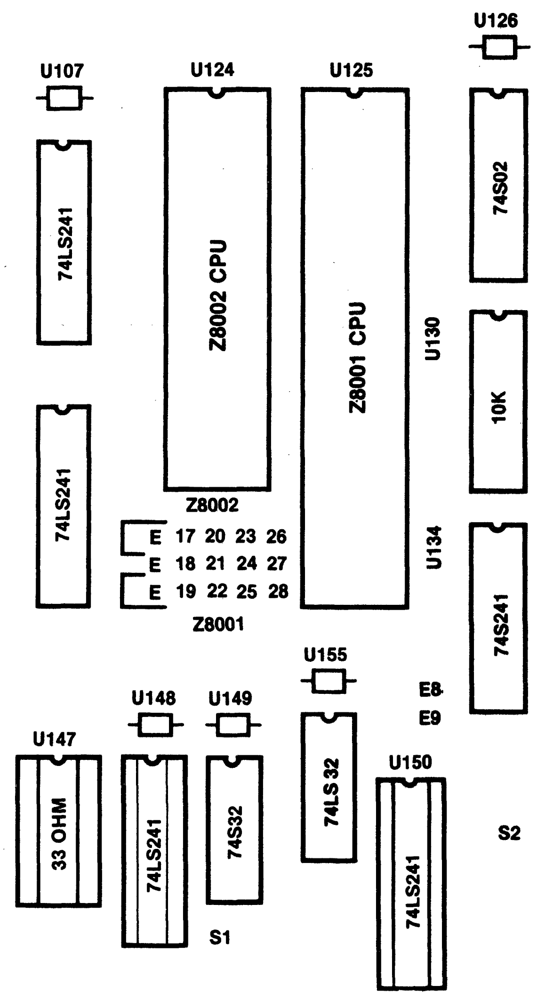


Figure 3-10. Location of CPU Jumpers on PC Board Component Layout

SECTION FOUR

Z-SCAN MONITOR TUTORIAL

4.1 INTRODUCTION

Z-SCAN monitor software is designed to utilize the facilities offered by a CRT (cathode ray tube) terminal. The entire CRT is used to present the required information, which gives a more complete picture of emulation status than would be possible on a printing terminal. This two-dimensional user interface also allows Z-SCAN to display, for user reference, all the commands that might be entered in a particular context.

These features make the monitor software very easy to use. The tutorial sessions in this section provide keystroke-by-keystroke and screen-display-by-screen-display introduction to the Z-SCAN monitor software. As the keystrokes and displays differ slightly between the Z8001 and Z8002 CPUs, two versions of the same tuturial are presented.

The tutorials are not designed to present every feature of the Z-SCAN monitor in detail. Instead, they give a feeling for the way the software operates. Section 6 gives definitive information about each of the Z-SCAN monitor commands.

4.2 TUTORIAL HARDWARE REQUIREMENTS

The majority of the session requires no equipment other than a Z-SCAN unit and a terminal. There is no need for Z8000-based target equipment, because Z-SCAN can run emulations even when no target is connected. The final part of each tutorial requires a Zilog host system for demonstrating the Z-SCAN down-loading facility. If you do not have a Zilog host system, you can still run the download demonstration if download software compatible with Z-SCAN exists on your host. For most hosts, this software is provided by the supplier of the Z8000 support software that operates on the host. If your host does not support the Z-SCAN download protocol, Section 7 provides the information required to write a suitable utility program. If you do not have a host system, the example programs can be copied from the Z-SCAN monitor ROM instead of being downloaded.

The parts of the tutorials that demonstrate the download facility do not depend on conditions set up in the previous sections of the tutorials. This means that you do not need to work through the first part of the tutorial script if you only want to use the part dealing with the Load command.

4.3 THE KEYBOARD AND USER CONTROLS

Figure 4-1 shows the layout of the Lear Siegler ADM 31 terminal keyboard. Before starting this tutorial, it is important to know the positions of the keys required and the symbols used in the text to designate these keys. This information is listed in Table 4-1. If your terminal is not an ADM-31, consult Appendix A and the terminal's documentation to find the corresponding keys on its keyboard.

Table 4-1. Key Names and Locations

Key Name	Key Symbol	Text Symbol	ADM-31 Keyboard Position
(letters)	A through Z	A through Z	Center left. Ensure cap lock key is lit.
(numbers)	0 through 9	0 through 9	Top left to center or numeric pad at right
return	RETURN	RETURN	Far right or center right
space	(blank)	space	Bottom
control-R	CTRL R	CTRL R	Press the control key and R simultaneously.
break	BREAK	BREAK	Top center right
less than	<	<	Bottom center - press shift key and comma simultaneously
greater than	>	>	Bottom center - press shift key and period simultaneously
cursor down	↓	down	Bottom right
cursor up	↑	up	Bottom right
cursor left	<--	left	Bottom right
cursor right	-->	right	Bottom right

Figure 4-1. Lear Siegler ADM 31 Keyboard Layout

An upgrade of Z-SCAN's Monitor (version 3.1) permits both upper and lower case terminal input to Z-SCAN.

NOTE

Z-SCAN Monitor Version 3.0 ignores lower case letters except in file names for Load/Send. The cap lock key on the Lear Siegler ADM31 must be illuminated. Monitor Version 3.1 and higher allow both upper and lower case letters for use on commands and the cap lock is no longer required.

Certain key symbols are set in boldface in the tutorial to aid in identifying error recovery points, (as described in Section 4.4.1). Enter these keys as you would normally.

Table 4-2 describes the four operations possible with the Z-SCAN front panel switches. Section 5.3 describes the effects of the TARGET/MONITOR, RESET and NMI switches.

Table 4-2. Z-SCAN Front Panel Switch Operation

Text Representation	User Action
Monitor RESET	1) Check that MONITOR/TARGET switch points to MONITOR 2) Push RESET switch up
Monitor NMI	1) Check that MONITOR/TARGET switch points to MONITOR 2) Push NMI switch up
Target RESET	1) Check that MONITOR/TARGET switch points to TARGET 2) Push RESET switch up
Target NMI	1) Check that MONITOR/TARGET switch points to TARGET 2) Push NMI switch up

Photographs of the Z-SCAN screen illustrate the sequence of operations. Each step in the sequence is separated by a comma and a space. Do not enter either. Table 4-3 gives examples of keying sequences as they are shown in the tutorial script and as they are actually entered.

Table 4-3. Key Sequence Examples

Script	Keystrokes Required
A, 8, 0, B	A80B
>, <, <, 0, 1	><<01
Q, S, R, A	QSRA

4.4 TUTORIAL PRESENTATION

Before beginning the tutorial, Z-SCAN must be powered up and connected to a terminal, as described in Section 3.5, steps 1 through 8. If you have a Zilog host system, it must also be connected to Z-SCAN in order to demonstrate the download feature. Host connection is detailed in Section 3.7, steps 1 through 7.

The tutorials are presented as a series of steps in tabular form. For each step, a sequence of operator actions is given. For the first few steps, you will probably want to enter each keystroke separately, examining its effect on the display before entering the next. As you become more familiar with the monitor, you will recognize common keystroke sequences that can be entered as a block. The Z-SCAN monitor has a type-ahead feature that allows it to accept new user input before it has finished processing previous input. Note, however, that type-ahead only operates when the monitor software is running, not when a user program is running during an emulation. You will also recognize that not all the keystrokes listed for each step are strictly necessary. Some redundant entries are included simply to illustrate their effect.

The accompanying text explains the effect of your input for each step in the script. In most cases, photographs of the screen highlight areas of interest. The text introduces a number of technical terms specific to Z-SCAN. The first appearence of each term is in boldface.

4.4.1 Error Recovery

It is quite likely that sometime during the tutorial you will make a keying error. Often this has no effect, because in many situations Z-SCAN ignores invalid input. You simply need to follow the incorrect keystroke with the correct one.

In other cases, an incorrect key can be accepted as valid input. When this happens, what you see on the screen at the end of a step is not the same as the photograph in the manual. It is important to backtrack and fix the incorrect parts of the screen before proceding to the next step in the tutorial. To make this easier, error recovery points are identified in the script. If, at the end of a step, the display is incorrect, proceed as follows:

1. If the cursor is not inside the parentheses on line 23 of the display, enter RETURN. If this does not move the cursor to line 23, try a second RETURN, BREAK or monitor NMI.

2. If the screen name at the left of the line in which the cursor now rests is not that shown in the most recent photograph, call up the correct screen by entering the first character of its name.

3. Re-enter tutorial input from and including the last boldface keystroke in the script to the end of the current step.

4. If the display is still incorrect, try to correct it by using unscripted key sequences. Make sure that the cursor is in the position shown on the photograph at the end of the sequence.

5. Should this fail to correct the error, the safest thing to do is to restart the tutorial from step 1. It should seldom be necessary to restart, especially as you become more familiar with the Z-SCAN commands.

4.5 TUTORIAL SCRIPT FOR Z8002

The Tutorial Script for the Z8002 begins on the following page. If a Z8001 is currently operating in your Z-SCAN unit, follow the tutorial of Section 4.7. Be sure not to type the commas or spaces shown throughout the key sequence.

Step	Key Sequence	Commentary
1.	**Monitor RESET,** RETURN	Z-SCAN is RESET. All information about the previous state of the hardware and software is lost. The monitor software uses the RETURN character to set up a baud rate generator, then displays a **menu** of the CRT terminal types supported by the software. The cursor (a steady or flashing bright square on most terminals) appears in the center of the bottom screen line.
2.	Terminal selection digit	To configure the monitor for your terminal, enter one of the digits listed in the menu. If your terminal is not one of those listed on the menu, consult Appendix A and the documentation for the terminal. Pick a digit that corresponds to a protocol supported by the terminal.
3.	RETURN	The CRT screen is cleared, and the **System screen** is displayed. The cursor rests on the name of the screen, which is in parentheses on line 23, part of the **menu area.** This screen gives information about the status of the Z-SCAN hardware, for example, the CPU type currently operating and the software revision level. The displayed baud rates and revision level may differ from those shown in the figure, but the CPU type must be the same. If it is not, follow the alternative tutorial of Section 4.7. If the display is corrupted, the digit entered in step 2 is incorrect and you must repeat the tutorial from step 1.

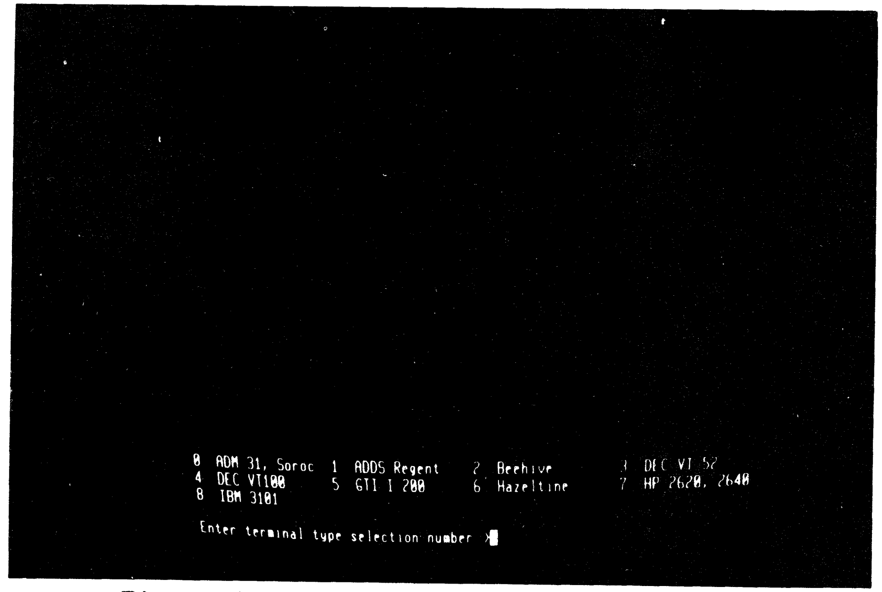


Figure 4-2. Terminal Selection Screen

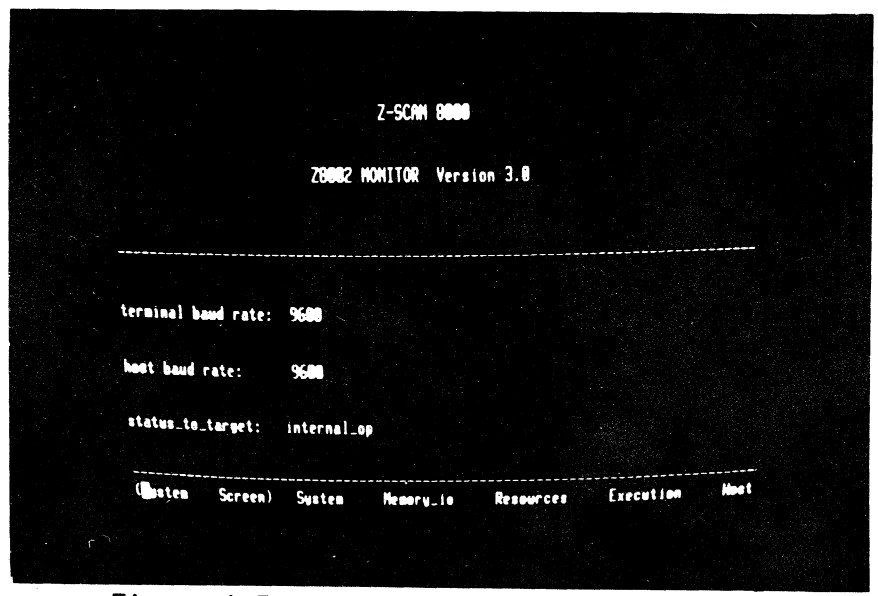


Figure 4-3. Z8002 Monitor System Screen

Step	Key Sequence	Commentary
4.	**M**	The single-keystroke **commands** you are allowed to enter appear as upper-case letters in the words outside the parentheses in the menu area. The command M calls up the **Memory_io** screen. Again, the cursor rests on the name of the screen, which appears on line 23 in the menu area.

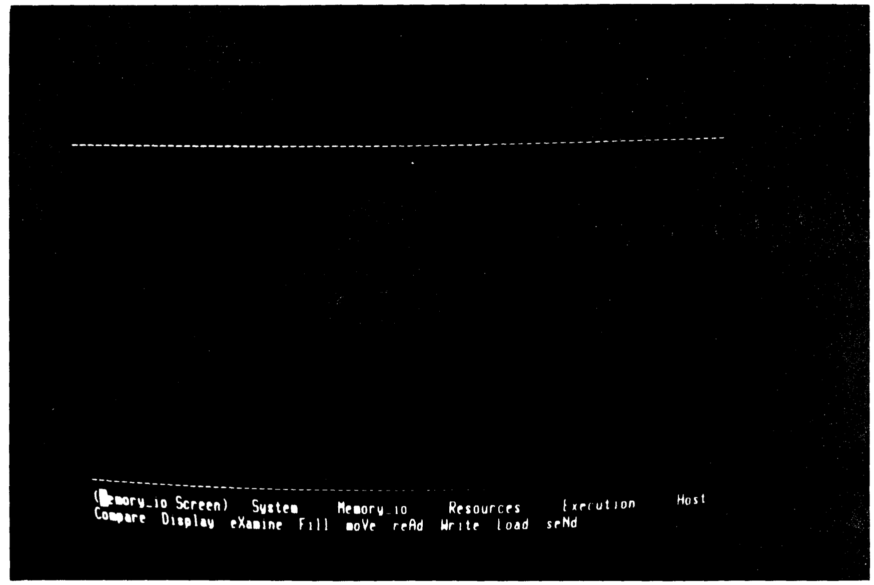


Figure 4-4. Z8002 Monitor Memory_io Screen

Step	Key Sequence	Commentary
5.	**S**, R	Among the valid commands shown in the menu area is S. Entering the command **reactivates** the system screen. A third screen, the Resources screen, can be called up by entering R. As usual, the cursor rests on the screen name, and legal commands are listed in the rest of the menu area.

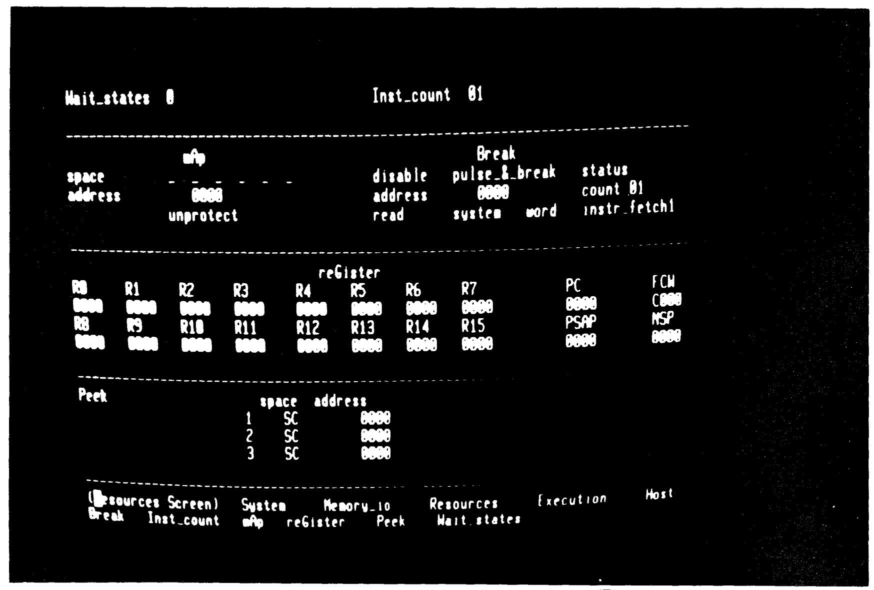


Figure 4-5. Z8002 Monitor Resources Screen

Step	Key Sequence	Commentary
6.	E	In step 5, you went from one screen to another by way of the System screen. However, it is usually possibly to move from one screen to another with a single keystroke. The Execution screen is **activated** by the command E.

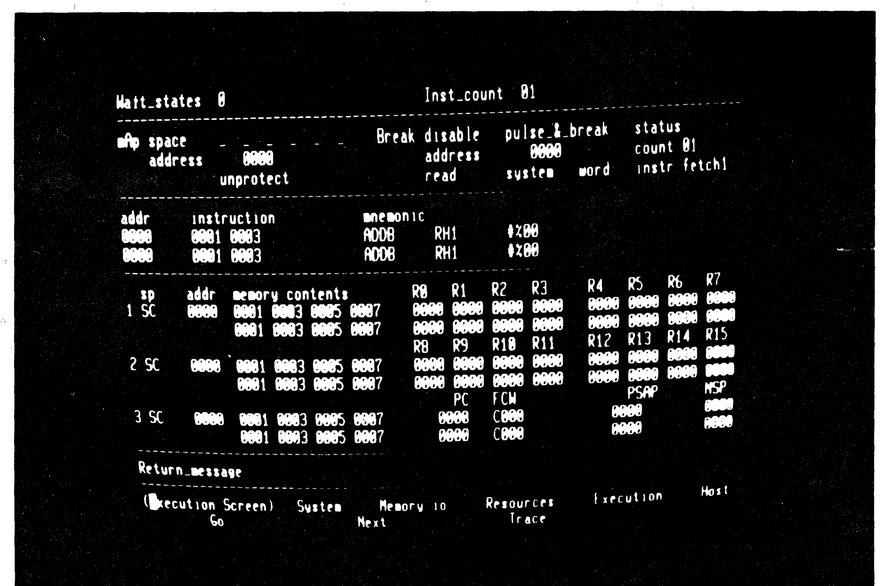


Figure 4-6. Z8002 Monitor Execution Screen

Step	Key Sequence	Commentary
7.	T	One display, the Trace screen, is accessible only from the Execution screen. Notice that there is no menu area because this screen does not support a variety of commands. It is dedicated to providing a detailed picture of program execution.

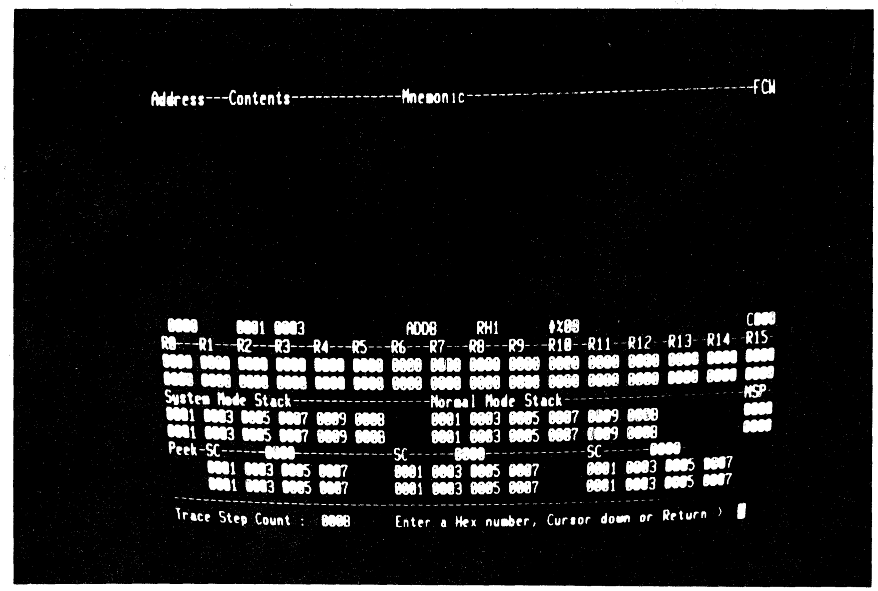


Figure 4-7. Z8002 Monitor Trace Screen

Step	Key Sequence	Commentary
8.	RETURN, **H**	Enter a RETURN to exit from the Trace screen to the Execution screen, then enter H. The Host command selects **Transparent** mode, allowing the terminal to communicate with a host system through Z-SCAN. You can enter the command even if no host is connected.

Figure 4-8. Host Screen, Transparent mode

Step	Key Sequence	Commentary
9.	BREAK	Transparent mode is terminated when the BREAK key is entered. If the System screen does not reappear, consult your terminal documentation -you may have to press another key at the same time as BREAK, or the key may be disabled by an option setting inside the terminal. A monitor reset can be used to end Transparent mode, but its use is not recommended because it destroys any information that was set up inside the Z-SCAN.
10.	R, A	So far the cursor has remained at the bottom of the screen except when the Host command was used. All of the **user-modifiable** fields on the Z-SCAN screens are outside the menu area. The fields are divided into groups, known as **subscreens.** Each subscreen is associated with a particular command and can be entered by keying the capital letter in the command name as it appears in the menu area. Note that as soon as you enter the A command, the first menu line changes to reflect the selected command, and the cursor moves to the top left field in the mAp subscreen.

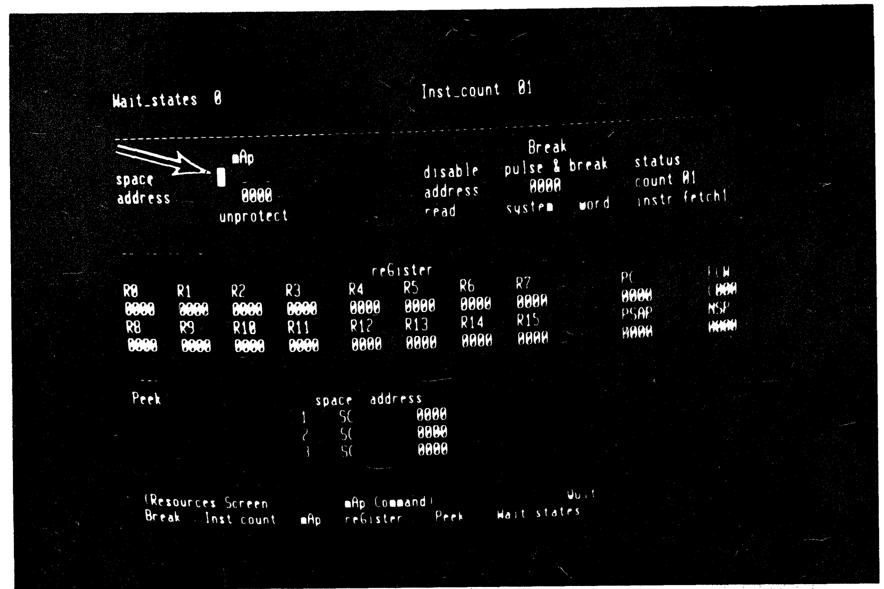


Figure 4-9. Cursor in mAp Subscreen

Step	Key Sequence	Commentary
11. RETURN, B		To move the cursor back to the menu area, enter a RETURN. The menu display does not change because the mAp command is still active. It is altered when a new command, Break, is activated. The cursor moves to the top left field in the Break subscreen.

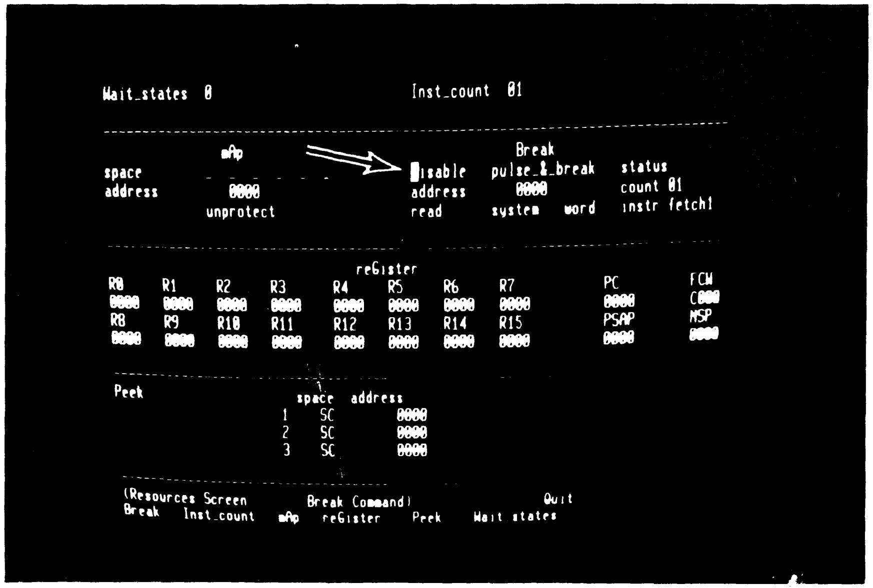


Figure 4-10. Cursor in Break Subscreen.

Step	Key Sequence	Commentary
12.	RETURN, **Q**, S, R, A	You should now be comfortable with activating screens and commands. The only new command in this sequence is Quit. It deactivates the current command and modifies the menu to show the names of the other screens.
13.	RETURN, S, R, **A**	It is not necessary to use the Quit command before moving to another screen. You can enter the initial letter of the new screen name even if it is not currently listed in the menu area.
14.	right, right, right left, left, left, left	Most subscreens consist of more than one field. Once the cursor is in a subscreen, it can be moved to the other fields in the same subscreen by using the cursor control keys. If the cursor left key is entered while the cursor is in the leftmost field,the cursor **wraps around** to the rightmost field in a subscreen line.

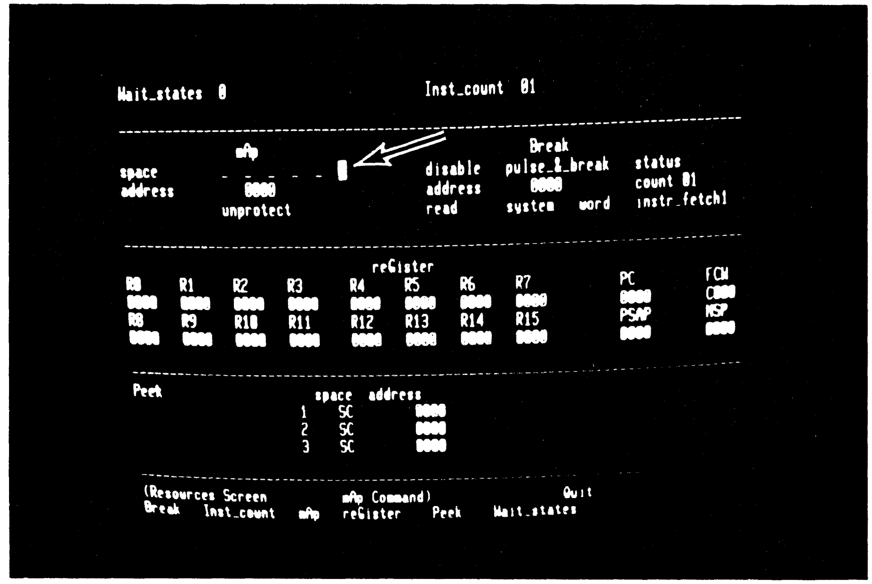


Figure 4-11. Horizontal Cursor Movement

Step	Key Sequence	Commentary
15.	right, down, down, down, up, left, right	The same wrap-around applies in the vertical direction. Note that when there is only one field on a particular line of a subscreen, the horizontal cursor movement keys cannot move the cursor out of that field. The cursor keys can never move the cursor out of the active subscreen.

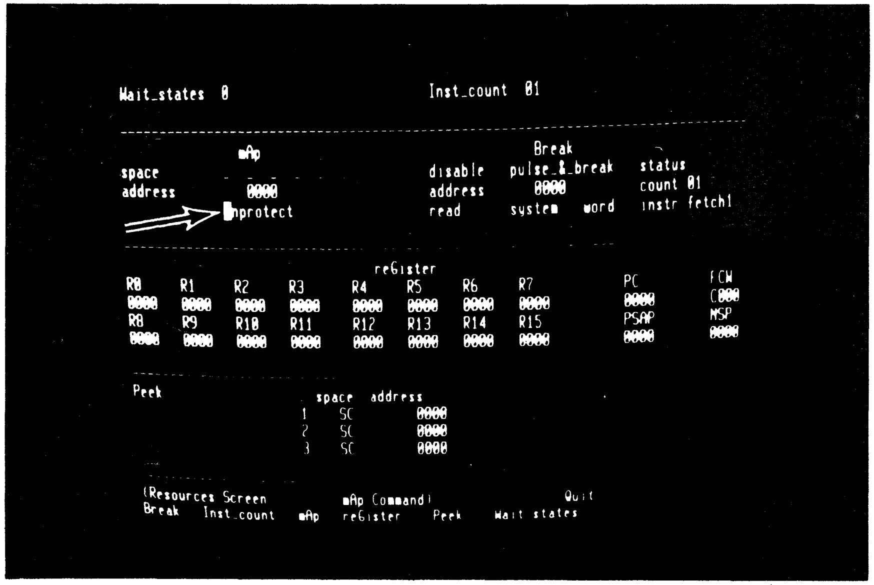


Figure 4-12. Vertical Cursor Movement

Step	Key Sequence	Commentary
16.	RETURN, RETURN	The RETURN key moves the cursor back to the menu area. Because the command remains active, a second RETURN moves the cursor to the top left field in its subscreen: there is no need to re-enter the command name.
17.	>, >, <, 0, 1, space G, F, H, CTRL R, >	Each of the six fields on the first line of the mAp subscreen corresponds to one of the Z8002's address spaces, and each has just two possible values. In the **default** state, an underbar is displayed, indicating that the 8K bytes of **mappable memory** will not respond to CPU accesses made to a particular address space during an emulation. In the alternative state, a two-letter abbreviation for the name of the address space (for example, SC for System Code) shows that the mappable memory will respond. You can step forward or backward through the possible values with the > and < keys or you can access them directly by entering 0 for the first choice and 1 for the second. Alternatively, space and F select the default and final values. CTRL R restores the field to the value it held when the cursor entered it. Other printable characters that are not hexadecimal digits do not affect the field.

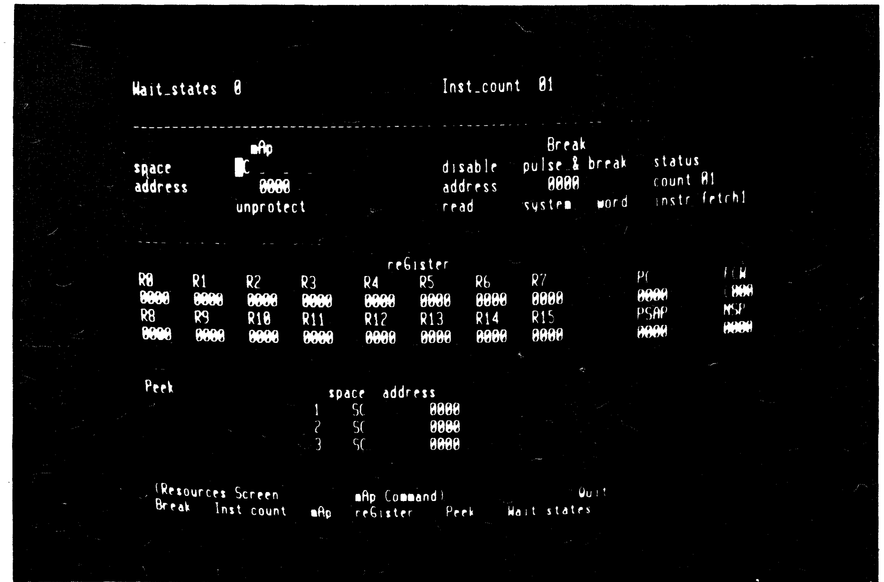


Figure 4-13. Enabling Mappable Memory

Step	Key Sequence	Commentary
18.	RETURN, B, 2	The emulation you are going to run requires a **breakpoint,** so you must enable the breakpoint logic by setting the first field of the Break subscreen to "enable*". This tells the logic to search for a simultaneous match in both the address field and the various status fields.
19.	down, right, >, >, >, <, 1, RETURN	The breakpoint address is not correct and must be changed. The address field contains four hexadecimal digits and can hold any value between 0000 and FFFF. Use > and < to move the cursor within the field, and enter new hex digits to change the value. You have now set a breakpoint that will be triggered when the first word of an instruction is read from system code location 0010.

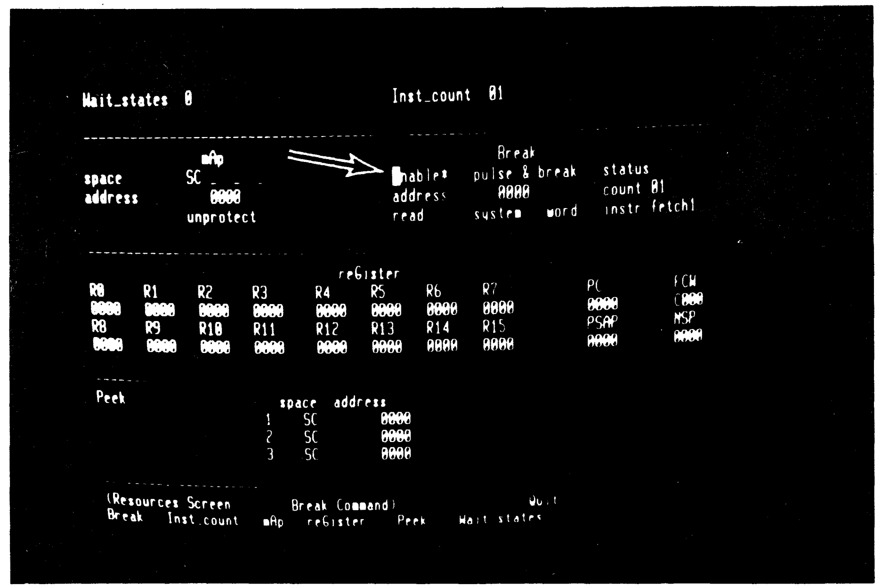


Figure 4-14. Enabling Break Logic

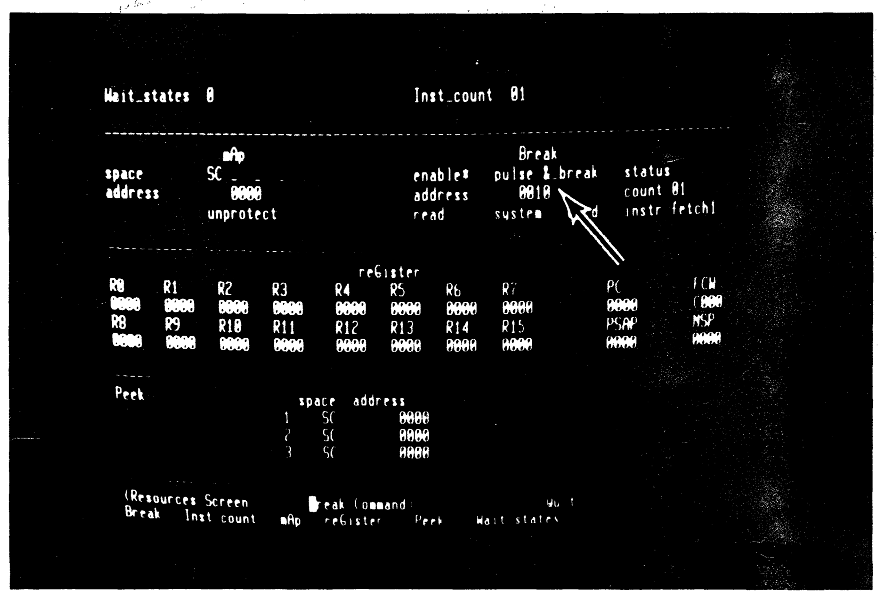


Figure 4-15. Setting Break Address

Step	Key Sequence	Commentary
20.	M	Move to the Memory-io screen. When it is displayed, notice that the top three lines are blank.
21.	F	Fill is listed as a valid command in the menu area. As soon as the command is activated, the cursor moves to the first field of the Fill subscreen, which appears at the top of the screen.
22.	left, 1, F, F, F, down A, 8, 0, B	Use the Fill command to fill mappable memory, which currently extends from address 0000 to 1FFF in system code space, with increment byte register instructions (opcode A80B, mnemonic INCB RH0, #12). In order to do this, you must change the contents of some of the fields on the subscreen. The Fill string can be up to 16 hex digits long, but only four are required in this case.
23.	RETURN	After the parameters have been set up, the command must be executed by entering a RETURN. Before execution starts, the cursor moves to the bottom of the central window area. The message "DONE" is displayed when execution is complete.

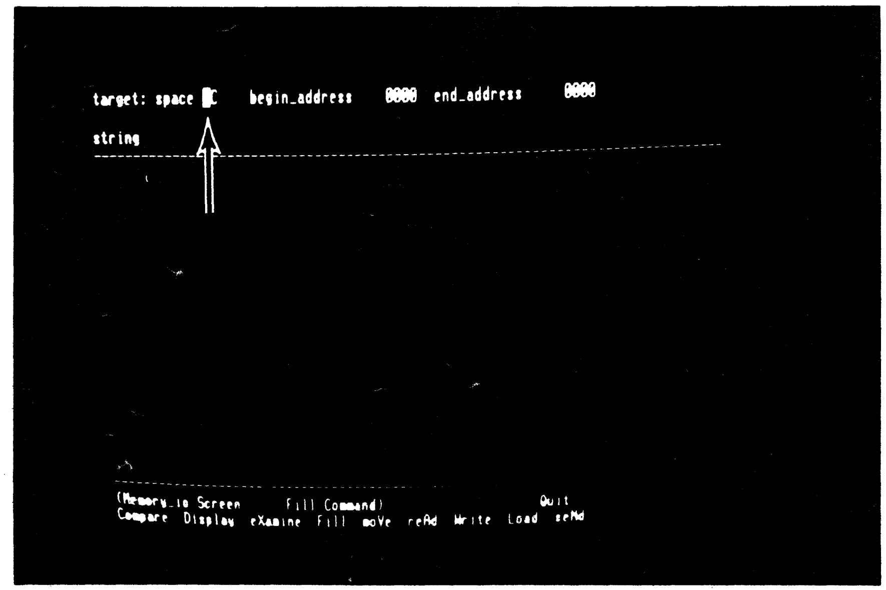


Figure 4-16. Default Fill Command Display

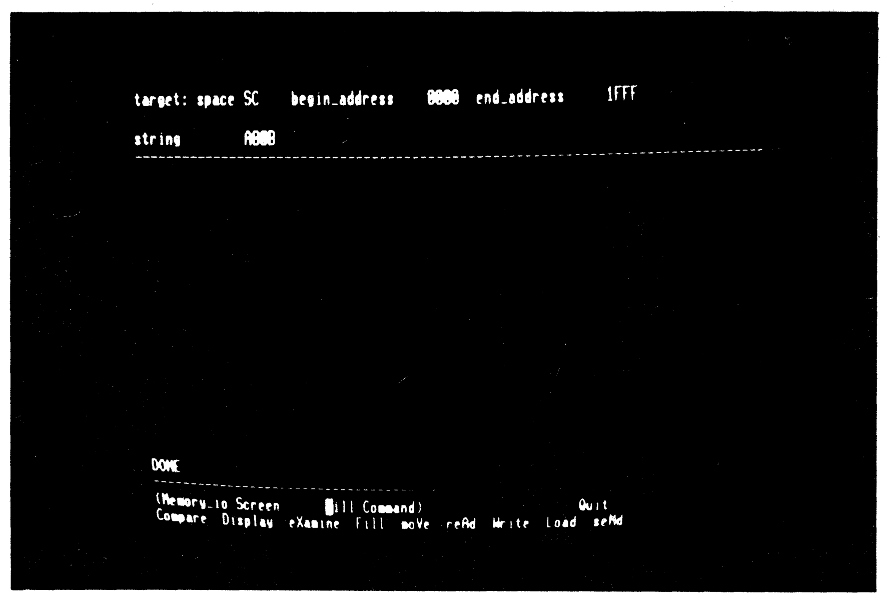


Figure 4-17. Execution of Fill Command

Step	Key Sequence	Commentary
24.	D, RETURN	The Z-SCAN Display command is used to look at the contents of memory. In order to look at the bottom of system code memory, you do not need to change the default parameters that appear at the top of the screen when the command is activated, so execute the command immediately. Addresses appear at the left of the screen, data in the center and at the right is an ASCII representation of the same data. Neither A8 nor 0B corresponds to a printable character. Periods are used to show this. The asterisks are delimiters.
25.	down, up, RETURN	After the Display command has filled the window area, the cursor rests at the bottom right of the screen. You can enter cursor down to display the next block of memory or cursor up to display the previous block. The command is terminated when RETURN is entered.

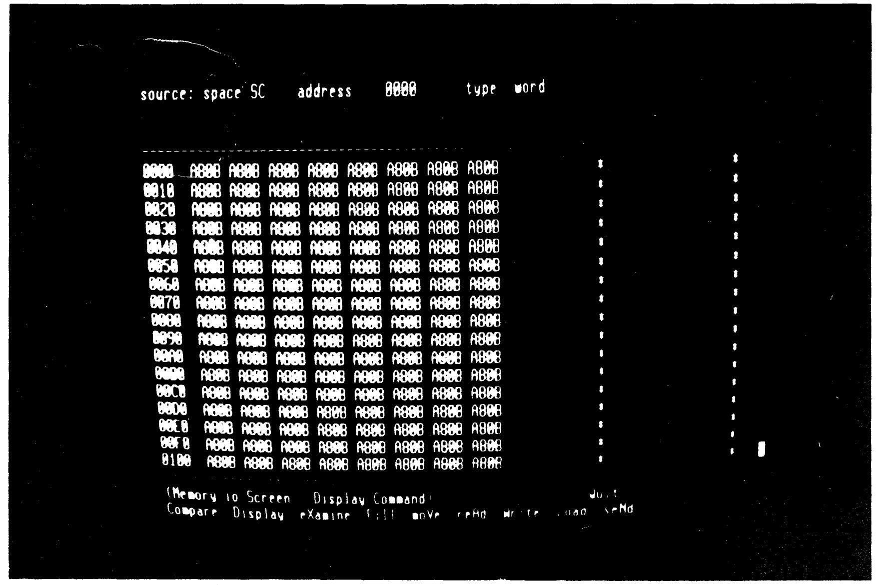


Figure 4-18. Display with Default Parameters

5/27/81

Step	Key Sequence	Commentary
26.	RETURN, left, 3 RETURN, RETURN	The command remains active as long as its name appears inside the parentheses on the menu line, so a second RETURN moves the cursor back into the parameter subscreen. Set the type field so that memory is displayed as **disassembled** nonsegmented Z8002 instructions.
27.	X, right, 1, F, F, C RETURN	The eXamine command allows you to look at and, if desired, modify the contents of memory. Like Fill and Display, it has a private subscreen. The first location you need to examine is the word at system code location 1FFC. Its current contents are displayed when the command is executed, and you are prompted for a new value to replace them.

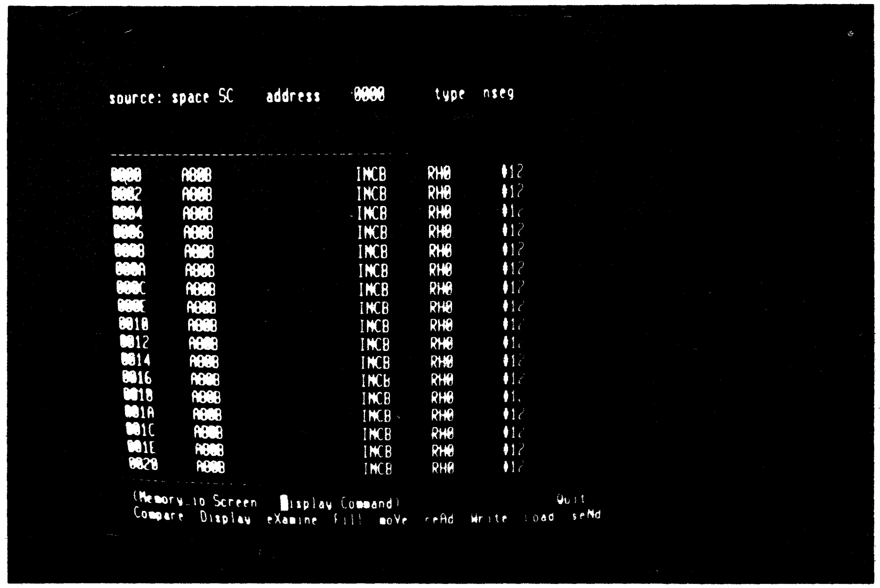


Figure 4-19. Disassembled Memory Display

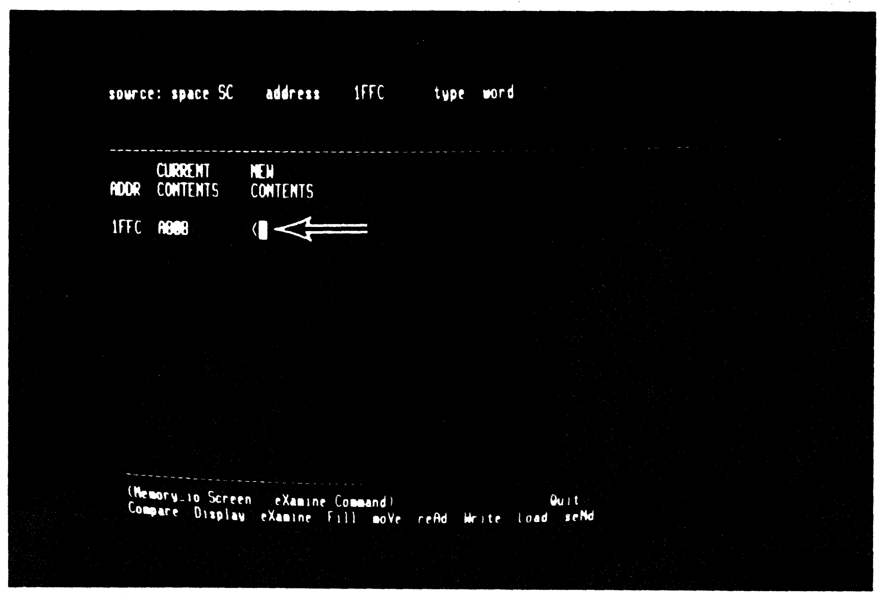


Figure 4-20. Set-up of eXamine Command

Step	Key Sequence	Commentary
28.	5, E, F, <, 0, 8, 1, 8 down	This step replaces the two INCB instructions at the top of mappable memory with an unconditional jump to location 0018 (opcode 5E08 0018, mnemonic JP %0018). The < key can be used to backspace over incorrect input. When sufficient digits have been entered to fill the **open** location, the new value is stored and the next location is opened automatically. The cursor down key opens the next location immediately, storing any digits that have been entered. The data seen in location 2000 may vary because no memory responds at that address.
29.	up, RETURN	Cursor up reopens the previous location, showing that the two digits entered in the previous step have been stored right justified in a field of zeros.

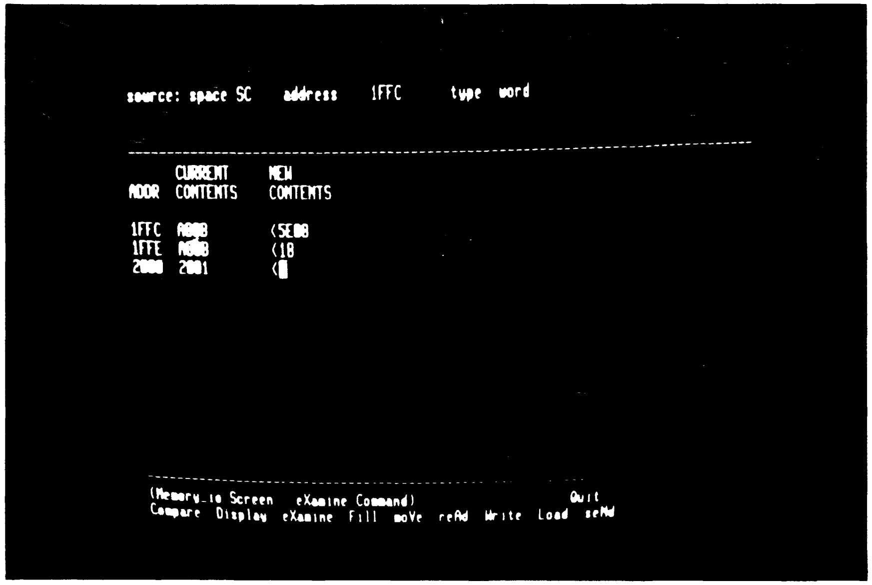


Figure 4-21. Modification of Memory Contents

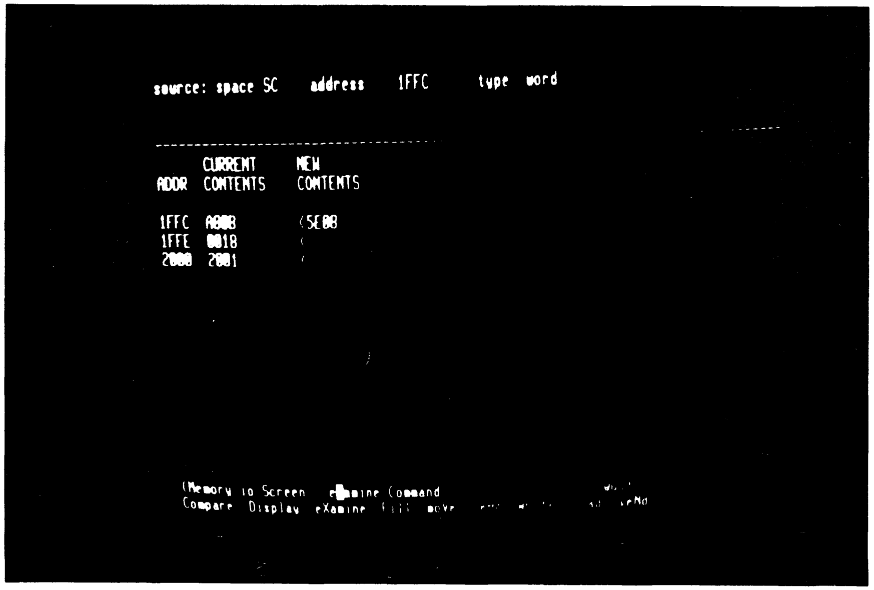


Figure 4-22. Checking Memory Contents

Step	Key Sequence	Commentary
30.	C, left, 1, 0, 0, 0 left, 1, RETURN	Just to check that everything is set up correctly, the Compare command is used to find the differences between the contents of the top and bottom 4K byte blocks of mappable memory. The byte count field for this commands is, like all Z-SCAN monitor numeric fields, hexadecimal (1000 hex = 4096 decimal = 4K). When the command is executed, it should reveal that just four bytes differ between the top and bottom halves of the memory. If it shows anything else, you have probably made a mistake somewhere and not corrected it. Type R, A or, if the cursor is at the bottom right of the screen, RETURN, R, A and repeat the tutorial from step 17.

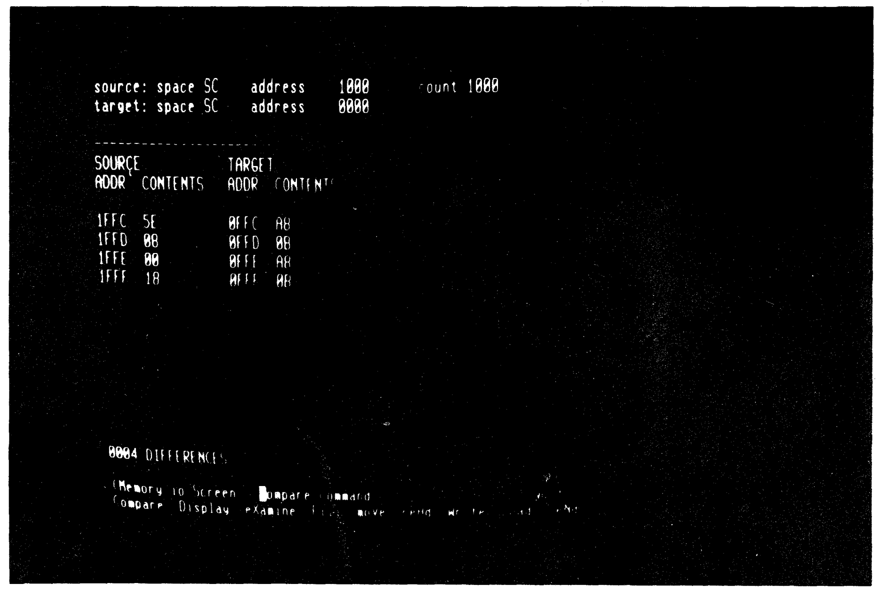


Figure 4-23. Use of the Compare Command

Step	Key Sequence	Commentary
31. E		The program in the mappable memory consists of 4,094 (decimal) INCB instructions and an unconditional jump. It can be run from the Execution screen. The default values of the Program Counter (PC) and Flag and Control Word (FCW) are suitable for running this first emulation. The emulation will run in system mode because bit 14 of the FCW is set. The Z8002 CPU always runs in nonsegmented mode. It ignores bit 15 which selects segmented mode and is also set in the default FCW value provided by the monitor.
32. N		The Next command **steps** through the number of instructions displayed in the Instruction_ count field at the top right of the screen. In this case, the count is one. After the single instruction has been executed, the whole screen is redisplayed, updating the emulation status. Three registers are affected: RH0, the high byte of R0 has been incremented by 12 (decimal); the PC has moved to next instruction and bit 15 of the FCW is now clear because of it cannot be set on the Z8002. The top instruction and register values reflect the state of the program after the emulation, the bottom values the state before.

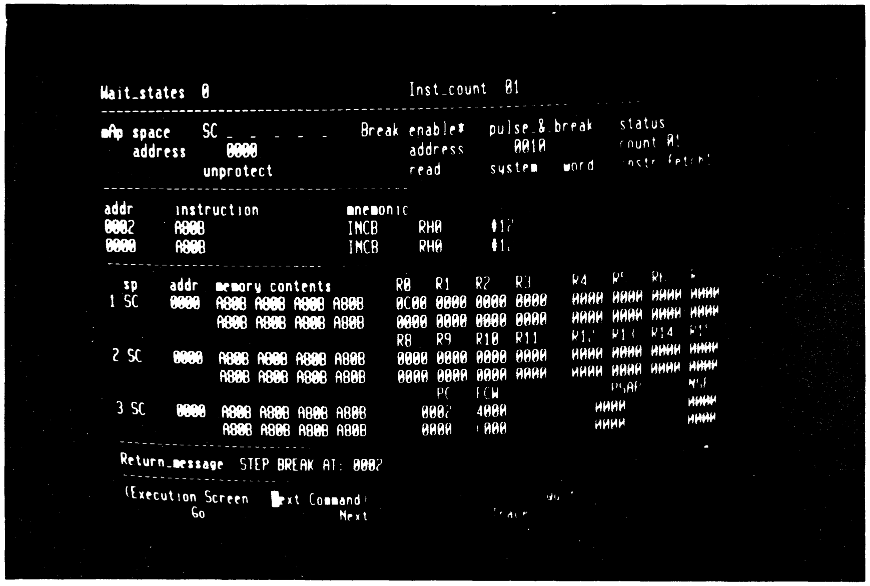


Figure 4-24. Instruction Step with Next Command

Step	Key Sequence	Commentary
33.	RETURN	Now that the Next command is active, it may be repeated by entering return. Again, the PC value changes and RH0 is incremented.

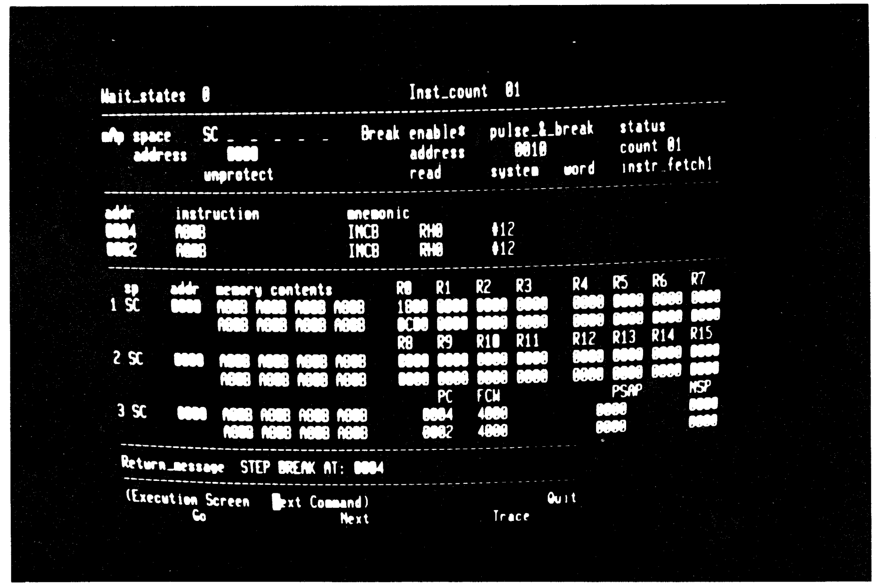


Figure 4-25. Second Instruction Step

Step	Key Sequence	Commentary
34.	G	The Go command starts an emulation which does not stop until a break condition is encountered. Your program should **trigger** the breakpoint logic when an instruction is fetched from location 0010. The breakpoint is honored after the instruction has been executed so that the emulation ends with the Program Counter pointing to the instruction at location 0012. Note that the termination message is different from that of the Next command.

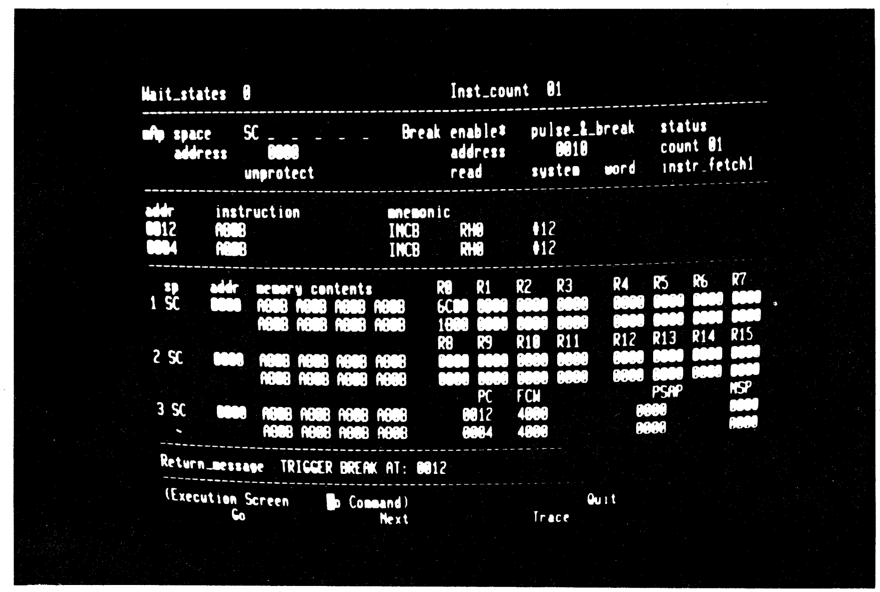


Figure 4-26. Running to Breakpoint with Go Command

Step	Key Sequence	Commentary
35.	T	Emulations can also be run from the Trace screen, which disassembles each instruction before it is executed. The instruction which appears in the center of the screen is the first to be executed when emulation starts. The bottom of the screen displays register and memory contents. The function of these fields will be explored later.
36.	down	Entering cursor down results in the execution of the number of instructions given in the count field at the bottom left of the screen. PC and FCW values are given for each instruction executed, and the first instruction executed is flagged with an asterisk in column 1. The remaining registers are not redisplayed until all the instructions have been executed. The FCW values at the right of the screen show that the value in RHO has overflowed and become negative.

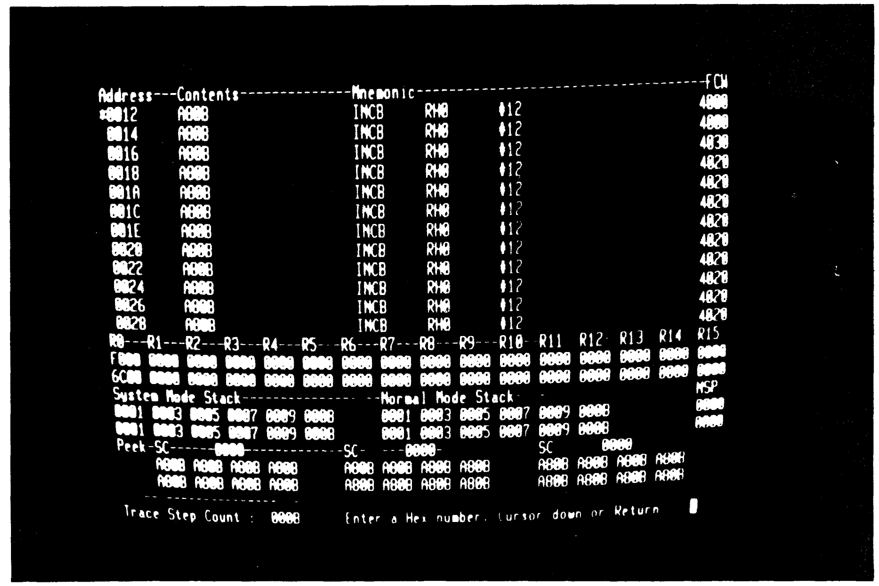


Figure 4-27. Use of the Trace Screen

Step	Key Sequence	Commentary
37.	RETURN, **G**	Return from the Trace screen to the Execution screen and start another emulation with the Go command. This time the breakpoint is not encountered - the program loops in the address range 0018 to 1FFC, avoiding location 0010. While the emulation is running, the cursor rests in the blanked return message line and the terminal keyboard is disabled.
38.	monitor NMI	The monitor NMI signal acts as a **manual break** request during emulations run from the Execution screen. The emulation terminates when execution of the current instruction is complete. The break address and register contents you see will probably be different from those in the photograph, but this does not matter.

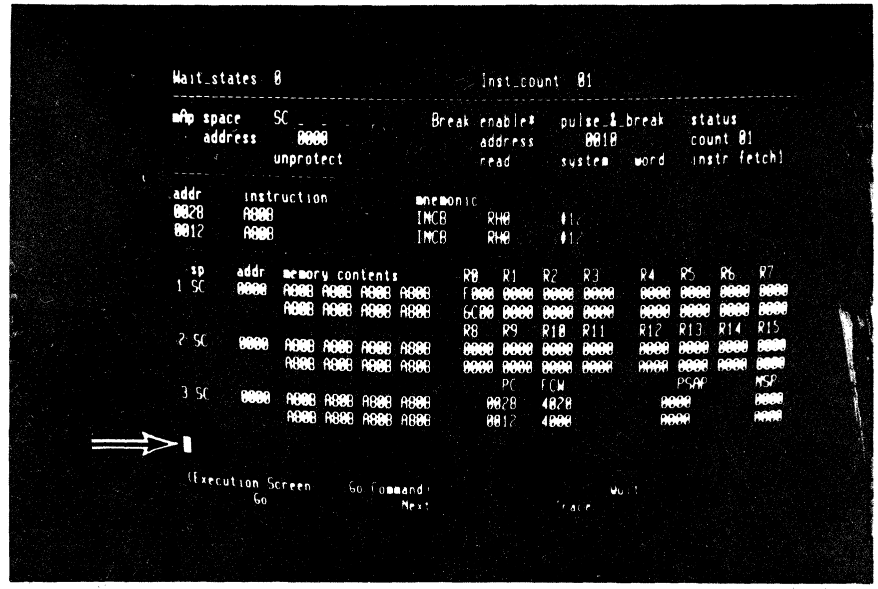


Figure 4-28. **Indefinite Emulation with Go Command**

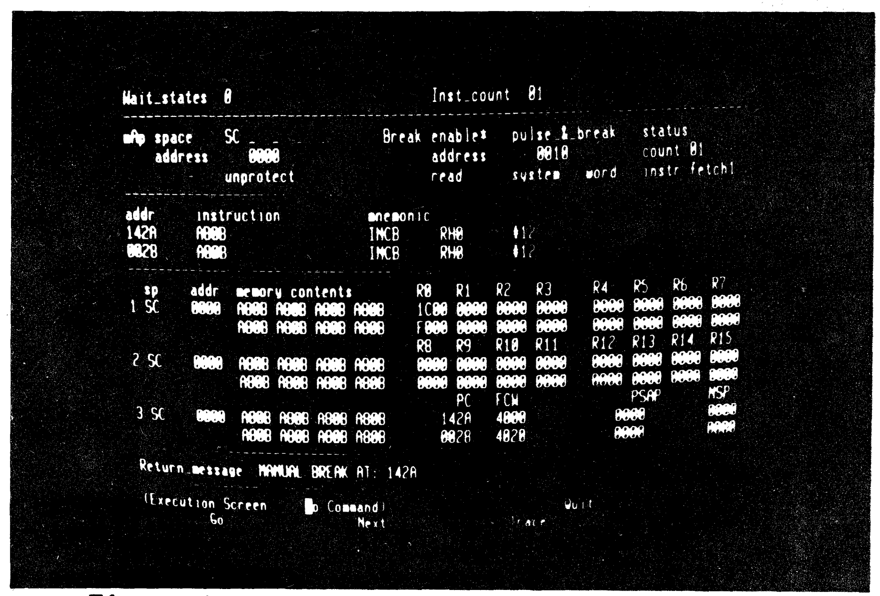


Figure 4-29. **Manual Break with NMI Switch**

Step	Key Sequence	Commentary
39.	M, X, right, space CTRL R, 0, 0, 1, 8	To explore further facilities offered by Z-SCAN, an instruction which reads and writes memory is required. Use the Memory_io screen eXamine command to insert an instruction at location 0018. Two of the keystrokes in this sequence are redundant. The space restores the address field to its default value and CTRL R cancels any changes made since the cursor entered the field.
40.	RETURN, 6, 9, 0, F, 1 0, RETURN	The instruction is "INC %0010,#16" (increment by 16 the word at location 0010). It has a two-word opcode, 690F 0010.
41.	C, BREAK, RETURN, left 0, RETURN	Check memory contents again by using the Compare command. Extra keystrokes in this sequence show that the BREAK key moves the cursor back to the menu area without executing the active command and that the monitor does not allow you to enter an illegal value in a numeric field: the previous value of the field is restored. When the command is executed it should show eight differences.

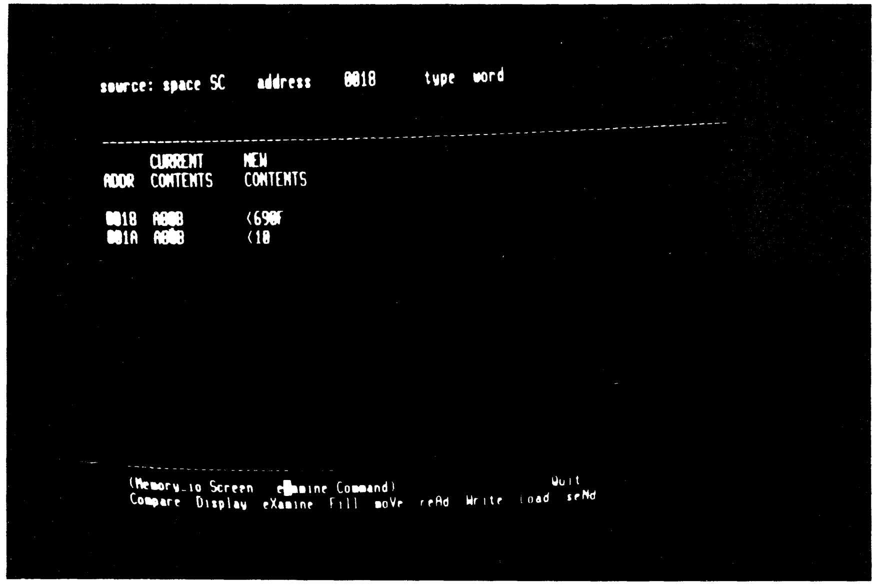


Figure 4-30. Insertion of New Instruction

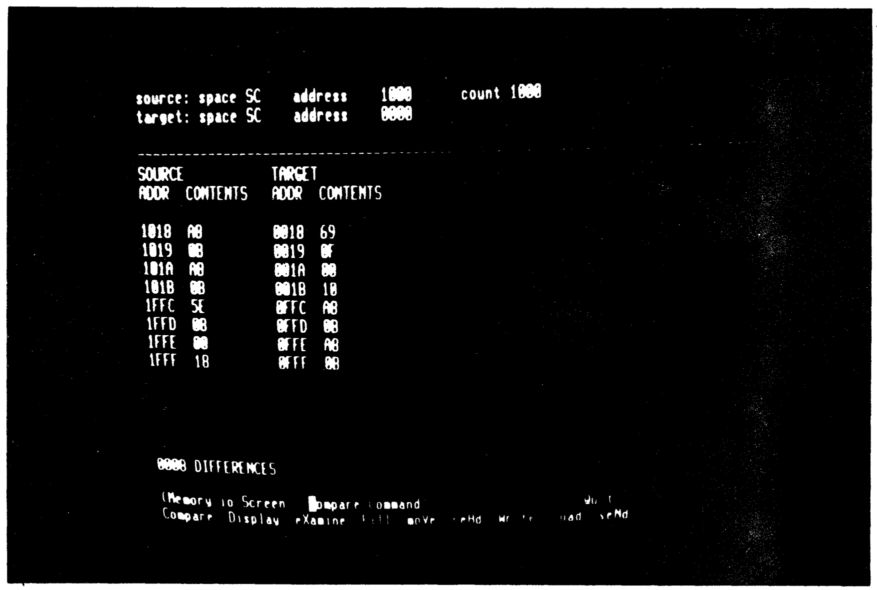


Figure 4-31. Check of Change with Compare Command

Step	Key Sequence	Commentary
42.	D, RETURN, RETURN	Display disassembled memory to show the new instruction at location 0018.
43.	R, P, right, >, >, 1, RETURN	The added instruction modifies the contents of location 0010 each time it is executed, so it is desirable to know how they have changed after each emulation is run. Z-SCAN displays the contents of selected locations on the Execution and Trace screens. The monitored addresses are set up by the Peek command on the Resources screen. Modify the first of the three addresses to 0010.

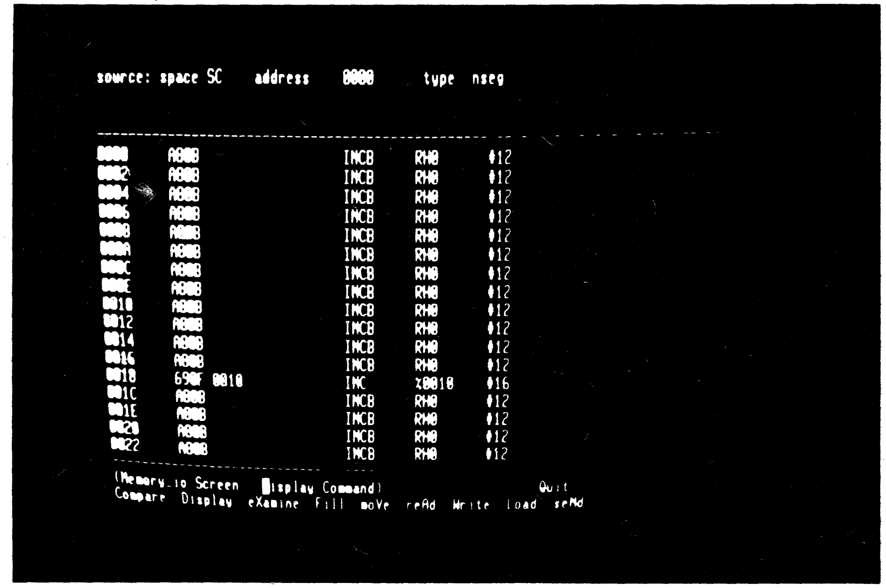


Figure 4-32. Display of Change

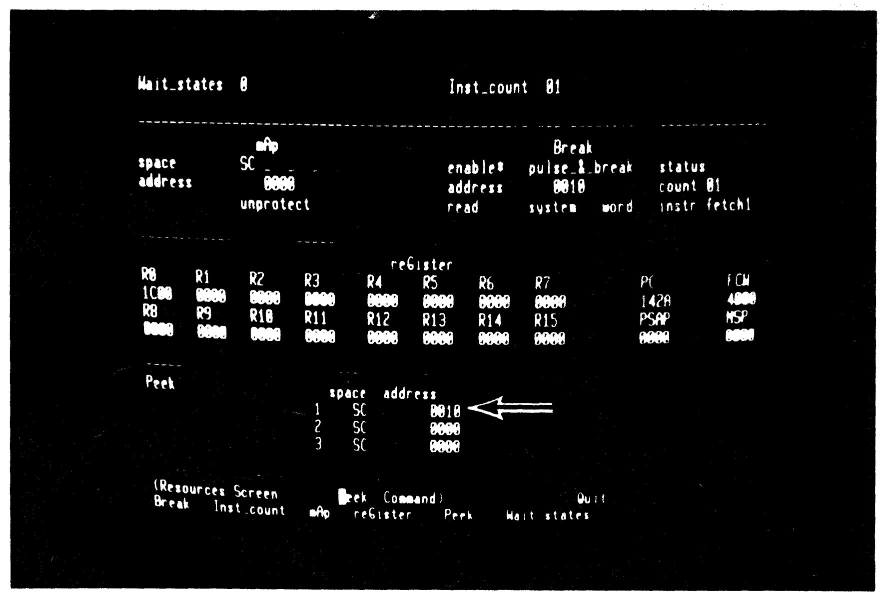


Figure 4-33. Setting Peek Parameters

Step	Key Sequence	Commentary
44.	E, G	Now call up the Execution screen and start an emulation. The top line of the first Peek field shows the contents of word locations 0010 through 0016 as they were before the emulation started.
45.	monitor NMI	You might think that this emulation should stop with a trigger break, because location 0010 is being read by the new instruction. The trigger logic does not fire because the break parameters are set up for an instruction fetch, not a data read, so the emulation must be terminated with a manual break. Looking at the Peek memory areas, you see that the contents of location 0010 have not changed during the emulation. Remember that the mappable memory has been set to respond only to system code space accesses. This explains why the system data accesses made by the new instruction do not affect it.

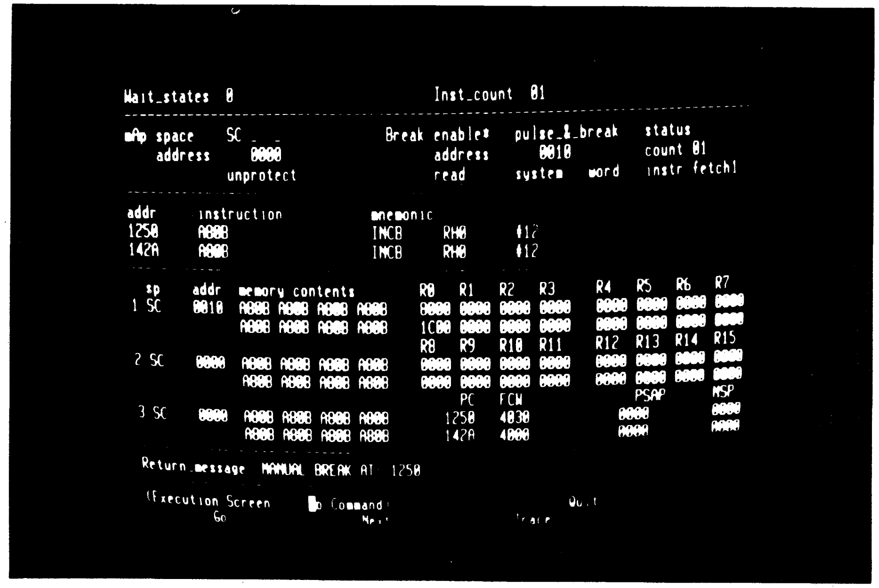


Figure 4-34. Second Manual Break

Step	Key Sequence	Commentary
46.	R, B, up, left, >, >, F, <, space, 1, 2, 9, RETURN	To fix these two problems, leave the Execution screen, which, though it displays data about mappable memory and the break condition, does not allow you to modify the parameters. Use the Resources screen Break command to set up a breakpoint on a data memory request. This is one of 16 possible values in the bus cycle type field. As usual, you can select a choice either by stepping through the table of possible values or by entering a number that corresponds to the required choice. The space character selects the default value.
47.	A, right, 1, RETURN	The second field in the mAp subscreen determines whether or not the mappable memory responds to system data accesses. Entering a 1 sets the field to "SD". The mappable memory now responds to two types of accesses. For this reason, it is not necessary to modify the memory space parameters of Peek. System code location 0010 is the same memory word as system data location 0010.

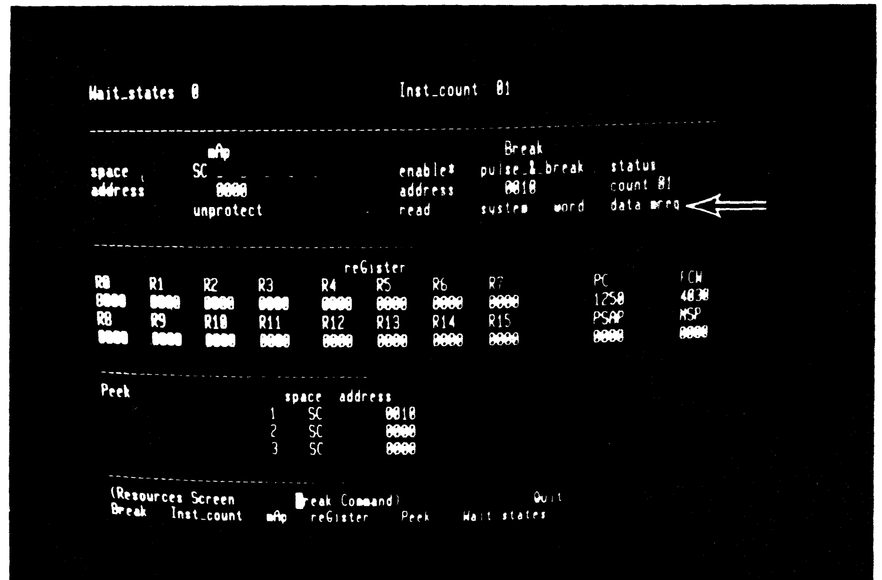


Figure 4-35. Modification of Break Parameters

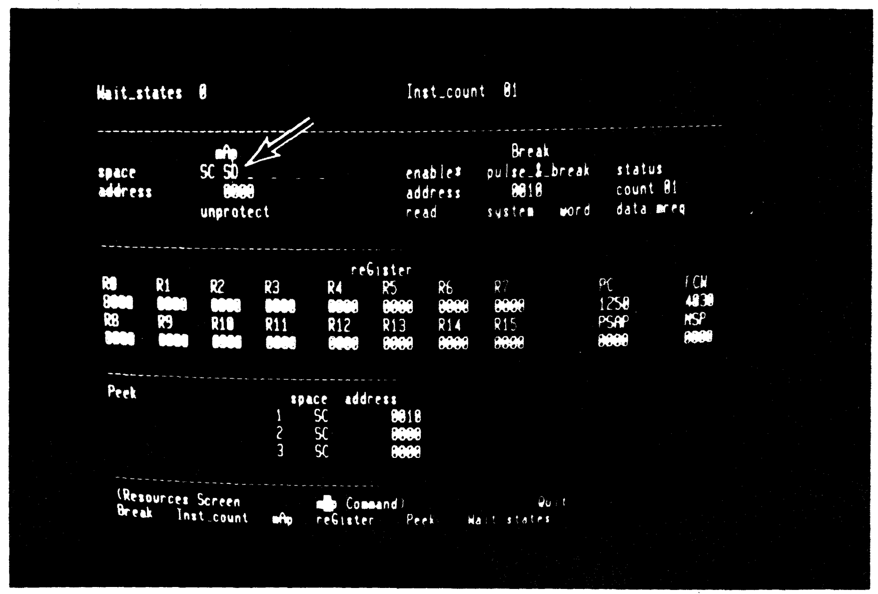


Figure 4-36. Modification of mAp Parameters

Step	Key Sequence	Commentary
48.	G, A, B, RETURN	The last action on this screen is to set up a new starting value for R0 with the reGister command.

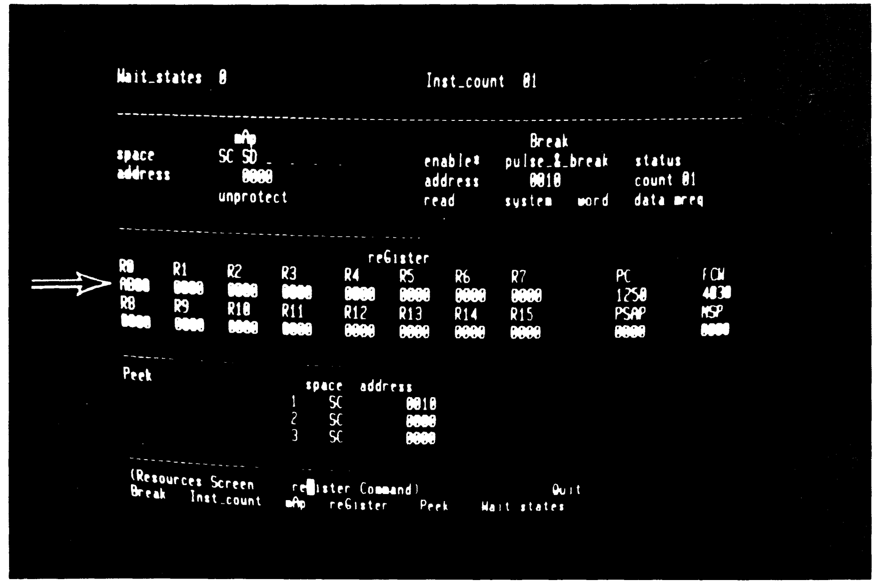


Figure 4-37. Modification of R0 Value

Step	Key Sequence	Commentary
49.	E, G	Start a new emulation. This time, the trigger fires almost immediately, and when the execution screen is redisplayed, you see that the contents of location 0010 have indeed changed from A80B to A81B. The Program Counter points to location 001C, the word after the instruction that caused the break condition to be met. The condition flags in the FCW reflect the fact that location 0010 holds a negative 2's complement number.

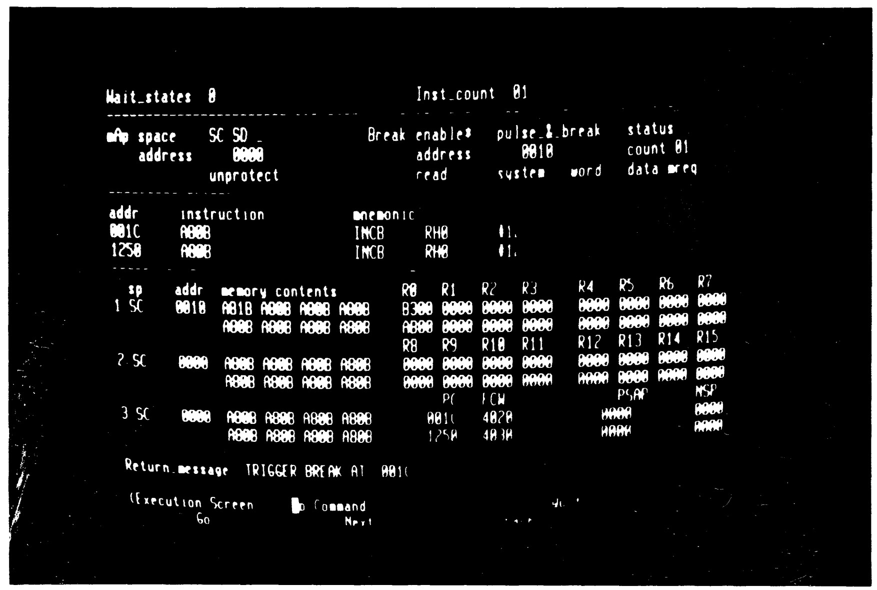


Figure 4-38. Trigger Break on Data Read

Step	Key Sequence	Commentary
50.	R, B, down, left, 5 RETURN	Associated with the breakpoint logic is a **pass counter.** If you load it with 51 hex (that is 81 decimal), the program loop is executed that number of times on the next emulation.
51.	E, G	After the emulation begins, there is a short delay before the breakpoint is encountered the number of times programmed. When the emulation ends, location 0010 has been incremented by 510 hex (51 x 10), showing that the correct number of passes has been made.

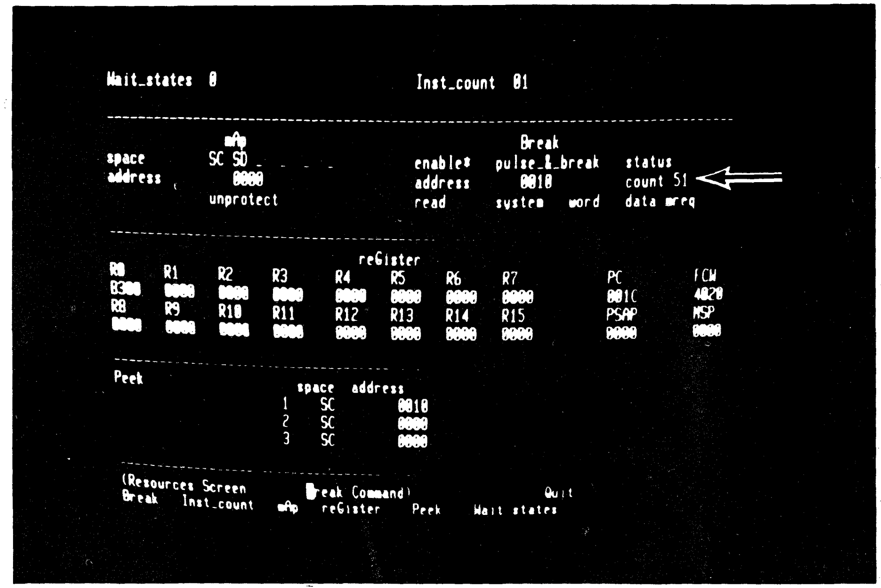


Figure 4-39. Adjusting Pass Counter

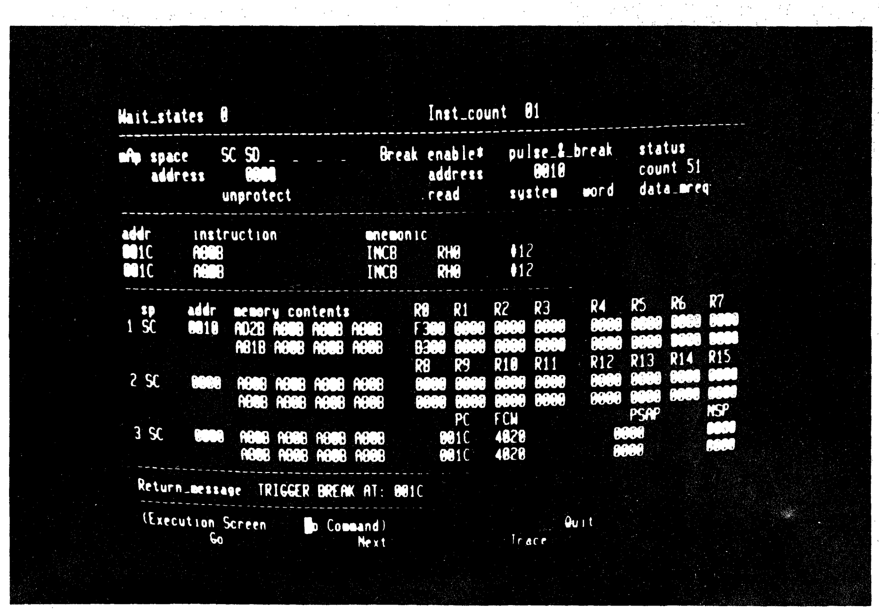


Figure 4-40. Break After Multiple Passes

Step	Key Sequence	Commentary
52.	R, A, up, 2, RETURN, B, space, RETURN	The INC instruction writes memory and can be used to show the Z-SCAN's **write protect** feature. To do this, disable the breakpoint and enable a write protect break.
53.	E, G	The next emulation terminates with a message warning of a write protect **violation.** Although the offending instruction has been executed, the contents of mappable memory remain unchanged and the data that the CPU attempted to write into memory is lost.

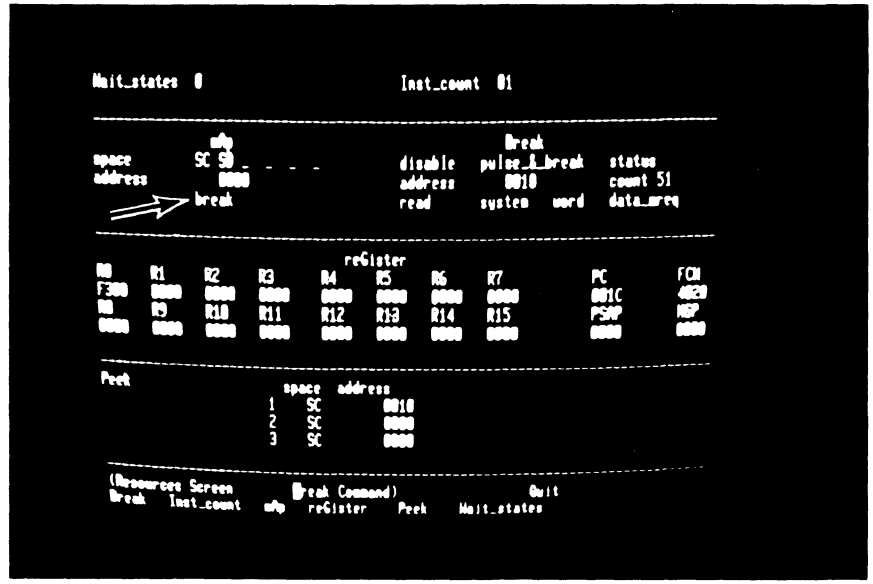


Figure 4-41. Selection of Write-Protect Break

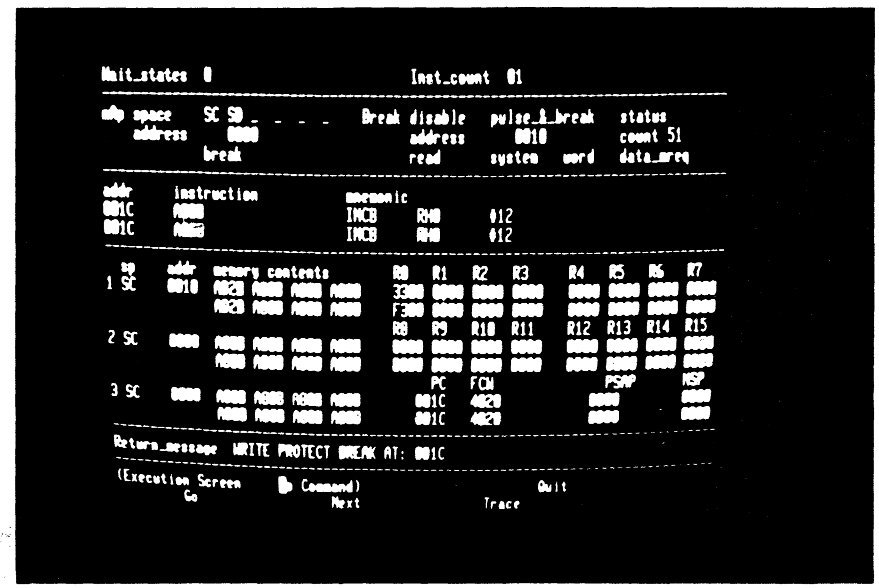


Figure 4-42. Break After Violation

Step	Key Sequence	Commentary
54.	R, A, up, space, RETURN	Clear the write protect break.
55.	B, 1, down, left, space, left, 1, F, F, A, RETURN	Now select a break on the first occurence of either any reference to location 1FFA (in any address space) or any word read from system data memory. "enable+" designates this mode of operation.
56.	E, G	Return to the Execution screen and run an emulation. It stops at location 1FFC because the address of the previous instruction has fired the trigger. The contents of location 0010 are unchanged, indicating that the instruction at location 0018 was not executed during the emulation.

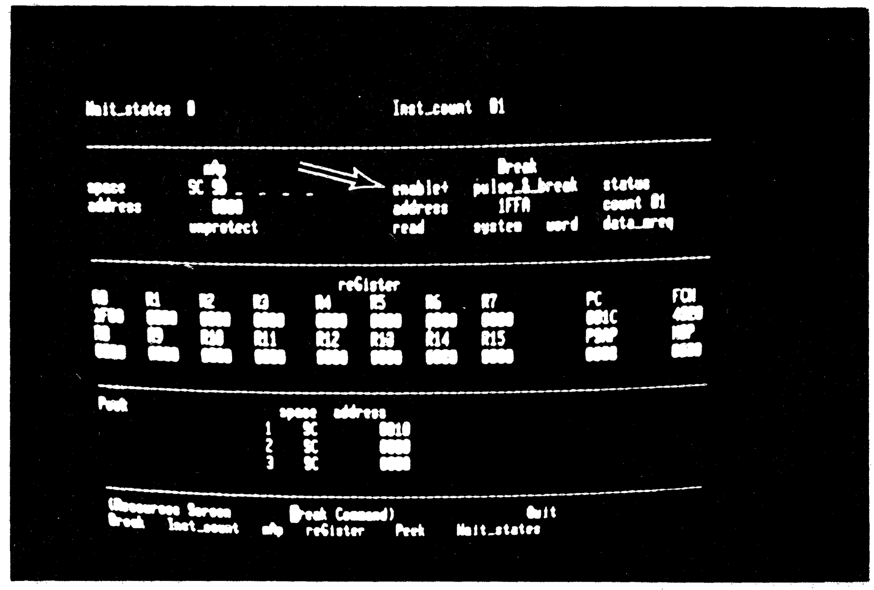


Figure 4-43. Set-up of Multiple Condition Break

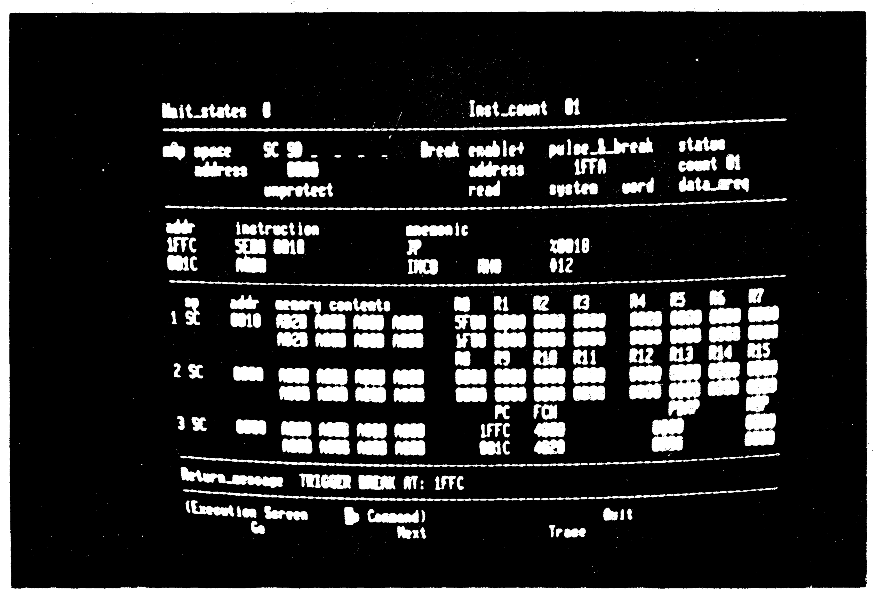


Figure 4-44. Break on Address Match

Step	Key Sequence	Commentary
57.	T, down	A trace terminates after only two instructions have been executed because a trigger is caused when the instruction at location 0018 performs a data memory read. Emulation stops as this event has precedence over the step count of 000B (11 decimal) instructions. A break message replaces the prompt that normally appears on the bottom screen line. The Peek display shows that the contents of location 0010 have changed.

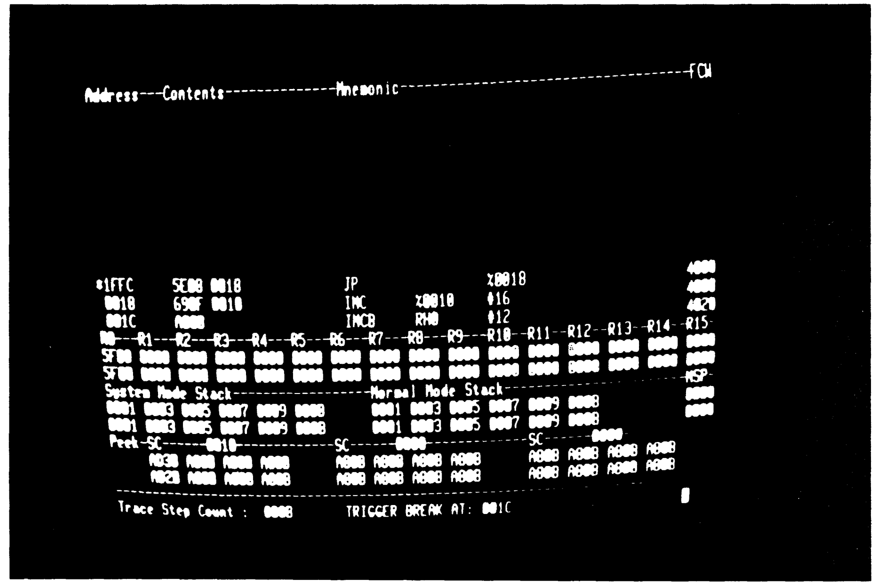


Figure 4-45. Data Read Break on Trace Screen

4.6 HOST SYSTEM USE WITH Z8002

The tutorial script continues on the next page. If your Z-SCAN is connected to a host system that supports the generation and downloading of Z8002 programs, perform steps 59 through 63, then move on to step 65. If the example program already exists on the host file system, you can skip all the steps except 63. If you do not have a suitable host, proceed directly to step 64.

Step	Key Sequence	Commentary
58.	RETURN, R, RETURN, R, A, 1, right, 1, right 1, right, 1, right, 1, right, 1, RETURN	The example program that is run in this part of the tutorial generates accesses to all six Z8002 memory spaces. Select the Resources screen and set up the mappable memory to respond to all types of access: code, data and stack references in both system and normal modes.

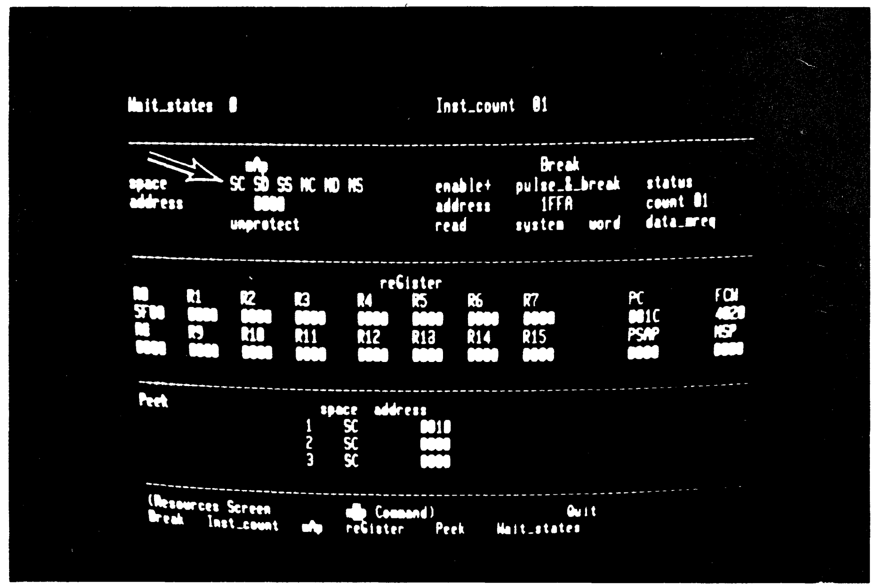


Figure 4-46. Enabling of All mAp Address Spaces

NOTE

If your Z-SCAN is connected to a host system that supports the generation and downloading of Z8002 programs, perform steps 59 through 63, then move on to step 65. If the example program already exists on the host file system, you can skip all the steps except 63. If you do not have a suitable host, proceed directly to step 64.

Step	Key Sequence	Commentary
59.	H	Before you can use the Z-SCAN **download** command, you must have a Z8002 program to load. Your host's utilities and support programs can be used to create it. Type H to enter Transparent mode.
60.	Bootstrap your system	Unless it is already up and running, load the operating system of your host. For Zilog PDS 8000 systems, press the reset button on the front panel of the system, then enter RETURN at the terminal keyboard. For ZDS/1 systems, press wait, then enter two returns. An operating system diskette must be present in drive zero or, for hard disk systems, the disks must be spinning. If you have a non-Zilog host, follow the bootstrap procedure described in its system manual.

Step	Key Sequence	Commentary
61.	Enter, assemble and image the example program	Figure 4-47 shows an example program that is compatible with Zilog's Z8000 PLZ/ASM assembler, version 2.02 or later. The commands needed by the Zilog RIO operating system to create it are listed in Figure 4-48. Assemblers on non-Zilog hosts probably require changes in the syntax of the source. Changes are acceptable provided that the memory image of the final program corresponds to the information at the left of Figure 4-47. Refer to the host documentation for more information. The program appears with expanded commentary in Appendix B of this manual.
62.	BREAK	Return to the Z-SCAN monitor environment.

```
LOC      OBJ CODE     EXAMNSG MODULE
                          $SECTION     EXAMNSG_P        ! Make imaging easy    !
                          $REL         %0000
                      INTERNAL
                      NEW_STATUS_AREA:                  ! Most entries unused !
                          $REL         %0002            ! Reset status         !
0002 4000   002A'         RESET        ARRAY [ 2 WORD ] := [ %4000, INIT ]
                          $REL    %0008                 ! Privileged instr.    !
0008 4004   002A'         PRIV_VECTOR ARRAY [ 2 WORD ] := [ %4004, INIT ]
                          $REL    %000C                 ! System call          !
000C 4000   003C'         SC_VECTOR    ARRAY [ 2 WORD ] := [ %4000, BREAKER ]
                          $REL    %0014                 ! Non-Maskable Int.    !
0014 4008   002A'         NMI_VECTOR   ARRAY [ 2 WORD ] := [ %4008, INIT ]

0018 0000   0000          PASS, LAST   WORD := 0        ! Data and stack area !
001C 0000   ...           NML_STK      ARRAY [ 4 WORD ] := 0
0024 0000   0000          SYS_STK      RECORD [ ID OLD_FCW OLD_PC WORD ] := 0
0028 0000
002A                  GLOBAL INIT PROCEDURE             ! Set up control reg's!
                      ENTRY                             !  and both stacks.    !
002A 7600   0000'         LDA      R0,NEW_STATUS_AREA
002E 7D0D                 LDCTL    PSAP,R0
0030 210F   002A'         LD       R15,#SYS_STK + SIZEOF SYS_STK
0034 7600   0024'         LDA      R0,NML_STK + SIZEOF NML_STK
0038 7D0F                 LDCTL    NSP,R0
003A 7F12                 SC       #%12                 ! Trap into BREAKER    !
003C                  END INIT

003C                  INTERNAL BREAKER PROCEDURE        ! Demonstrate bus      !
                      ENTRY                             !  cycle types.        !
003C A9F5                 INC      R15,#SIZEOF SYS_STK ! Fix up system stack !
003E 670E   0026'         BIT      SYS_STK.OLD_FCW,#14 ! Check previous mode !
0042 E604                 JR       Z,ELSE_              ! If mode was system   !
0044 7D02                     LDCTL    R0,FCW           !  set normal mode by !
0046 A30E                     RES      R0,#14           !  clearing bit 14     !
0048 7D0A                     LDCTL    FCW,R0           !  of FCW;             !
004A E808                     JR       FI_              !  else do I/O.        !
004C 2101   ABCD      ELSE_:  LD       R1,#%ABCD        ! Dummy port address. !
0050 3D12                     IN       R2,@R1           ! I/O read             !
0052 3F13                     OUT      @R1,R3           ! I/O write            !
0054 3B05   1234              SIN      R0,%1234         ! Special I/O read     !
0058 3B37   1234              SOUT     %1234,R3         ! Special I/O write    !
                      FI_:                              ! Memory op's follow: !
005C 7602   0018'         LDA      R2,PASS              ! Internal operation   !
0060 2124                 LD       R4,@R2               ! Data read            !
0062 93F4                 PUSH     @R15,R4              ! Stk write            !
0064 29F0                 INC      @R15                 ! Stk read, stk write !
0066 57F0   0018'         POP      PASS,@R15            ! Stk read, data write!
006A 3304   FFAC          LDR      LAST,R4              ! Code write           !
006E 7FEF                 SC       #%EF                 ! Trap sequence        !
0070                  END BREAKER

                      END EXAMNSG
```

Figure 4-47. Z8002 Program Example

```
>
COPYRIGHT, ZILOG, INC. 1979
             All rights reserved.
             No part of this software may be copied or used without
             the express written consent of ZILOG, INC.

THURSDAY, NOVEMBER  1, 1979
RIO REL 2.2
%DATE 810424
FRIDAY, APRIL 24, 1981
%B;SET TABSIZE=4;EDIT EXAMNSG.S
B
EDIT 2.1
NEW FILE
INPUT
EXAMNSG MODULE
    $SECTION    EXAMNSG_P       ! Make imaging easy    !
    $REL        %0000
INTERNAL

        . . . . .

    SC      #%EF                ! Trap sequence        !
END BREAKER

END EXAMNSG

EDIT
>QUIT
%Z8000ASM EXAMNSG
Z8000ASM 2.02
Pass 1 complete

   0 errors
Assembly complete
%IMAGER EXAMNSG.OBJ ($=0000 EXAMNSG_P) {0000 0080} E=002A O=EXAMNSG
IMAGER 2.0
68 BYTES LOADED
%EXTRACT EXAMNSG
RECORD COUNT = 0001  RECORD LENGTH = 0200   NO. OF BYTES IN LAST RECORD = 008
ENTRY POINT = 002A  LOW ADDRESS = 0000  HIGH_ADDRESS = 0080  STACK SIZE = 000
SEGMENTS:
0000 007F
%
```

NOTE

If the file EXAMSEG is created on a diskette, the record length shown by the EXTRACT command will be 0080.

Figure 4-48. Z8002 Program Creation with RIO

Step	Key Sequence	Commentary
63.	M, L, down, E, X, A, M, N, S, G, RETURN	Set up and execute the Memory-io screen Load command. The program name is EXAMNSG (nonsegmented example), and it is to be loaded into system code memory. As the file is loaded, an incrementing number field appears toward the top left of the screen. This is a count of the number of records transferred from the host to target memory. Each record carries 30 or fewer bytes. When the loading is complete, the entry address of the program is displayed. If any error message appears, enter H and check the following: o Does the program file EXAMNSG exist? o Is its name correct? o Does the download utility LOAD exist? If no message appears when the command is executed, the host has not responded to the Load command sent by Z-SCAN. Terminate the load by entering BREAK, then type H and establish why this happened. When you have fixed the fault, return to the Z-SCAN monitor environment and type M, L, return.

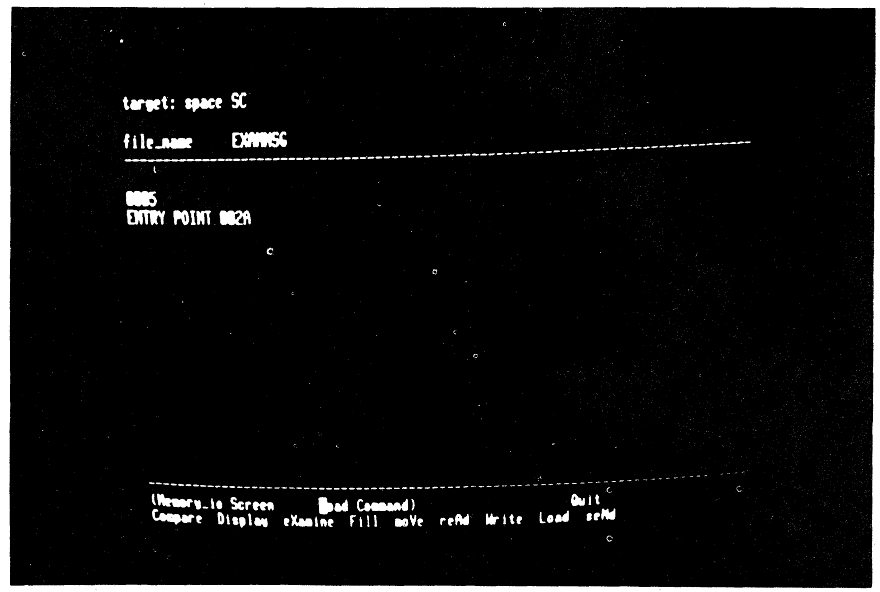


Figure 4-49. Loading of Z8002 Example Program

Step	Key Sequence	Commentary

If you have completed the four previous steps, skip the next one.

64.	M, V, 6, right, 5, F, 8, 0, right, 0, 0, 7, 0, RETURN	A copy of the example program shown in Figure 4-47 exists in the Z-SCAN monitor ROM. Use the Memory_io screen moVe command to copy it into the mappable memory.

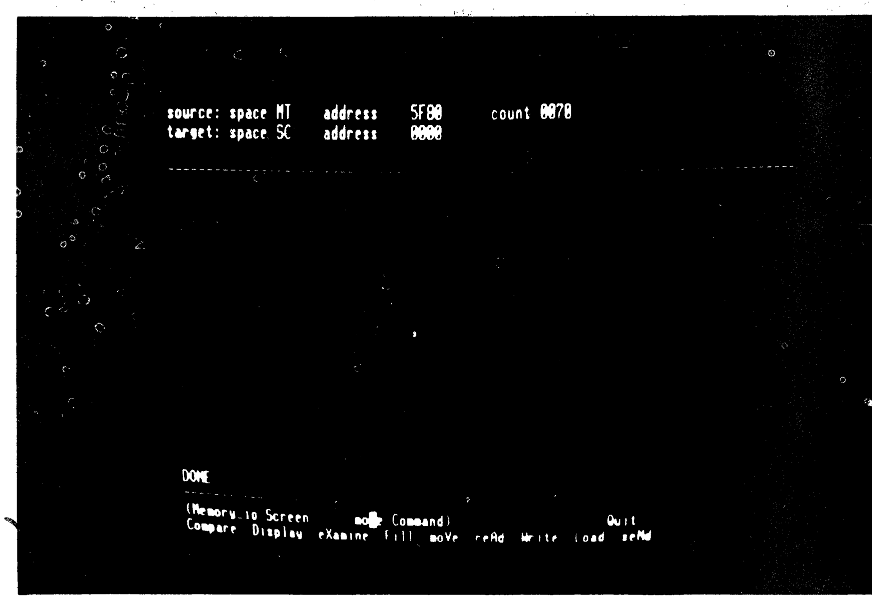


Figure 4-50. Copying Program with move Command

Step	Key Sequence	Commentary
65.	R, G, space, left 4, 0, 0, 0, left, 0, 0, 2, A, RETURN	Set up the PC and FCW so that the program starts at location 002A in system mode. At the same time, restore R0 to its default value.
66.	P, 1, right, 0, 0, 1, 8, down, 0, 0, 1, C, left, 5, down, 2, right, 0, 0, 2, 4, RETURN	The program has data, normal stack and system stack areas. Set up the Peek fields to monitor their contents before and after each emulation is run.

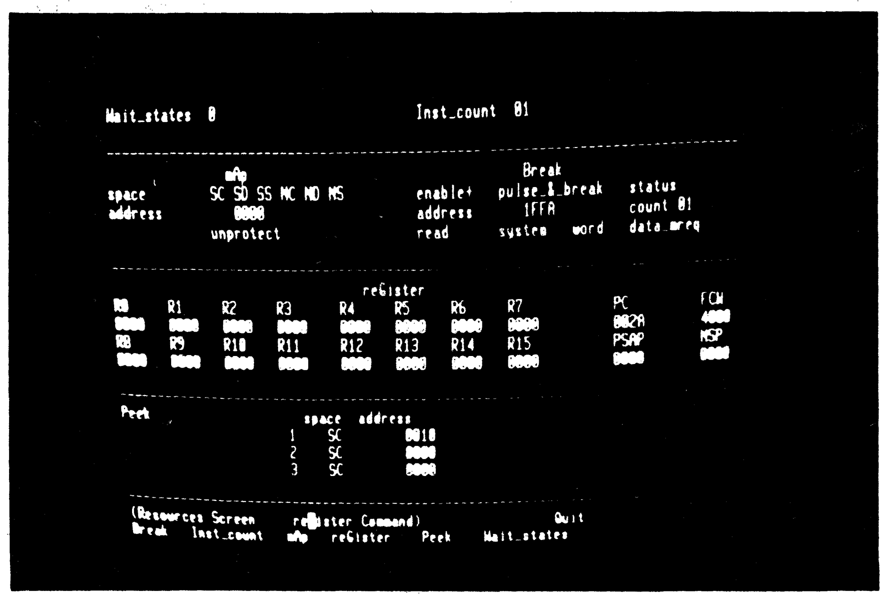


Figure 4-51. reGister Initialization

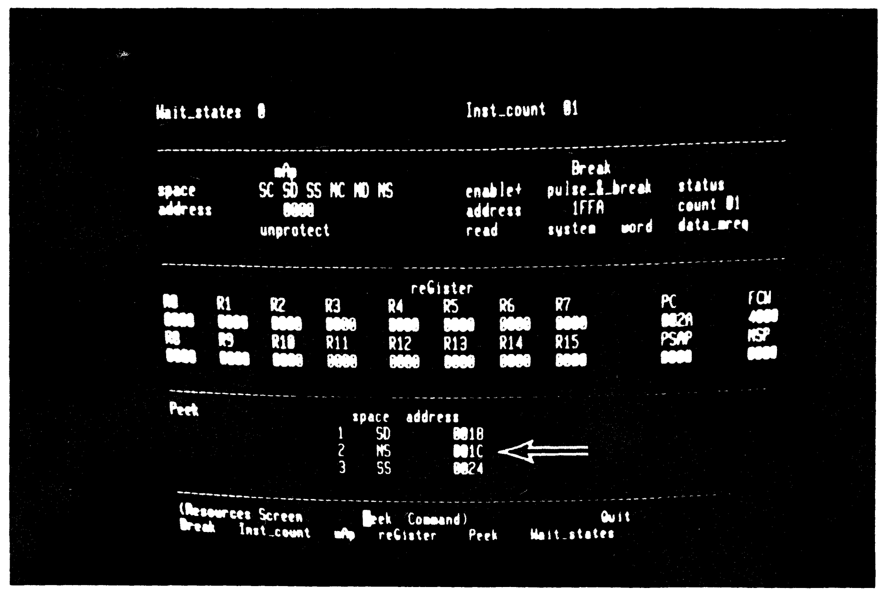


Figure 4-52. Set-up of Peek Parameters

Step	Key Sequence	Commentary
67.	B, 2, up, left, space, up, left, 0, 0, 2, A, RETURN	Finally, set up a breakpoint on the first instruction of the initialization routine of the example program.
68.	E, G	The Z-SCAN is now ready to run the program. There is a trigger break after the first instruction is executed. At this point, the only change is in the PC value.
69.	T, 4, down, 1, down	Trace the next five instructions, entering numbers to change the default Trace step count. At the end of the sequence, both system and normal stack pointers have been set up, changing the displays for the two **stack areas.** The final instruction, a System Call, pushes three words of data onto the system stack, producing a further change. The data also appears in the third Peek area.

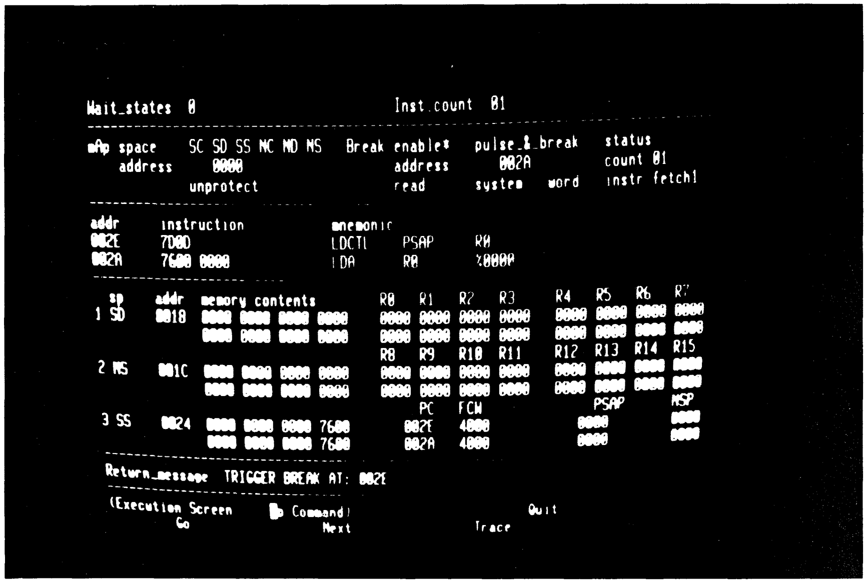


Figure 4-53. Emulation and Breakpoint

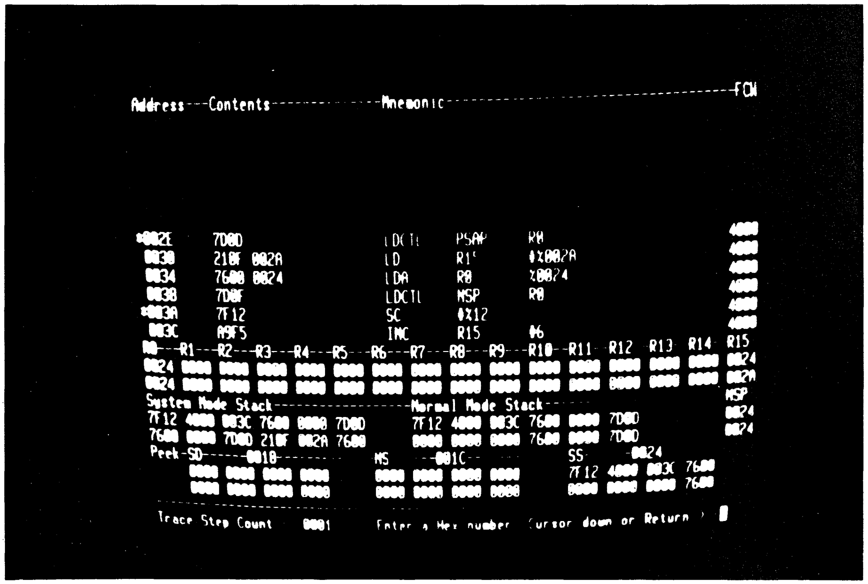


Figure 4-54. Tracing Initialization Routine

Step	Key Sequence	Commentary
70.	1, 0, 0, 0, down, BREAK	A large number of instructions can be traced by entering a hexidecimal number (bottom right of menu area). Tracing begins when you enter cursor down. Let the display run for awhile and observe that the program loops, alternately setting and clearing bit 14 of the FCW to move in and out of system mode. Tracing can be stopped at any time by entering the terminal BREAK key. The redisplayed memory content fields show that the contents of the data area and of both stacks have changed.
71.	RETURN, G, target RESET	Run the program. The emulation can be terminated with a target RESET because the status loaded from locations 0002 and 0004 in response to the input makes the CPU execute the instruction on which the breakpoint is set. The initial conditions of the program are not fully restored by the RESET because the data and stack areas may no longer be zero.

Figure 4-55. Trace of Main Routine

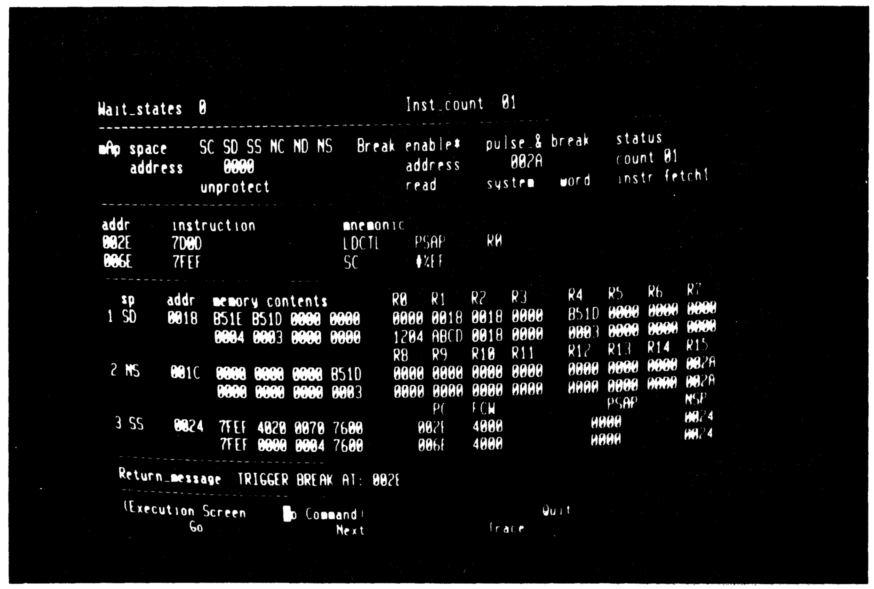


Figure 4-56. Trigger Due to Target Reset

Step	Key Sequence	Commentary
72.	RETURN, target NMI	A target NMI also terminates an emulation. Again, the initialization routine is entered in response to the input. The cause of the entry can be distinguished because the reset and NMI flag register values differ.

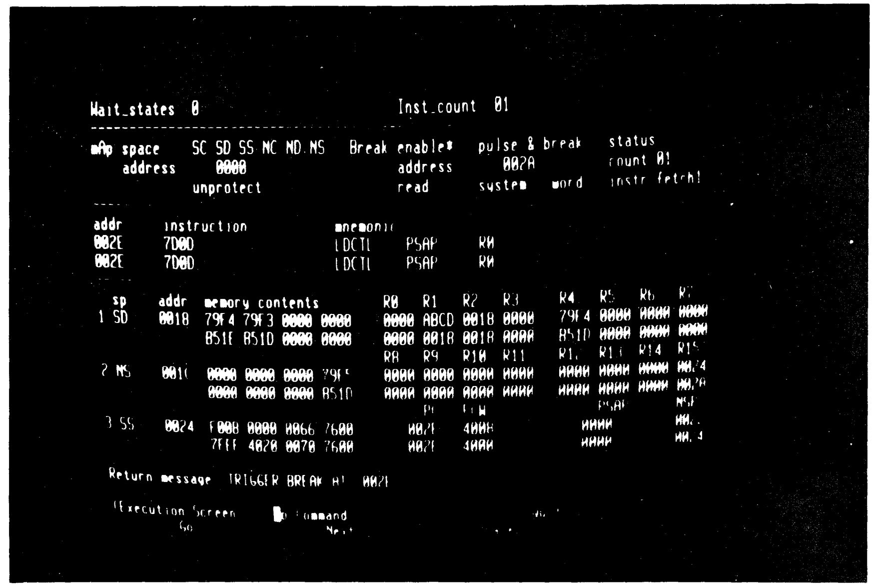


Figure 4-57. Trigger Due to Target NMI

Step	Key Sequence	Commentary
73.	R, B, down, 1, down, 1, left, A, RETURN	This tutorial does not explore the full possibilities of the program, which can generate a wide variety of bus cycle types in both system and normal modes. Experiment with it if you want to explore the Z-SCAN's features in more depth. As a start, set up a breakpoint on a system stack write of data pattern 002A.

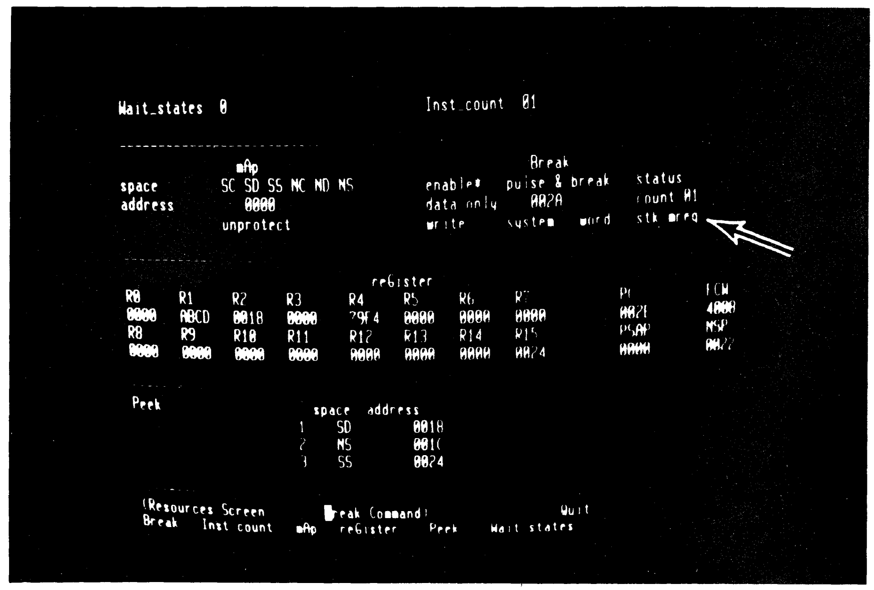


Figure 4-58. Set-up of Stack Write Break

Step	Key Sequence	Commentary
74.	E, G	This last emulation can run for as long as four seconds before the instruction at address 0064 writes the data pattern matching the programmed break condition. Z-SCAN may not stop the emulation before the next instruction is executed because the data match is detected only at the end of the last bus cycle of the INC instruction. Because of this, the next instuction, POP, is executed before the emulation terminates. This leaves the PC pointing to the LDR instruction.

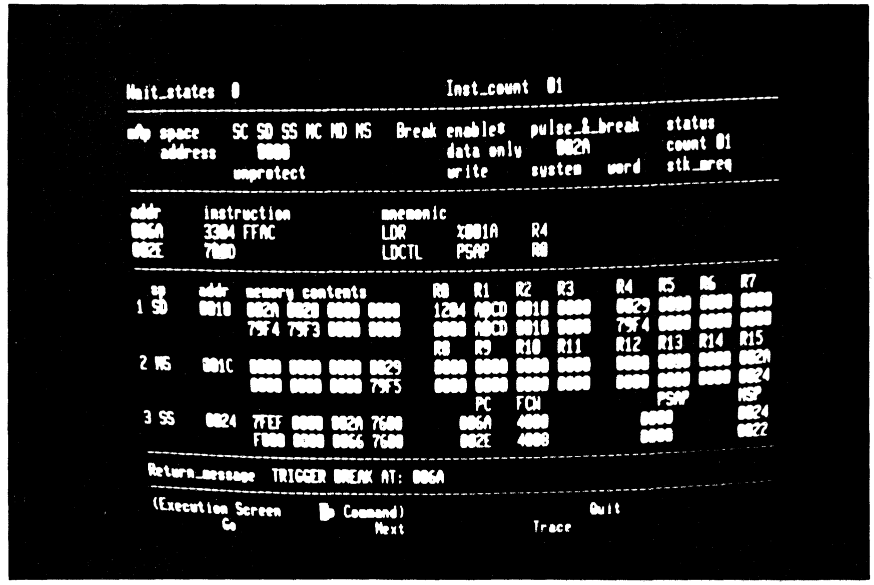


Figure 4-59. Break Following Stack Write

4.7 TUTORIAL SCRIPT FOR Z8001

The tutorial script for the Z8001 begins on the following page. If a Z8002 is currently operating in your Z-SCAN unit, turn back to the script in Section 4.5. Be sure not to type the commas or spaces shown throughout the key sequence.

Step	Key Sequence	Commentary
1.	**Monitor** RESET, RETURN	Z-SCAN is RESET. All information about the previous state of the hardware and software is lost. The monitor software uses the RETURN character to set up a baud rate generator, then it displays a **menu** of the CRT terminal types supported by the software. The cursor (a steady or flashing bright square on most terminals) appears in the center of the bottom screen line.
2.	Terminal selection digit	To configure the monitor for your terminal, enter one of the digits listed in the menu. If your terminal is not one of those listed on the menu, consult Appendix A and the documentation for the terminal. Pick a digit that corresponds to a protocol supported by the terminal.
3.	RETURN	The CRT screen is cleared, and the **System screen** is displayed. The cursor rests on the name of the screen, which is in parentheses on line 23, part of the **menu area.** This screen gives information about the status of the Z-SCAN hardware, for example, the CPU type currently operating in Z-SCAN and the software revision level. The displayed baud rates and revision levels may differ from those shown in the figure, but the CPU type must be the same. If it is not, follow the alternative tutorial of Section 4.5. If the display is corrupted, the digit entered in step 2 is incorrect and you must repeat the tutorial from step 1.

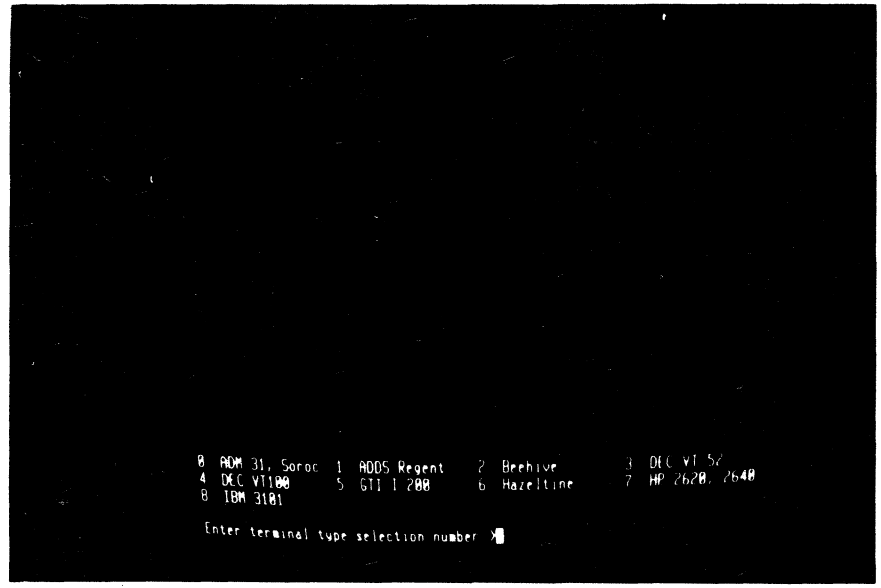


Figure 4-60. Z8001 Terminal Selection Screen

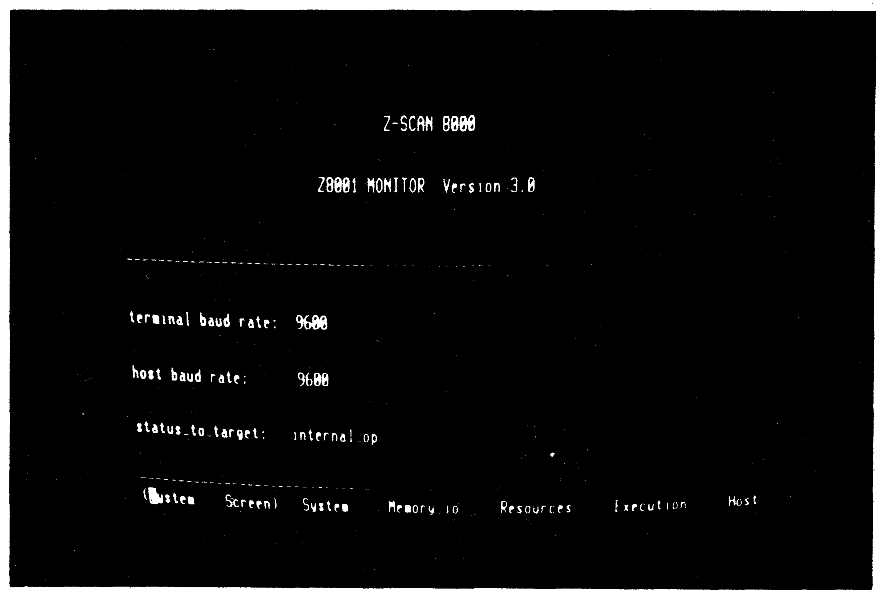


Figure 4-61. Z8001 Monitor System Screen

Step	Key Sequence	Commentary
4.	**M**	The single-keystroke **commands** you are allowed to enter appear as upper-case letters in the words outside the parentheses in the menu area. The command M calls up the **Memory_io** screen. Again, the cursor rests on the name of the screen, which appears on line 23 in the menu area.

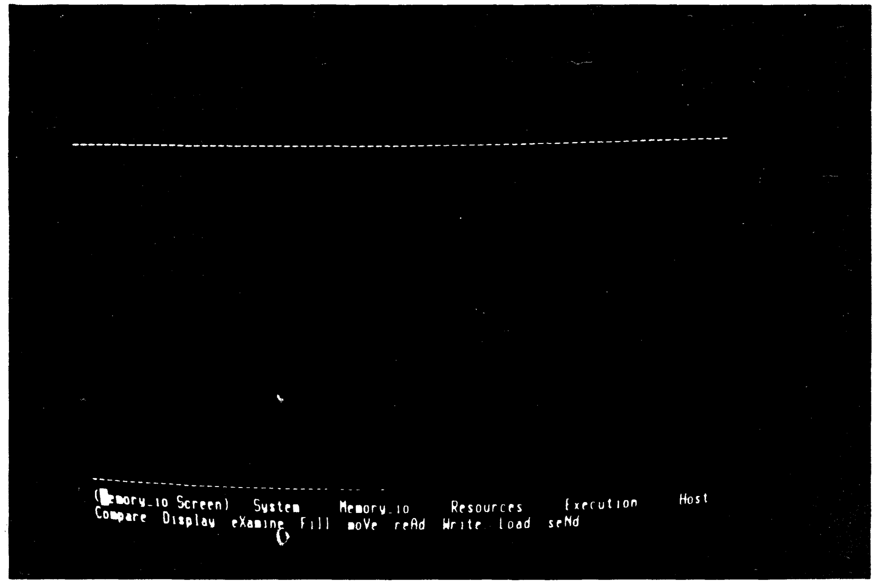


Figure 4-62. Z8001 Monitor Memory_io Screen

Step	Key Sequence	Commentary
5.	S, R	Among the valid commands shown in the menu area is S. Entering the command **reactivates** the system screen. A third screen, the Resources screen, can be called up by entering R. As usual, the cursor rests on the screen name, and legal commands are listed in the rest of the menu area.

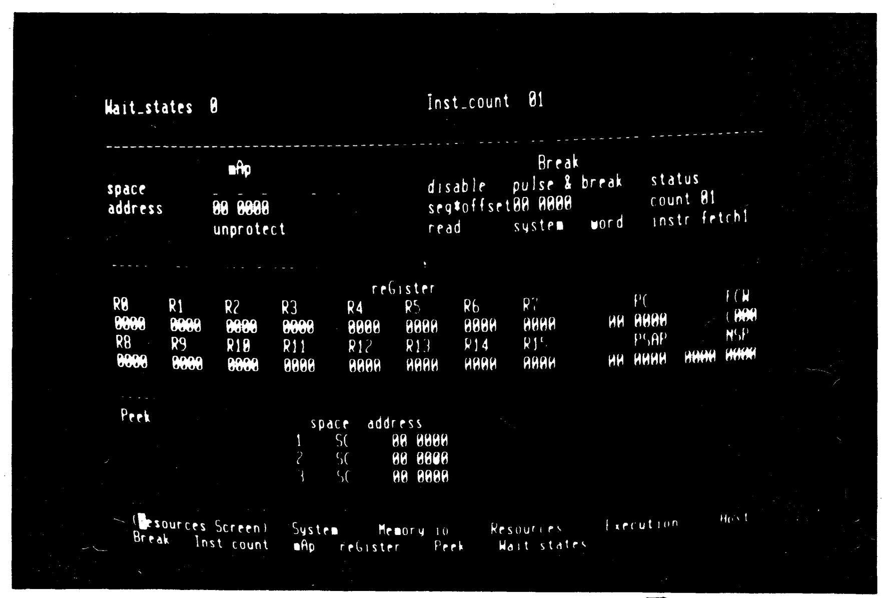


Figure 4-63. Z8001 Monitor Resources Screen

Step	Key Sequence	Commentary
6.	E	In step 5, you went from one screen to another by way of the System screen. However, it is usually possibly to move from one screen to another with a single keystroke. The Execution screen is **activated** by the command E.
7.	T	One display, the Trace screen, is accessible only from the Execution screen. Notice that there is no menu area because this screen does not support a variety of commands. It is dedicated to providing a detailed picture of program execution.

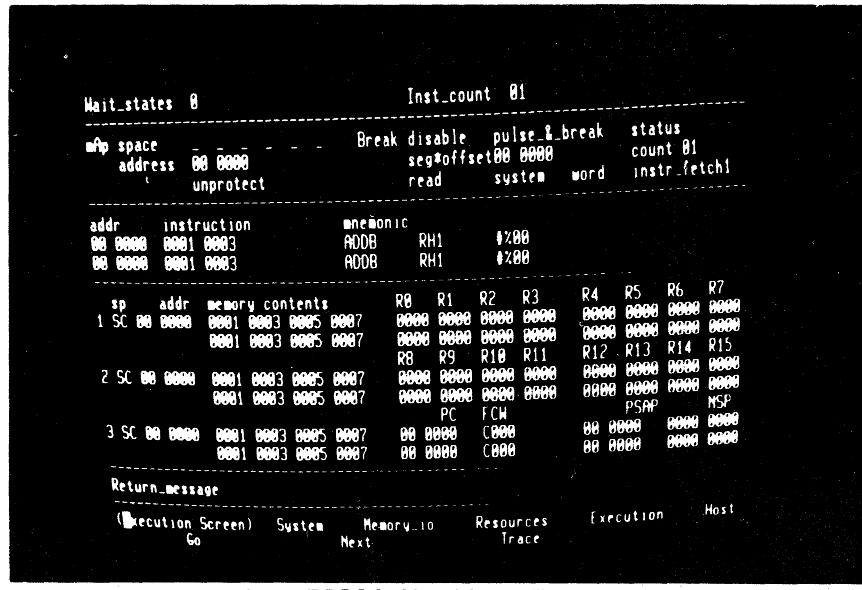


Figure 4-64. Z8001 Monitor Execution Screen

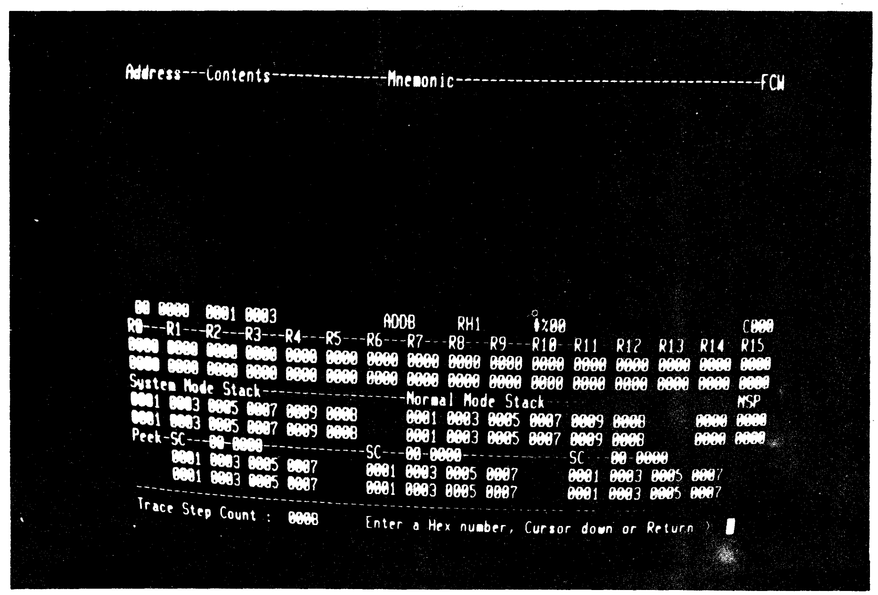


Figure 4-65. Z8001 Monitor Trace Screen

Step	Key Sequence	Commentary
8.	RETURN, H	Enter a RETURN to exit from the Trace screen to the Execution screen, then enter H. The Host command selects **Transparent** mode, allowing the terminal to communicate with a host system through Z-SCAN. You can enter the command even if no host is connected.

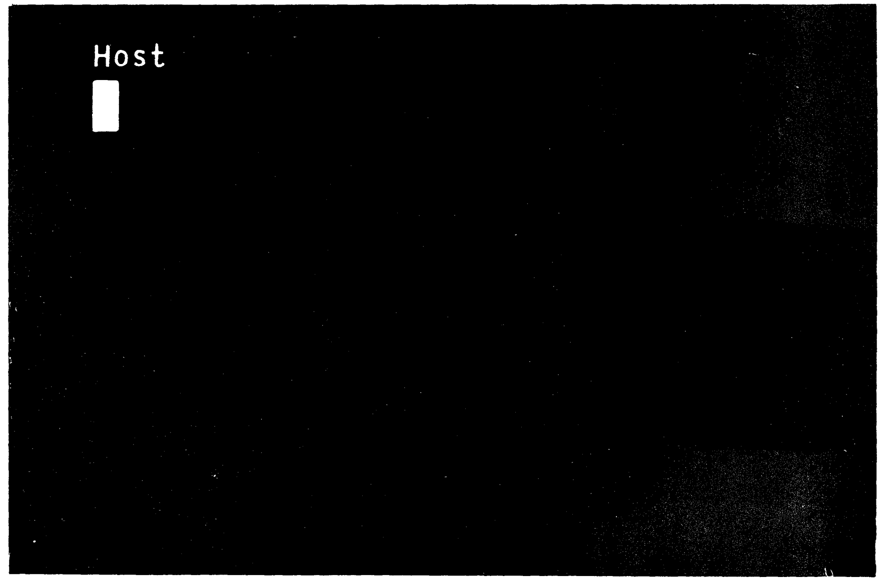


Figure 4-66. Host Screen, Transparent Mode

Step	Key Sequence	Commentary
9.	BREAK	Transparent mode is terminated when the BREAK key is entered. If the System screen does not reappear, consult your terminal documentation -you may have to press another key at the same time as BREAK, or the key may be disabled by an option setting inside the terminal. A monitor RESET can be used to end Transparent mode, but its use is not recommended because it destroys any information that was set up inside the Z-SCAN.
10.	R, A	So far the cursor has remained at the bottom of the screen except when the Host command was used. All of the **user-modifiable** fields on the Z-SCAN screens are outside the menu area. The fields are divided into groups, known as **subscreens.** Each subscreen is associated with a particular command and can be entered by keying the capital letter in the command name as it appears in the menu area. Note that as soon as you enter the A command, the first menu line changes to reflect the selected command (Resources screen, mAp command), and the cursor moves to the top left field in the mAp subscreen.

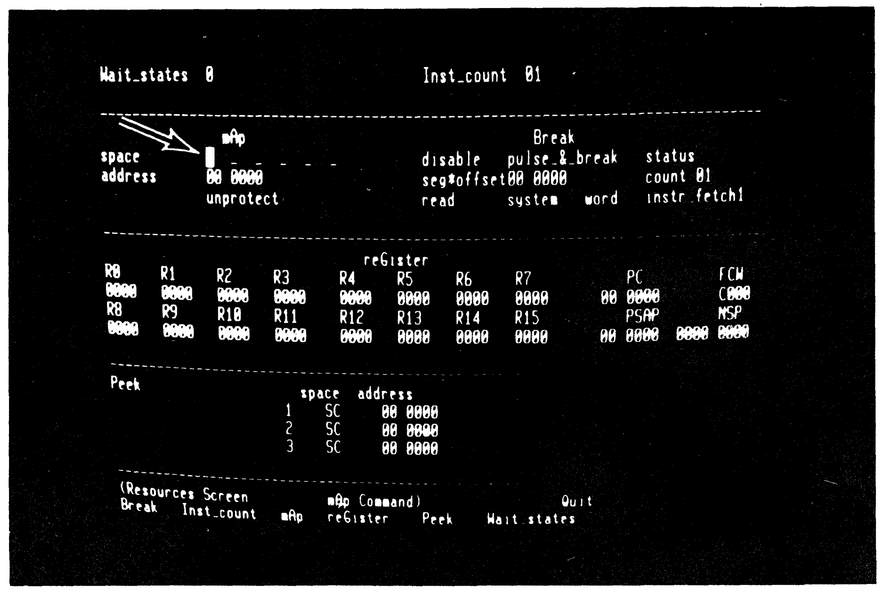


Figure 4-67. Cursor in mAp Subscreen

Step	Key Sequence	Commentary
11.	RETURN, B	To move the cursor back to the menu area, enter a RETURN. The menu display does not change because the mAp command is still active. It is altered when a new command, Break, is activated. The cursor moves to the top left field in the Break subscreen.

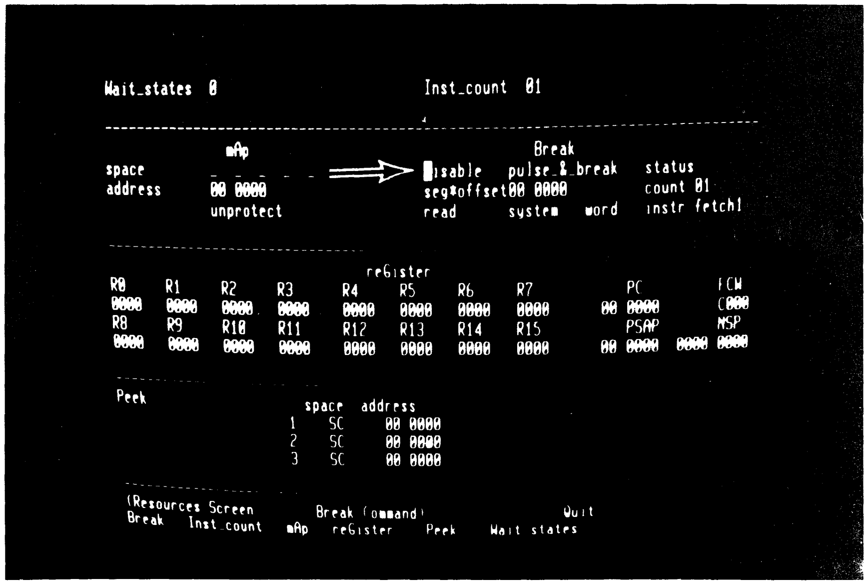


Figure 4-68. Cursor in Break Subscreen

Step	Key Sequence	Commentary
12.	RETURN, **Q**, S, R, A	You should now be comfortable with activating screens and commands. The only new command in this sequence is Quit. It deactivates the current command and modifies the menu to show the names of the other screens.
13.	RETURN, S, R, **A**	It is not necessary to use the Quit command before moving to another screen. You can enter the initial letter of the new screen name even if it is not currently listed in the menu area.
14.	right, right, right left, left, left, left	Most subscreens consist of more than one field. Once the cursor is in a subscreen, it can be moved to the other fields in the same subscreen by using the cursor control keys. If the cursor left key is entered while the cursor is in the leftmost field, the cursor **wraps around** to the rightmost field in a subscreen line.
15.	right, down, down, down, up, left, right	The same wrap-around applies in the vertical direction. Note that when there is only one field on a particular line of a subscreen, the horizontal cursor movement keys cannot move the cursor out of that field. The cursor keys can never move the cursor out of the active subscreen.

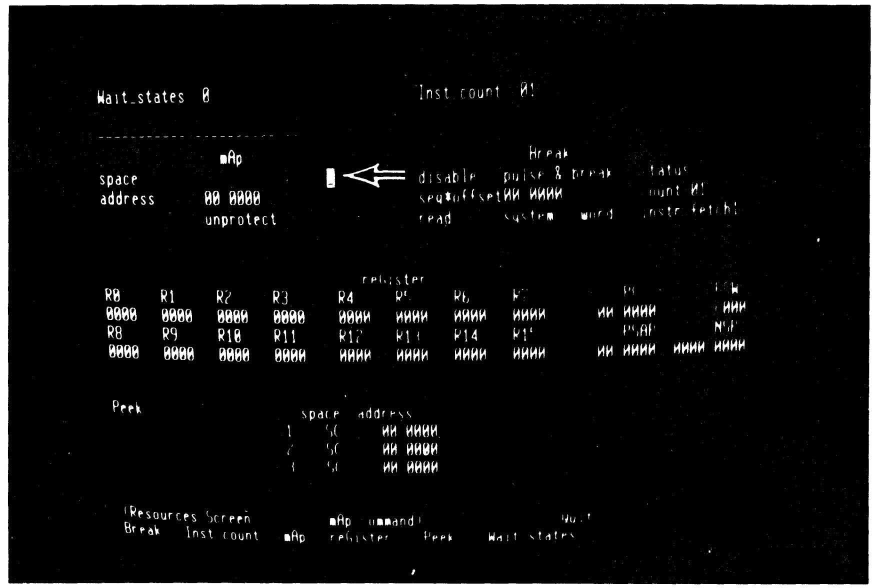


Figure 4-69. Horizontal Cursor Movement

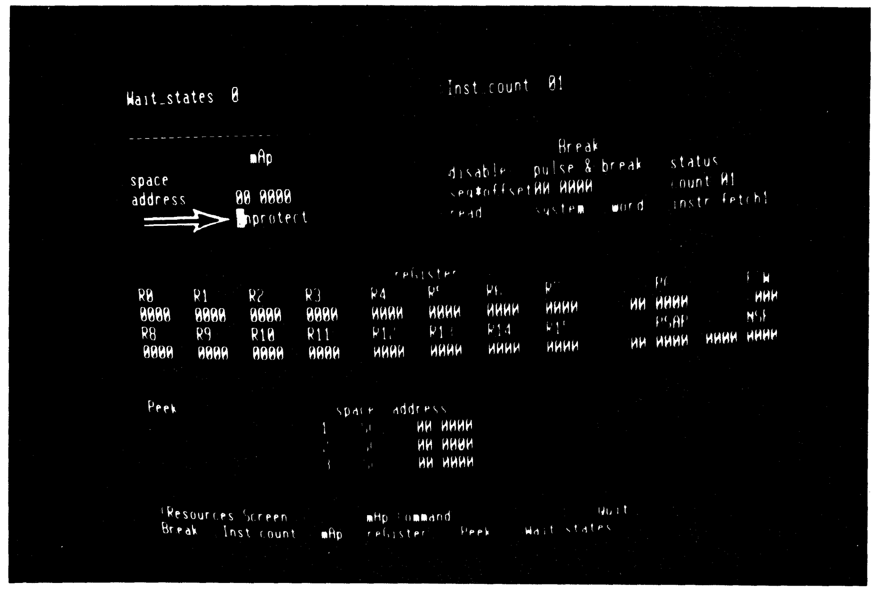


Figure 4-70. Vertical Cursor Movement

Step	Key Sequence	Commentary
16.	RETURN, RETURN	The RETURN key moves the cursor back to the menu area. Because the command remains active, a second RETURN moves the cursor to the top left field in its subscreen: there is no need to re-enter the command name.
17.	>, >, <, 0, 1, space G, F, H, CTRL R, >	Each of the six fields on the first line of the mAp subscreen corresponds to one of the Z8001's address spaces, and each has just two possible values. In the **default** state, an underbar is displayed, indicating that the 8K bytes of **mappable memory** will not respond to CPU accesses made to a particular address space during an emulation. In the alternative state, a two-letter abbreviation for the name of the address space (for example, SC for System Code) shows that the mappable memory will respond. You can step forward or backward through the possible values with the > and < keys or you can access them directly by entering 0 for the first choice and 1 for the second. Alternatively, space and F select the default and final values. CTRL R restores the field to the value it held when the cursor entered it. Other printable characters that are not hexadecimal digits do not affect the field.

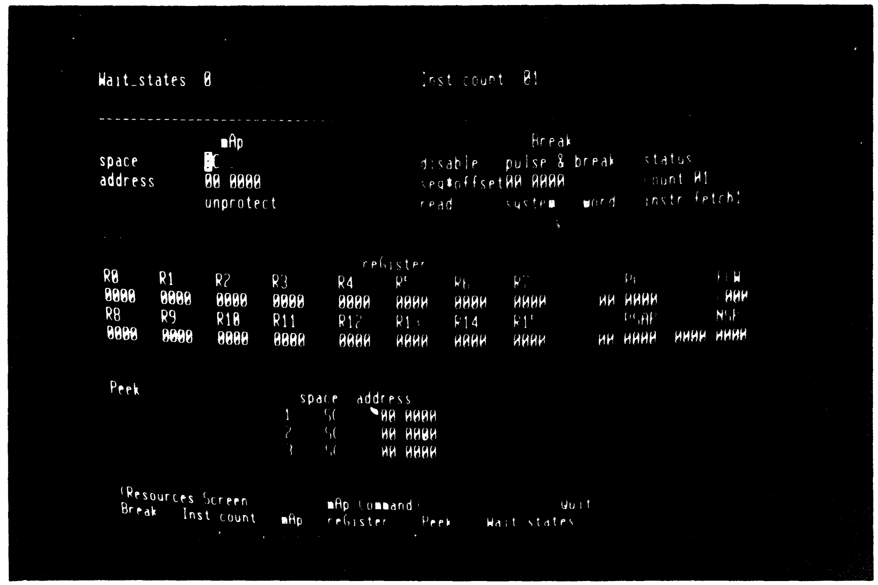


Figure 4-71. Enabling Mappable Memory

Step	Key Sequence	Commentary
18.	RETURN, B, 2	The emulation you are going to run requires a **breakpoint,** so you must enable the breakpoint logic by setting the first field of the Break subscreen to "enable*". This tells the logic to search for a simultaneous match in the segment field, the offset field and the various status fields.
19.	down, right, right, >, >, >, <, 1, RETURN	The breakpoint segment number is correct but the offset must be changed. The address field contains four hexadecimal digits and can hold any value between 0000 and FFFF. Use > and < to move the cursor within the field, and enter new hex digits to change the value. You have now set a breakpoint which will be triggered when the first word of an instruction is read from system code location 0010 in segment 00.

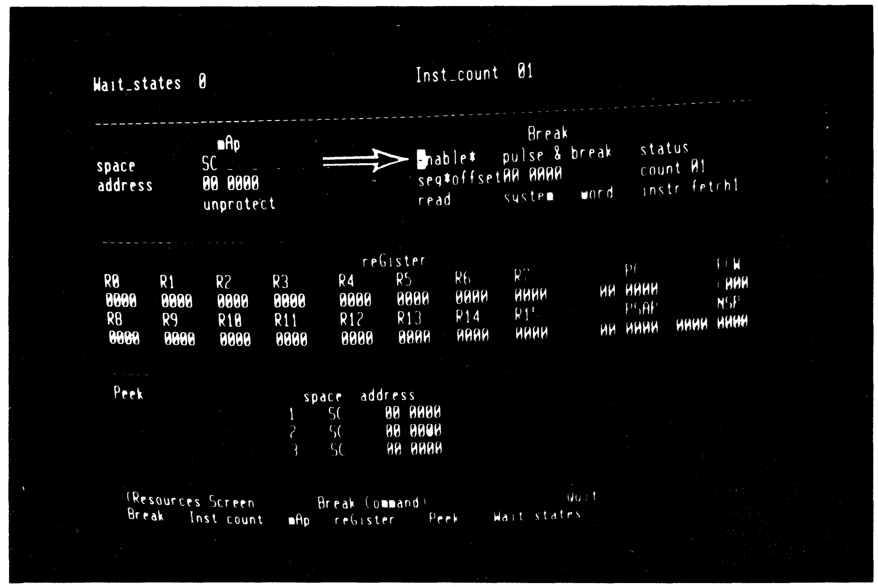


Figure 4-72. Enabling Break Logic

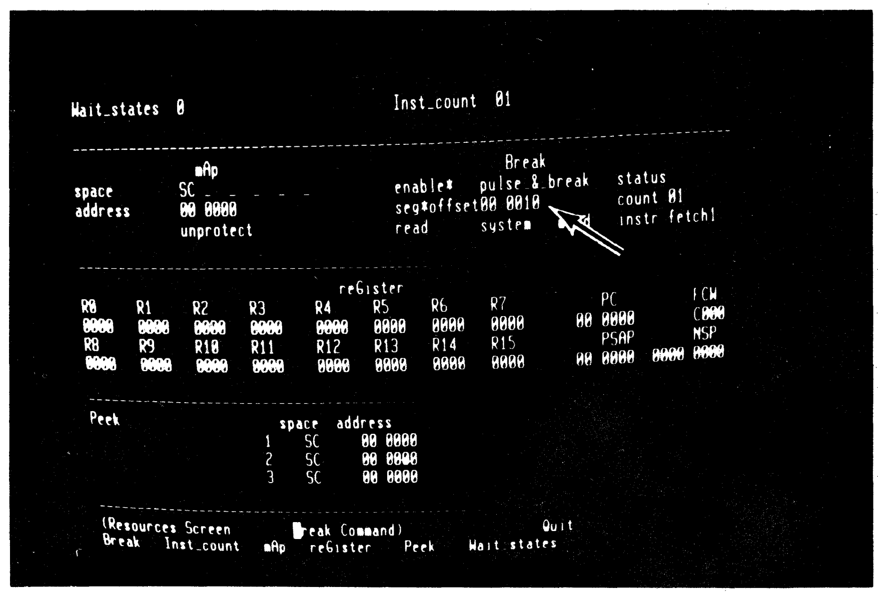


Figure 4-73. Setting Break Address

Step	Key Sequence	Commentary
20.	M	Move to the Memory-io screen. When it is displayed, notice that the top three lines are blank.
21.	F	Fill is listed as a valid command in the menu area. As soon as the command is activated, the cursor moves to the first field of the Fill subscreen which appears at the top of the screen.
22.	left, 1, F, F, F, down A, 8, 0, B	Use the Fill command to fill mappable memory, which currently extends from address 0000 to 1FFF in the first segment of system code space, with increment byte register instructions (opcode A80B, mnemonic INCB RH0, #12). In order to do this, you must change the contents of some of the fields on the subscreen. The Fill string can be up to 16 hex digits long, but only four are required in this case.
23.	RETURN	After the parameters have been set up, the command must be executed by entering a RETURN. Before execution starts, the cursor moves to the bottom of the central window area. The message "DONE" is displayed when execution is complete.

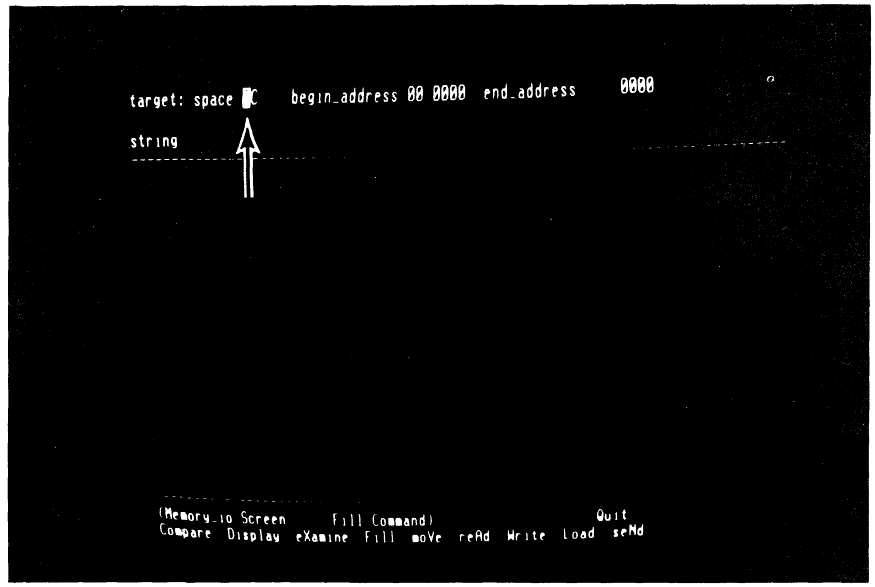


Figure 4-74. Default Fill Command Display

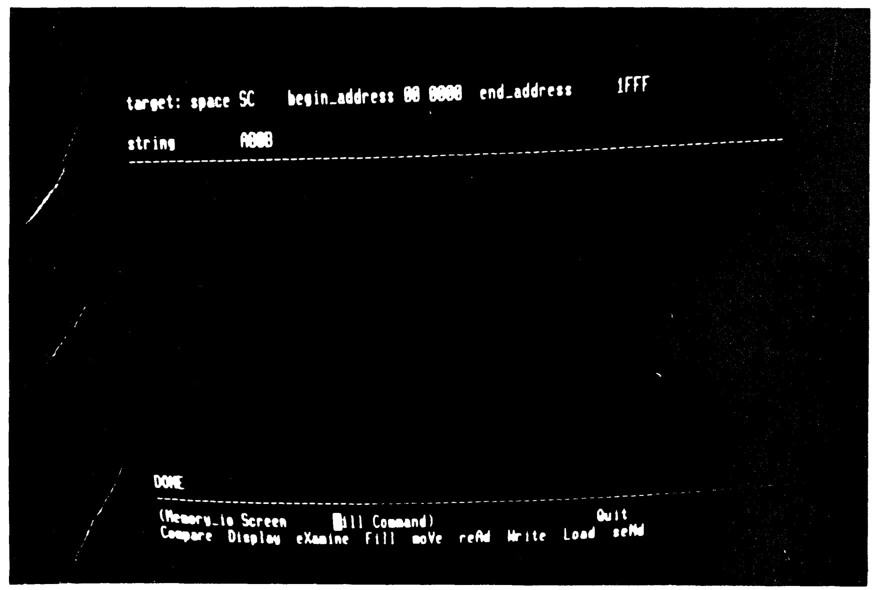


Figure 4-75. Execution of Fill Command

Step	Key Sequence	Commentary
24.	D, RETURN	The Z-SCAN Display command is used to look at the contents of memory. In order to look at the bottom of system code memory, you do not need to change the default parameters that appear at the top of the screen when the command is activated, so execute the command immediately. Addresses appear at the left of the screen, data in the center and at the right is an ASCII representation of the same data. Neither A8 nor 0B corresponds to a printable character. Periods are used to show this. The asterisks are delimiters.
25.	down, up, RETURN	After the Display command has filled the window area, the cursor rests at the bottom right of the screen. You can enter cursor down to display the next block of memory or cursor up to display the previous block. The command is terminated when RETURN is entered.

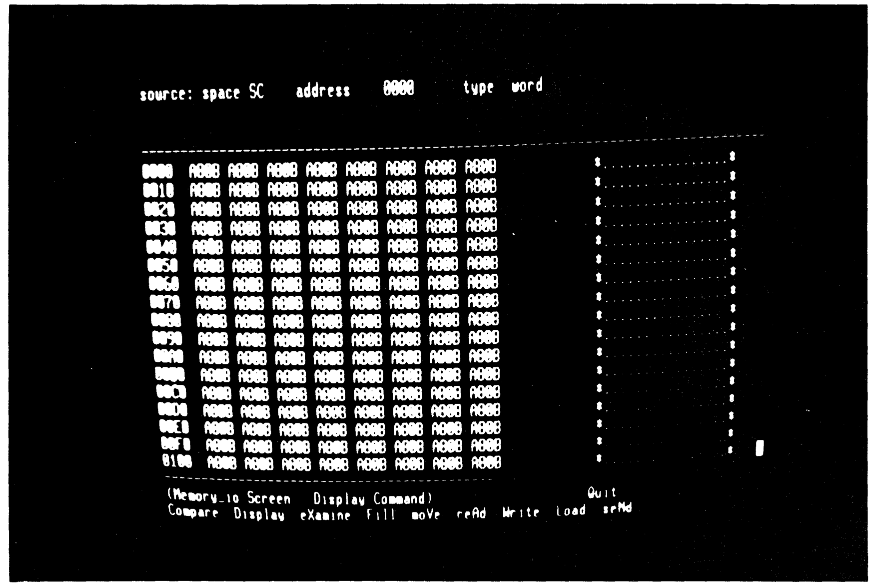


Figure 4-76. Display with Default Parameters

Step	Key Sequence	Commentary
26.	RETURN, left, 4 RETURN, RETURN	The command remains active as long as its name appears inside the parentheses on the menu line, so a second return moves the cursor back into the parameter subscreen. Set the type field so that memory is displayed as **disassembled** segmented Z8001 instructions.
27.	X, right, right, 1, F, F, C, RETURN	The eXamine command allows you to look at and, if desired, modify the contents of memory. Like Fill and Display, it has a private subscreen. The first location you need to examine is the word at system code location 1FFC in segment 00. Its current contents are displayed when the command is executed, and you are prompted for a new value to replace them.

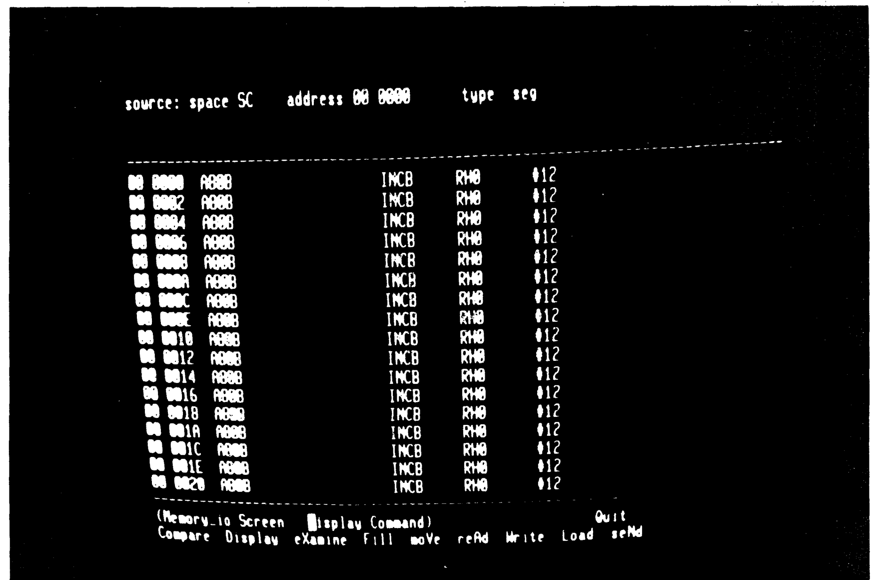


Figure 4-77. Disassembled Memory Display

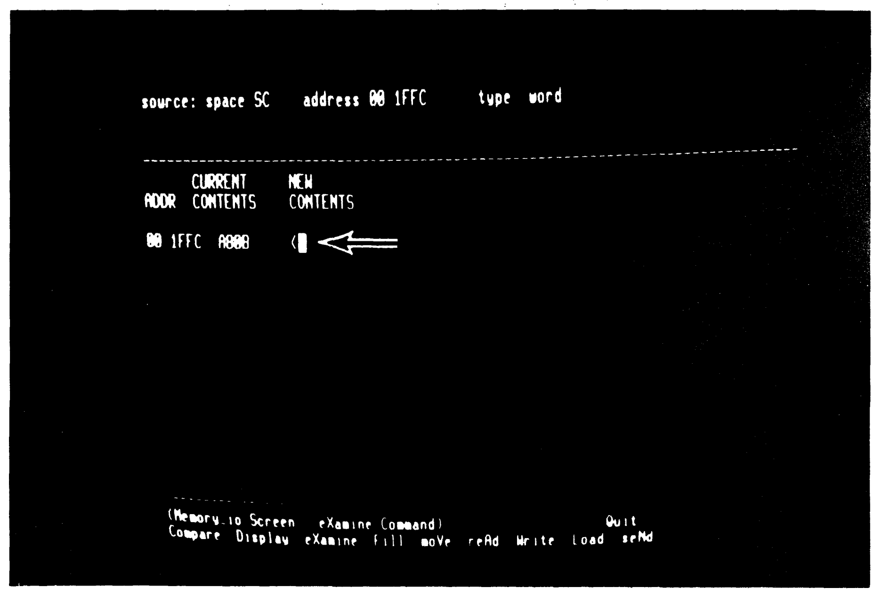


Figure 4-78. Set-up of eXamine Command

Step	Key Sequence	Commentary
28.	5, E, F, <, 0, 8, 1, 8 down	This step replaces the two INCB instructions at the top of mappable memory with an unconditional jump to location segment 00 0018 (opcode 5E08 0018, mnemonic JP <<00>> %0018). Short offset addressing is used to save bytes. The < key can be used to backspace over incorrect input. When sufficient digits have been entered to fill the **open** location, the new value is stored and the next location is opened automatically. The cursor down key opens the next location immediately, storing any digits which have been entered. The data seen in location 2000 may vary because no memory responds at that address.
29.	up, RETURN	Cursor up reopens the previous location, showing that the two digits entered in the previous step have been stored right justified in a field of zeros.

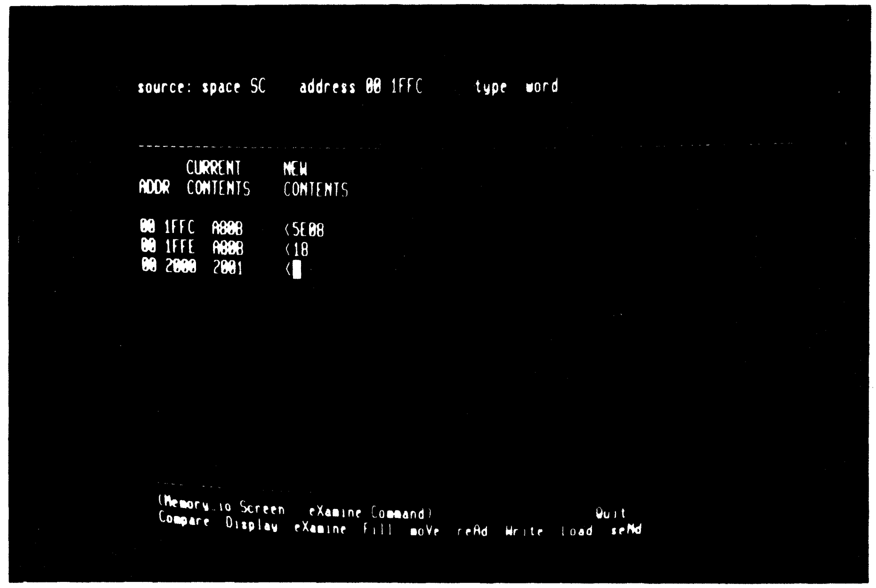


Figure 4-79. Modification of Memory Contents

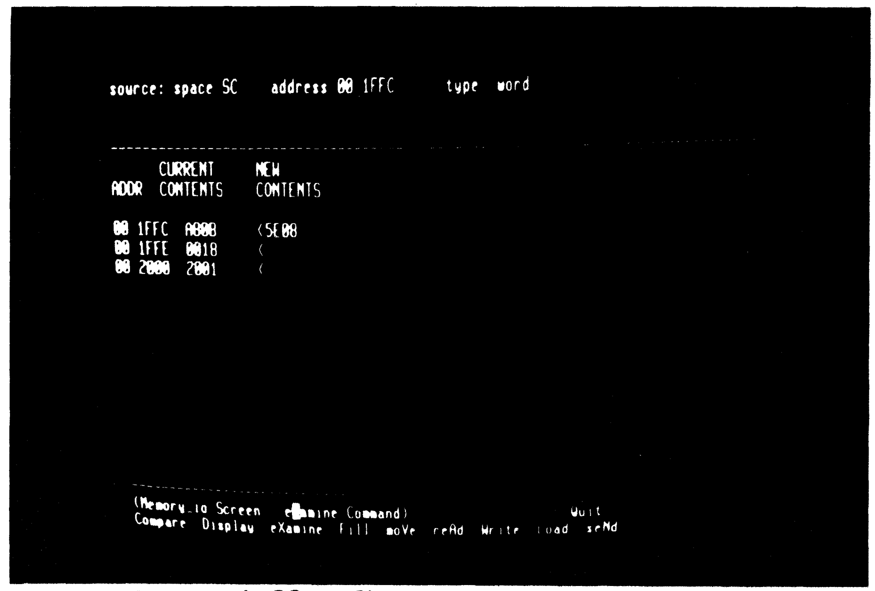


Figure 4-80. Checking Memory Contents

Step	Key Sequence	Commentary
30.	C, left, 1, 0, 0, 0 left, 1, RETURN	Just to check that everything is set up correctly, the Compare command is used to find the differences between the contents of the top and bottom 4K byte blocks of mappable memory. The byte count field for this commands is, like all Z-SCAN monitor numeric fields, hexadecimal (1000 hex = 4096 decimal = 4K). When the command is executed, it should reveal that just four bytes differ between the top and bottom halves of the memory. If it shows anything else, you have probably made a mistake somewhere and not corrected it. To recover type R, A or, if the cursor is at the bottom right of the screen, RETURN, R, A and repeat the tutorial from step 17.

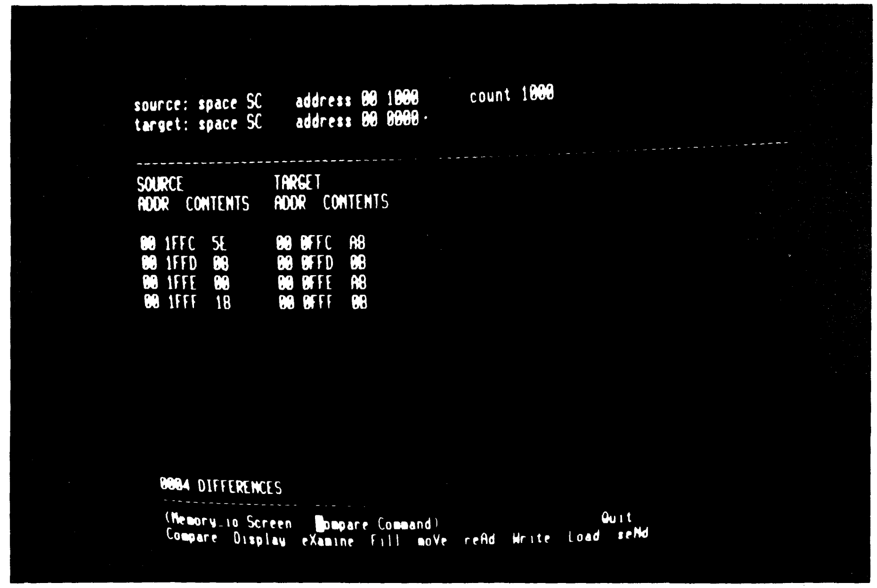


Figure 4-81. Use of the Compare Command

Step	Key Sequence	Commentary
31. E		The program in the mappable memory consists of 4,094 (decimal) INCB instructions and an unconditional jump. It can be run from the Execution screen. The default values of the Program Counter (PC) and Flag and Control Word (FCW) are suitable for running this first emulation. The emulation will run in system mode because bit 14 of the FCW is set. Bit 15 selects segmented mode and is also set in the default FCW value provided by the monitor.
32. N		The Next command **steps** through the number of instructions displayed in the Instruction_count field at the top right of the screen. In this case, the count is one. After the single instruction has been executed, the whole screen is redisplayed, updating the emulation status. Two registers are affected: RH0, the high byte of R0 has been incremented by 12 (decimal) and the PC offset has moved to next instruction. The top row of the instruction and register values reflect the state of the program after the emulation. The bottom row values reflect the state of the program at the end of the previous emulation.

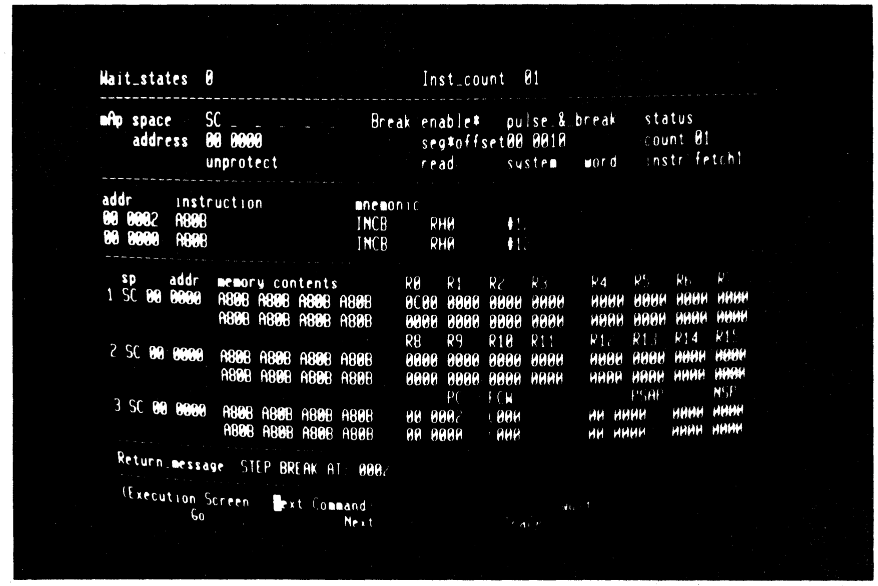


Figure 4-82. Instruction Step with Next Command

Step	Key Sequence	Commentary
33. RETURN		Now that the Next command is active, it may be repeated by entering RETURN. Again, the PC value changes and RHO is incremented.
34. G		The Go command starts an emulation which does not stop until a break condition is encountered. Your program should **trigger** the breakpoint logic when an instruction is fetched from location 0010. The breakpoint is honored after the instruction has been executed so that the emulation ends with the Program Counter pointing to the instruction at location 0012. Note that the termination message is different from that of the Next command.

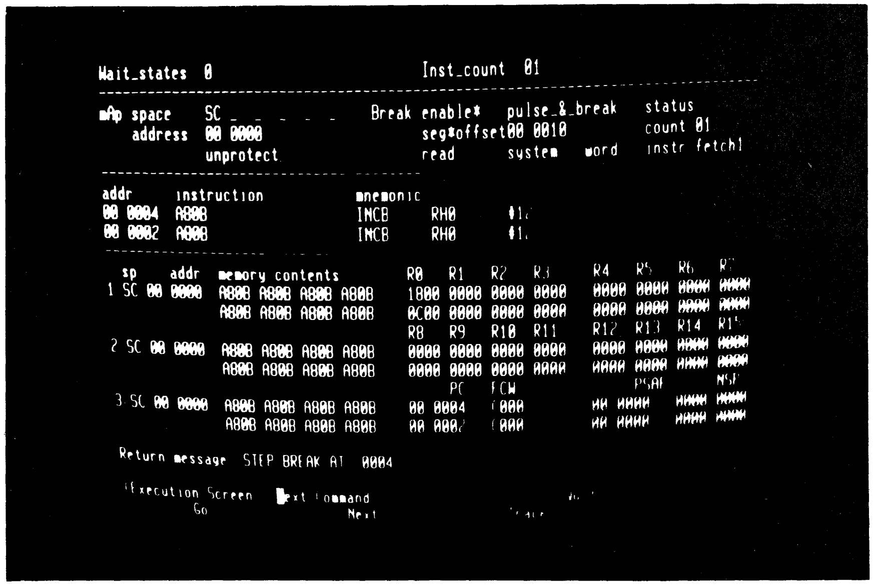


Figure 4-83. Second Instruction Step

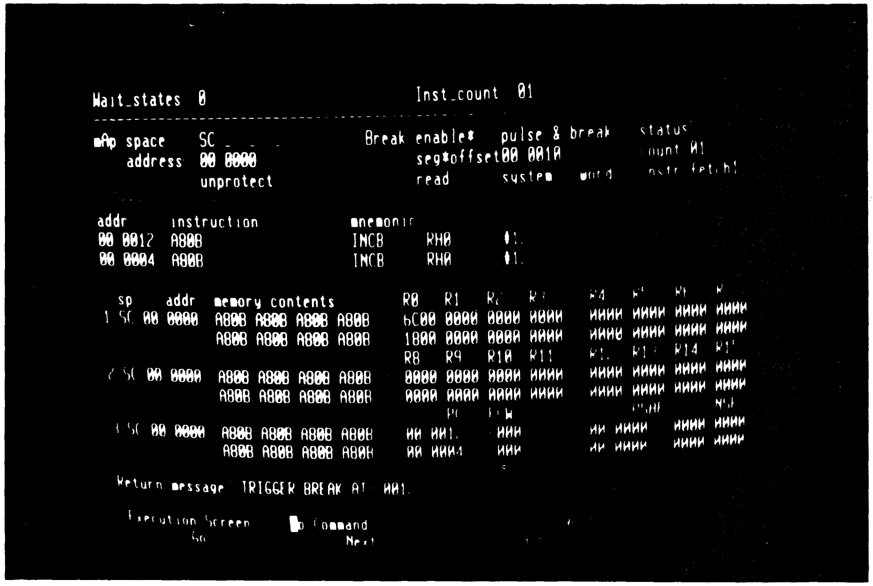


Figure 4-84. Running to Breakpoint with Go Command

Step	Key Sequence	Commentary
35. T		Emulations can also be run from the Trace screen, which disassembles each instruction before it is executed. The instruction which appears in the center of the screen is the first to be executed when emulation starts. The bottom of the screen displays register and memory contents. The function of these fields will be explored later.
36. down		Entering cursor down results in the execution of the number of instructions given in the count field at the bottom left of the screen. PC and FCW values are given for each instruction executed, and the first instruction executed is flagged with an asterisk in column 1. The remaining registers are not redisplayed until all the instructions have been executed. The FCW values at the right of the screen show that the value in RHO has overflowed and become negative.

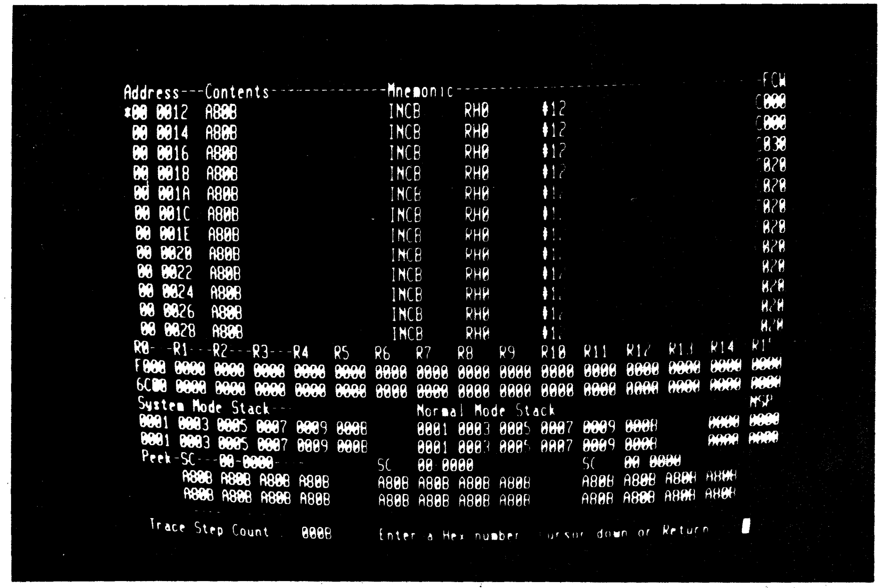


Figure 4-85. Use of the Trace Screen

Step	Key Sequence	Commentary
37.	RETURN, G	Return from the Trace screen to the Execution screen and start another emulation with the Go command. This time the breakpoint is not encountered - the program loops in the address range 0018 to 1FFC, avoiding location 0010. While the emulation is running, the cursor rests in the blanked return message line and the terminal keyboard is disabled.
38.	monitor NMI	The monitor NMI signal acts as a **manual break** request during emulations run from the Execution screen. The emulation terminates when execution of the current instruction is complete. The break address and register contents which you see will probably be different from those in the photograph, but this does not matter.

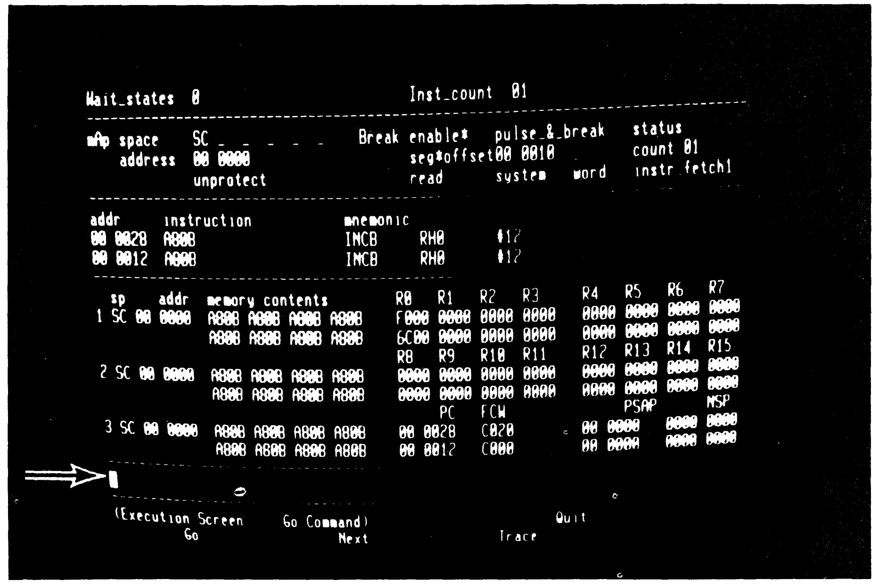


Figure 4-86. Indefinite Emulation with Go Command

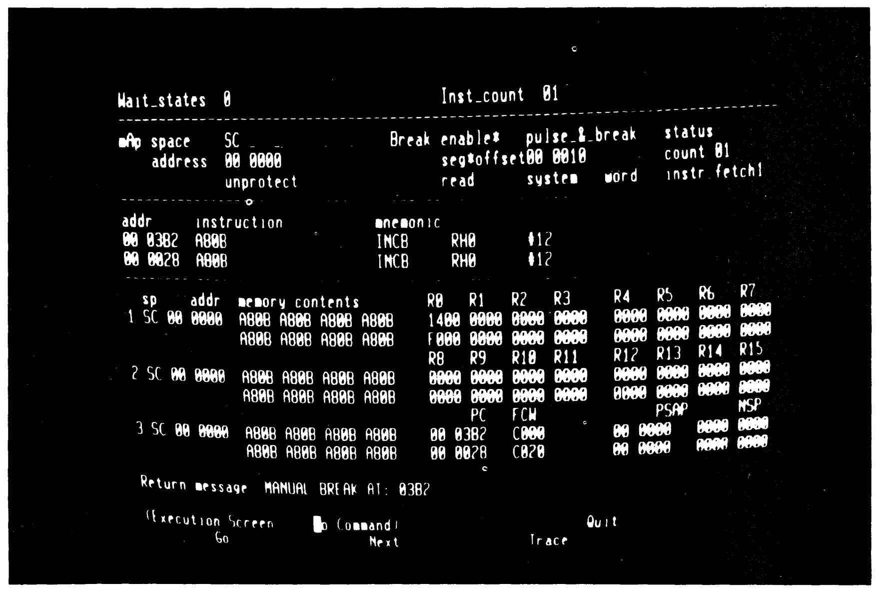


Figure 4-87. Manual Break with NMI Switch

Step	Key Sequence	Commentary
39.	**M**, X, right, right, space, CTRL R, 0, 0, 1, 8	To explore further facilities offered by Z-SCAN, an instruction which reads and writes memory is required. Use the Memory_io screen eXamine command to insert an instruction at location 0018. Two of the keystrokes in this sequence are redundant. The space restores the address field to its default value and CTRL R cancels any changes made since the cursor entered the field.
40.	RETURN, 6, 9, 0, F, 1 0, RETURN	The instruction is INC <<00>> %0010 , #16 in segment 00 (increment by 16 the word at location 0010). If short offset addressing is used, it has a two-word opcode, 690F 0010.
41.	**C**, BREAK, RETURN, left 0, RETURN	Check memory contents again by using the Compare command. Extra keystrokes in this sequence show that the BREAK key moves the cursor back to the menu area without executing the active command and that the monitor does not allow you to enter an illegal value in a numeric field: the previous value of the field is restored. When the command is executed it should show eight differences.

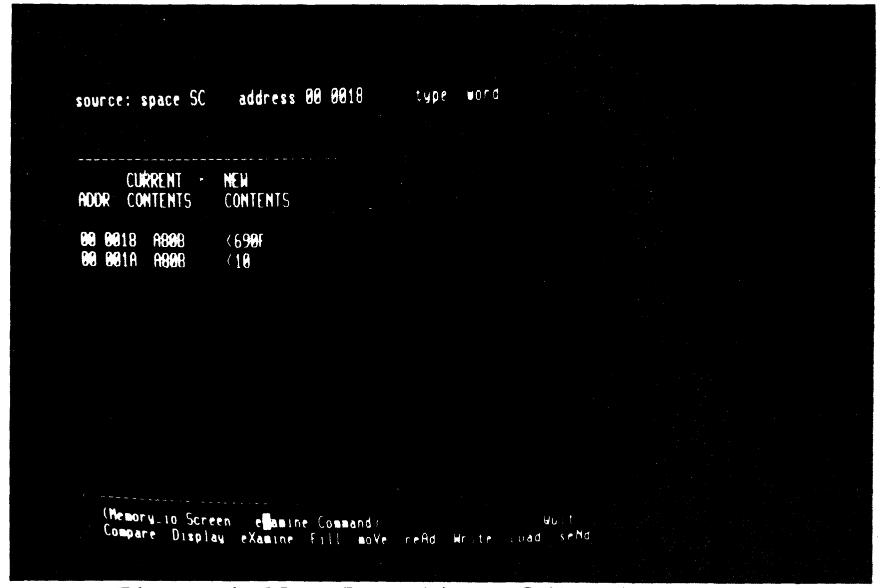


Figure 4-88. Insertion of New Instruction

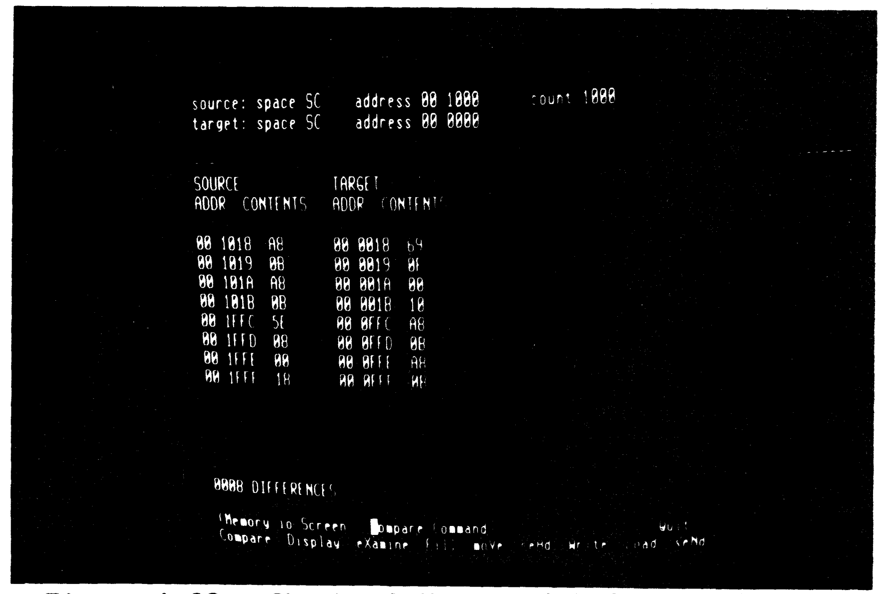


Figure 4-89. Check of Change with Compare Command

Step	Key Sequence	Commentary
42.	D, RETURN, RETURN	Display disassembled memory to show the new instruction at location 0018.

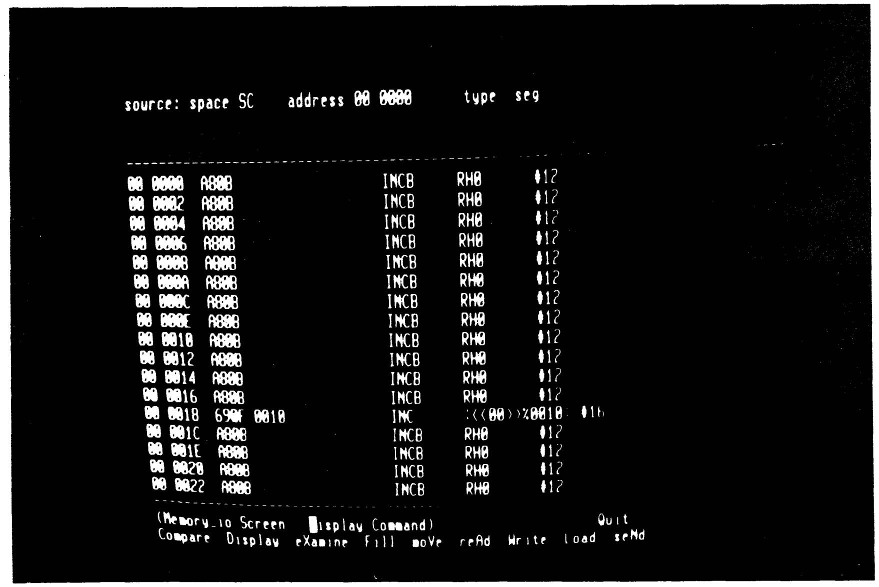


Figure 4-90. Display of Change

Step	Key Sequence	Commentary
43.	R, P, left, >, >, 1, RETURN	The added instruction modifies the contents of location 0010 each time it is executed, so it is desirable to know how they have changed after each emulation is run. The Z-SCAN displays the contents of selected locations on the Execution and Trace screens. The monitored addresses are set up by the Peek command on the Resources screen. Modify the first of the three addresses to 0010.

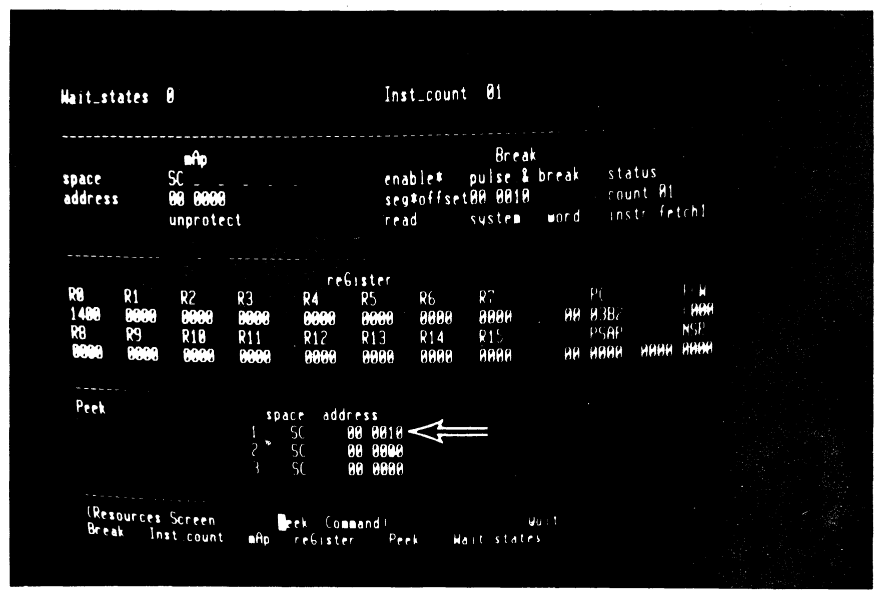


Figure 4-91. Setting Peek Parameters

Step	Key Sequence	Commentary
44.	E, G	Now call up the system screen and start an emulation. The top line of the first Peek field shows the contents of word locations 0010 through 0016 as they were before the emulation started.
45.	monitor NMI	You might think that this emulation should stop with a trigger break, because location 0010 is being read by the new instruction. The trigger logic does not fire because the break parameters are set up for an instruction fetch, not a data read, so the emulation must be terminated with a manual break. Looking at the Peek memory areas, you see that the contents of location 0010 have not changed during the emulation. Remember that the mappable memory has been set to respond only to system code space accesses. This explains why the system data accesses made by the new instruction do not affect it.

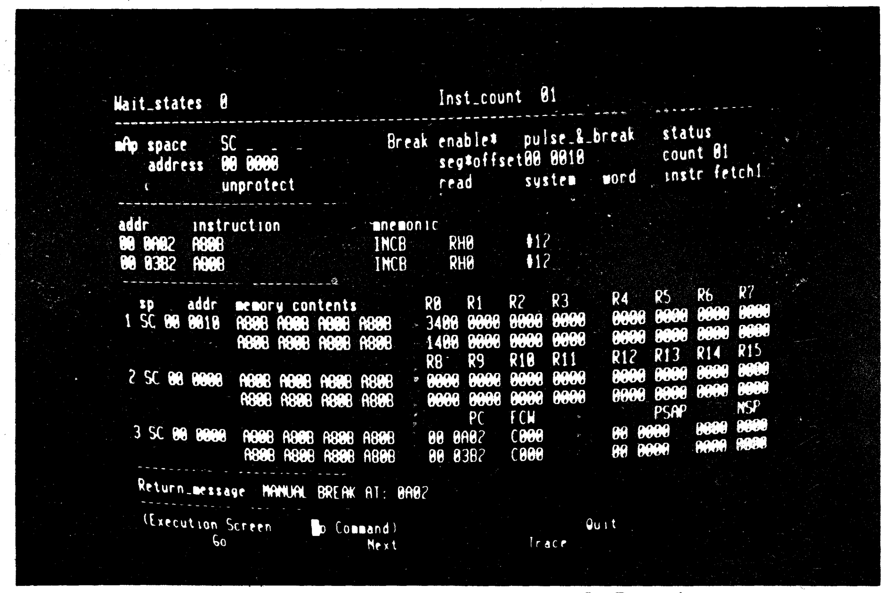


Figure 4-92. Second Manual Break

Step	Key Sequence	Commentary
46.	R, B, up, left, >, >, F <, space, 1, 2, 9, RETURN	To fix these two problems, leave the Execution screen, which, though it displays data about mappable memory and the break condition, does not allow you to modify the parameters. Use the Resources screen Break command to set up a breakpoint on a data memory request. This is one of 16 possible values in the bus cycle type field. As usual you can select a choice eiher by stepping through the table of possible values or by entering a number that corresponds to the required choice. The space character selects the default value.
47.	A, right, 1, RETURN	The second field in the mAp sub-screen determines whether or not the mappable memory responds to system data accesses. Entering a 1 sets the field to "SD". The mappable memory now responds to two types of accesses. For this reason, it is not necessary to modify the memory space parameters of Peek. System code location 0010 is the same memory word as system data location 0010.

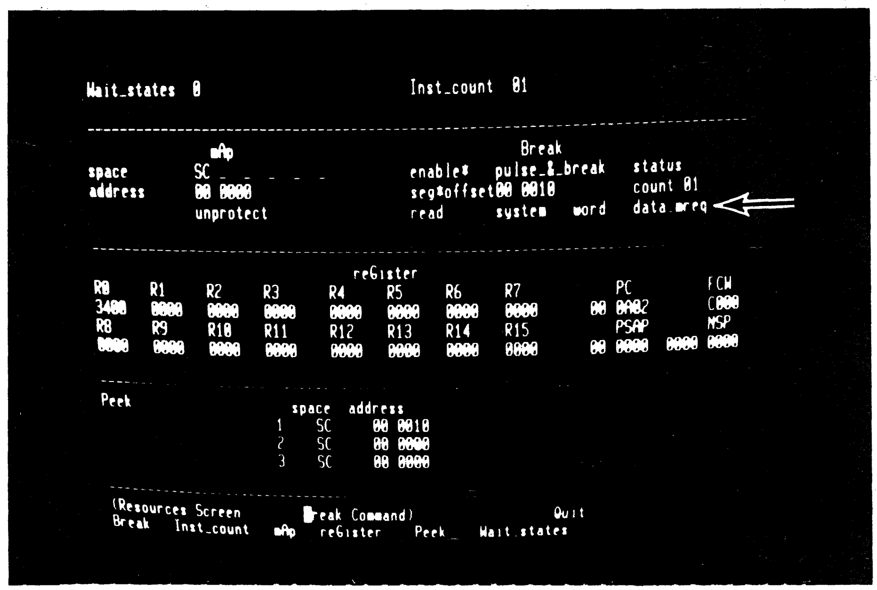


Figure 4-93. Modification of Break Parameters

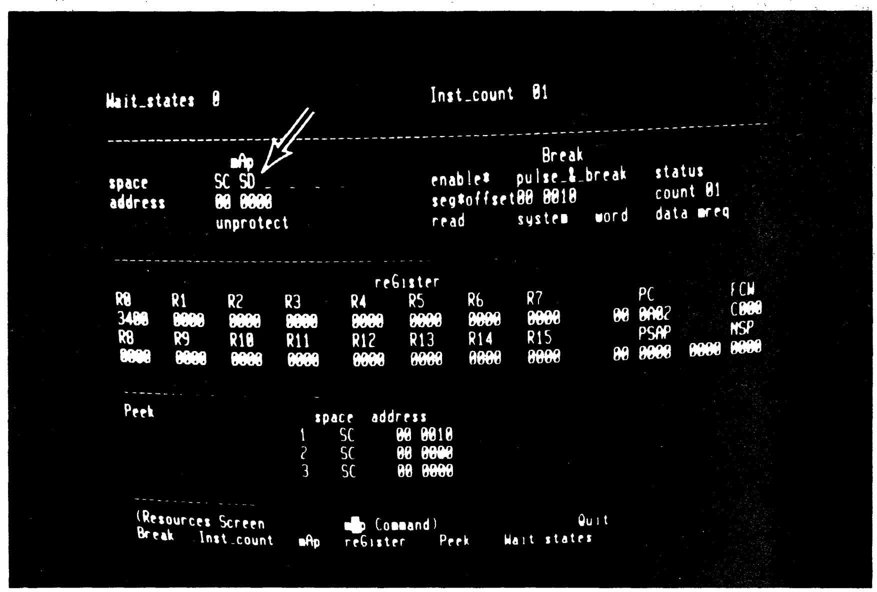


Figure 4-94. Modification of mAp Parameters

Step	Key Sequence	Commentary
48.	G, A, B, RETURN	The last action on this screen is to set up a new starting value for R0 with the reGister command.

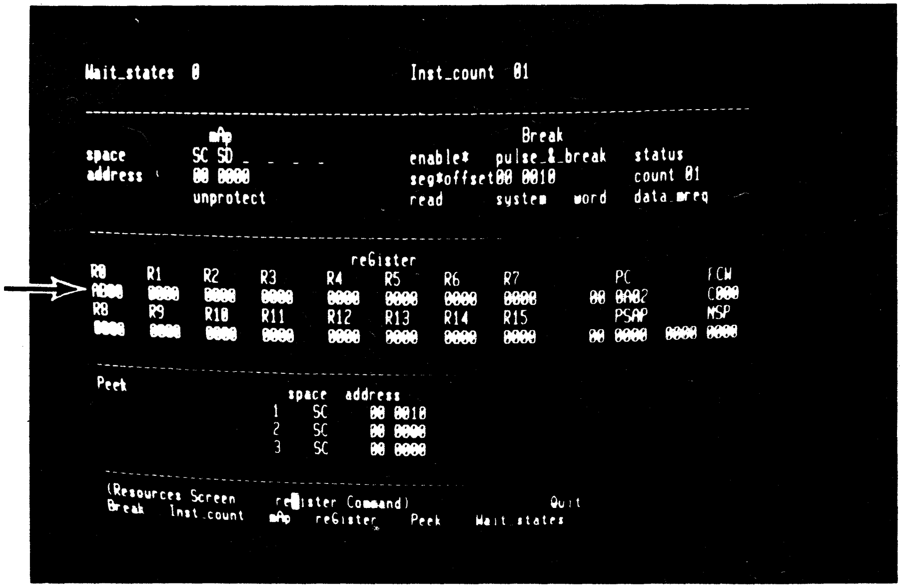


Figure 4-95. Modification of R0 Value

Step	Key Sequence	Commentary
49.	E, G	Start a new emulation. This time, the trigger fires almost immediately, and when the execution screen is redisplayed, you see that the contents of location 0010 have indeed changed from A80B to A81B. The Program Counter points to location 001C, the word after the instruction that caused the break condition to be met. The condition flags in the FCW reflect the fact that location 0010 holds a negative 2's complement number.

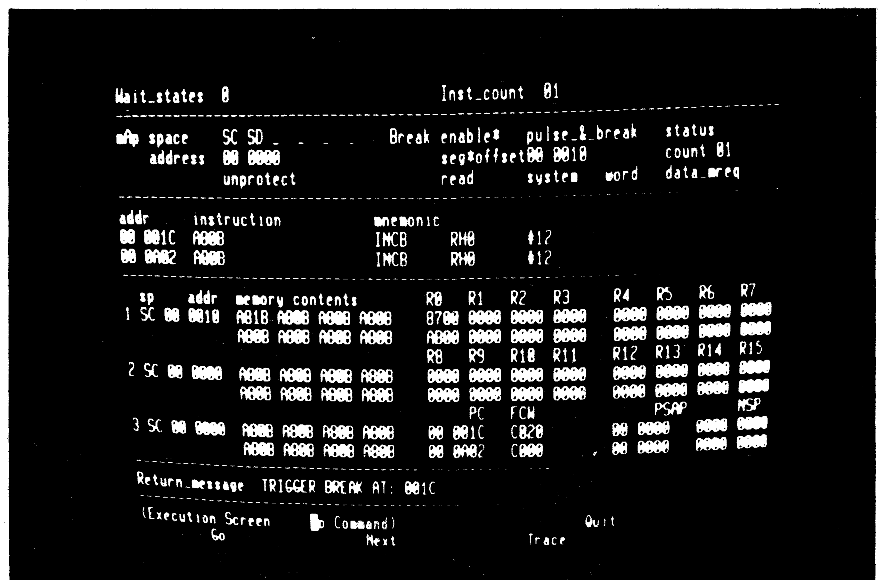


Figure 4-96. Trigger Break on Data Read

Step	Key Sequence	Commentary
50.	R, B, down, left, 5 RETURN	Associated with the breakpoint logic is a **pass counter.** If you load it with 51 hex (that is 81 decimal), the program loop is executed that number of times on the next emulation.
51.	E, G	After the emulation begins, there is a short delay before the breakpoint is encountered the number of times programmed. When the emulation ends, location 0010 has been incremented by 510 hex (51 x 10), showing that the correct number of passes has been made.

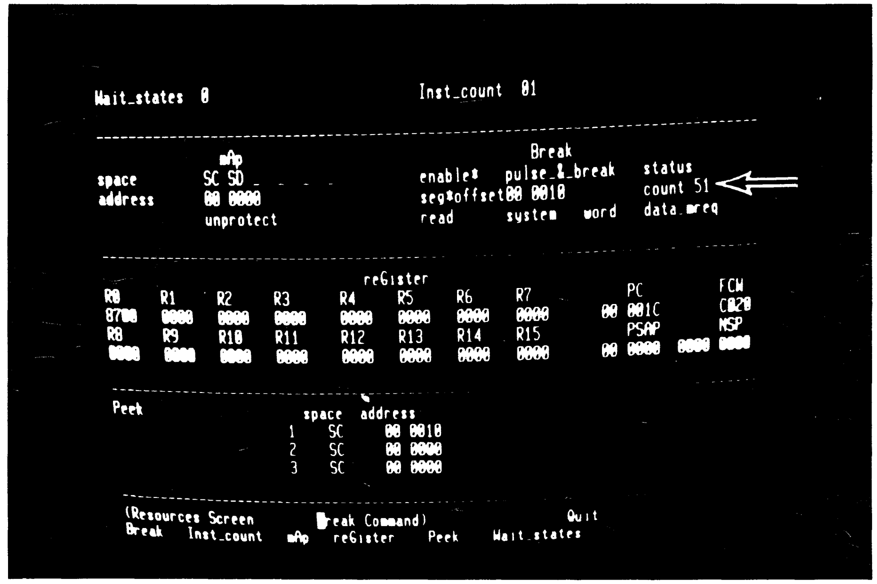


Figure 4-97. Adjusting Pass Counter

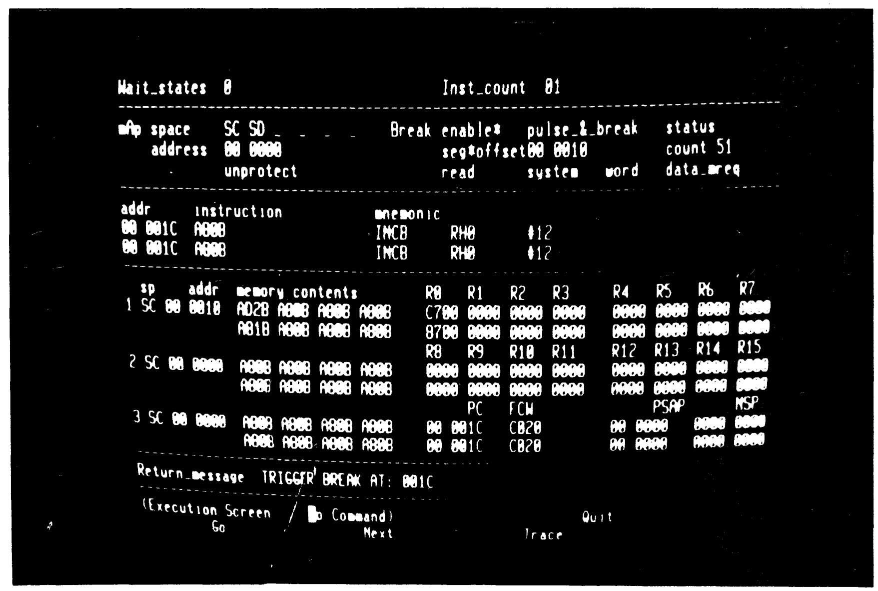


Figure 4-98. Break After Multiple Passes

Step	Key Sequence	Commentary
52.	R, A, up, 2, RETURN, B, space, RETURN	The INC instruction writes memory and can be used to show the Z-SCAN's **write protect** feature. To do this, disable the breakpoint and enable a write protect break.
53.	E, G	The next emulation terminates with a message warning of a write protect **violation.** Although the offending instruction has been executed, the contents of mappable memory remain unchanged and the data that the CPU attempted to write into memory is lost.

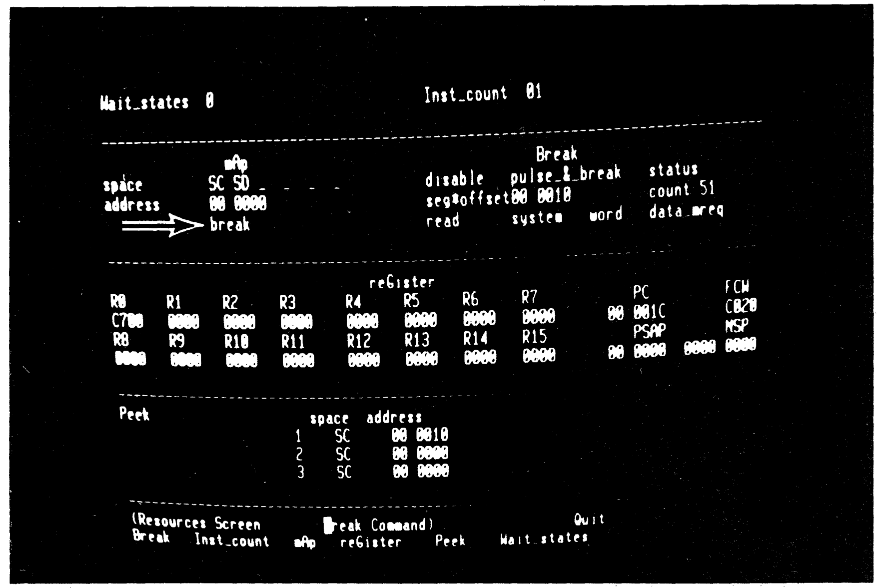


Figure 4-99. Selection of Write-Protect Break

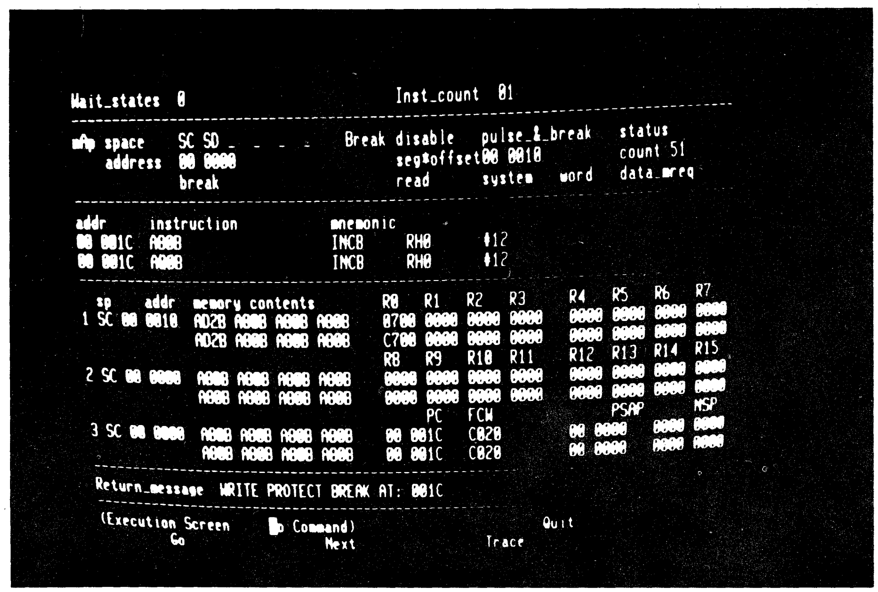


Figure 4-100. Break After Violation

Step	Key Sequence	Commentary
54.	R, A, up, space, RETURN	Clear the write protect break.

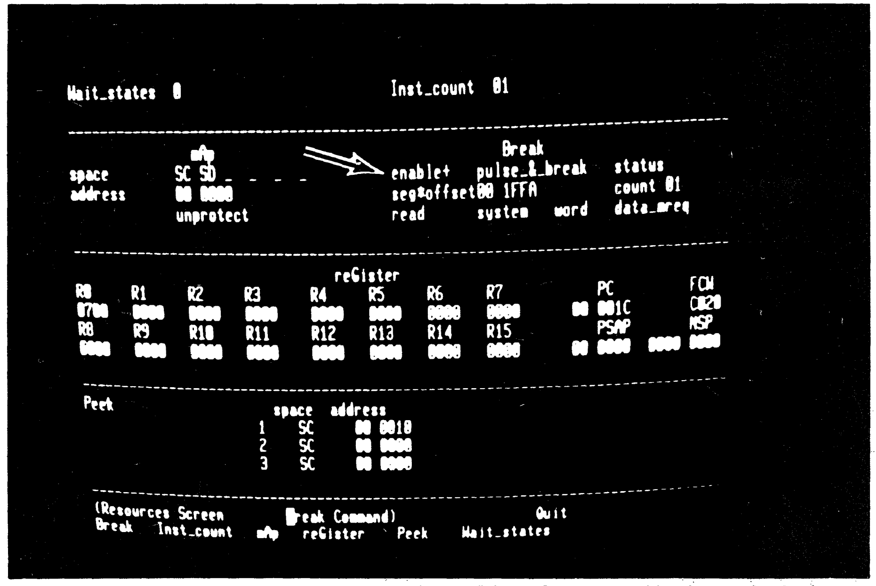


Figure 4-101. Set-up of Multiple Condition Break

Step	Key Sequence	Commentary
55.	B, 1, down, left, space, left, 1, F, F, A, RETURN	Now select a break on the first occurence of either any reference to segment 00, location 1FFA (in any address space) or any word read from system data memory. "enable+" designates this mode of operation.

Step	Key Sequence	Commentary
56.	E, G	Return to the Execution screen and run an emulation. It stops at location 1FFC because the address of the previous instruction has fired the trigger. The contents of location 0010 are unchanged, indicating that the instruction at location 0018 was not executed during the emulation.
57.	T, down	A trace terminates after only two instructions have been executed because a trigger is caused when the instruction at location 0018 performs a data memory read. Emulation stops as this event has precedence over the step count of 000B (11 decimal) instructions. A break message replaces the prompt that normally appears on the bottom screen line. The Peek display shows that the contents of location 0010 have changed.

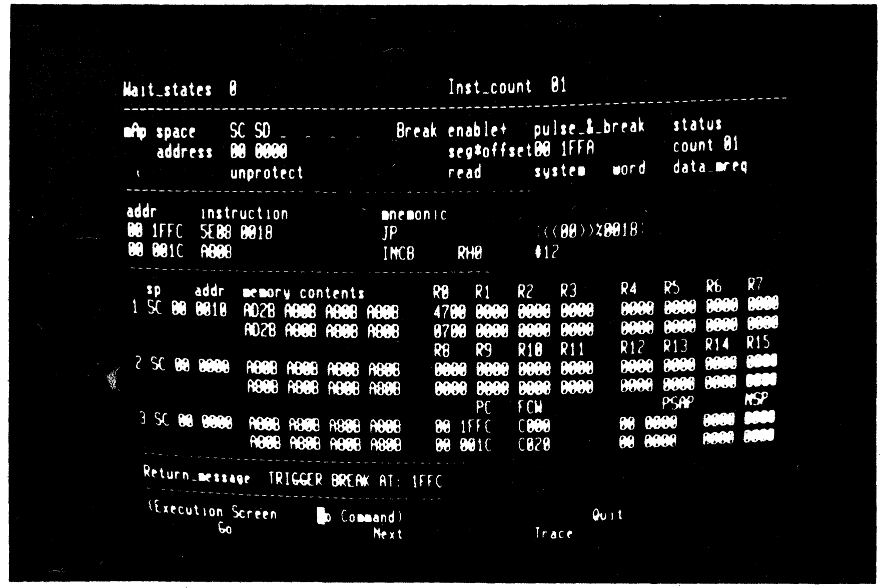


Figure 4-102. Break on Address Match

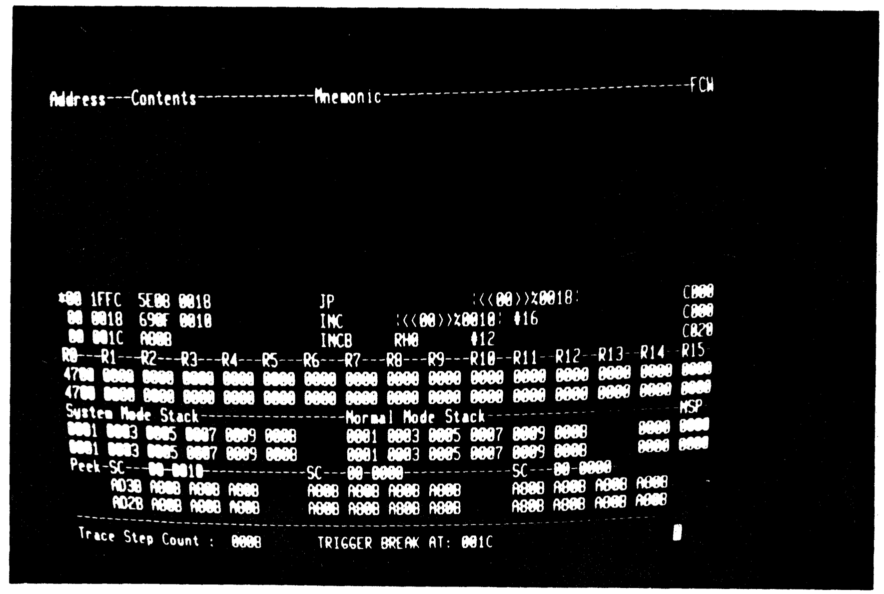


Figure 4-103. Data Read Break on Trace Screen

4.8 HOST SYSTEM USE WITH Z8001

The tutorial script continues on the next page. If your Z-SCAN is connected to a host system that supports the generation and downloading of Z8001 programs, perform steps 59 through 63, then move on to step 65. If the example program already exists on the host file system, you can skip all the steps except 63. If you do not have a suitable host, proceed directly to step 64.

Step	Key Sequence	Commentary
58.	RETURN, R, RETURN, **R**, A, 1, right, 1, right 1, right, 1, right, 1, right, 1, RETURN	The example program that is run in this part of the tutorial generates accesses to all six Z8001 memory spaces. Select the Resources screen and set up the mappable memory to respond to all types of access: code, data and stack references in both system and normal modes.

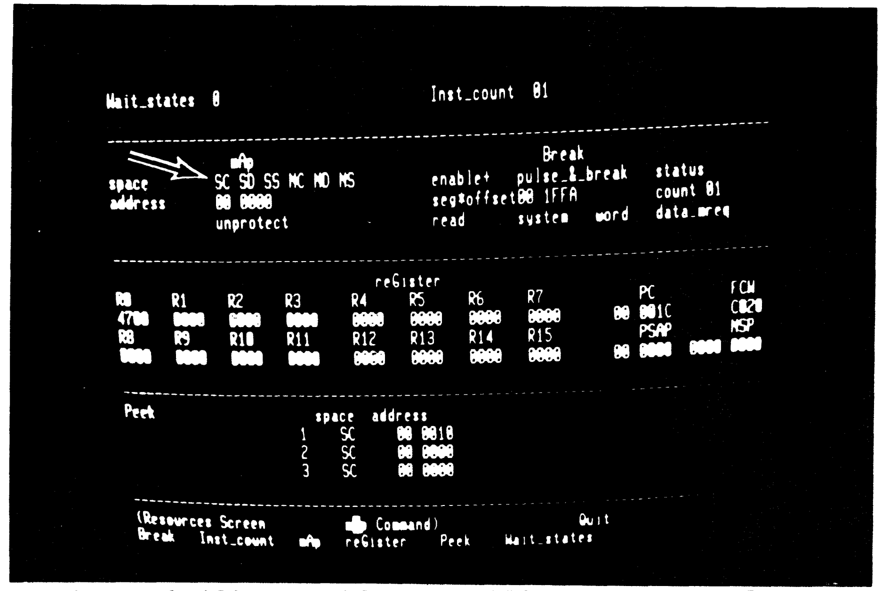


Figure 4-104. Enabling of All mAp Address Spaces

NOTE

If your Z-SCAN is connected to a host system that supports the generation and downloading of Z8001 programs, perform steps 59 through 63, then move on to step 65. If the example program already exists on the host file system, you can skip all the steps except 63. If you do not have a suitable host, proceed directly to step 64.

Step	Key Sequence	Commentary
59.	H	Before you can use the Z-SCAN **download** command, you must have Z8001 program to load. Your host's utilities and support programs can be used to create it. Type H to enter Transparent mode.
60.	Bootstrap your system	Unless it is already up and running, load the operating system of your host. For Zilog PDS 8000 systems, press the RESET button on the front panel of the system, then enter RETURN at the terminal keyboard. For ZDS/1 systems, press wait, then enter two returns. An operating system diskette must be present in drive zero or, for hard disk systems, the disks must be spinning. If you have a non-Zilog host, follow the bootstrap procedure described in its system manual.
61.	Enter, assemble and image the example program	Figure 4-105 shows an example program that is compatible with Zilog's Z8000 PLZ/ASM assembler, version 2.02 or later. The commands needed by the Zilog RIO operating system to create it are listed in Figure 4-106. Assemblers on non-Zilog hosts probably require changes in the syntax of the source. Changes are acceptable provided that the memory image of the final program corresponds to the information at the left of Figure 4-105. Refer to the host documentation for more information. The program appears with expanded commentary in Appendix B of this manual.

```
LOC       OBJ CODE
                              EXAMSEG MODULE
                                  $SEGMENTED
                                  $SECTION    EXAMSEG_P       ! Make imaging easy    !
                                  $REL     %0000
                              INTERNAL
                              NEW_STATUS_AREA:                ! Most entries unused !
0000 0000   C000                  RESET       ARRAY [ 2 LONG ] := [ %C000, INIT ]
0004 8000'  0044'
                                  $REL     %0010              ! Privileged instr.    !
0010 0000   C004                  PRIV_VECTOR ARRAY [ 2 LONG ] := [ %C004, INIT ]
0014 8000'  0044'
                                  $REL     %0018              ! System call          !
0018 0000   C000                  SC_VECTOR   ARRAY [ 2 LONG ] := [ %C000, BREAKER ]
001C 8000'  005A'
                                  $REL     %0028              ! Non-Maskable Int.    !
0028 0000   C008                  NMI_VECTOR  ARRAY [ 2 LONG ] := [ %C008, INIT ]
002C 8000'  0044'
0030 0000   0000                  PASS, LAST  WORD := 0       ! Data and stack area !
0034 0000   ...                   NML_STK     ARRAY [ 4 WORD ] := 0
003C 0000   0000                  SYS_STK RECORD [ID OLD_FCW WORD OLD_PC LONG] := 0
0040 0000   0000
0044                          GLOBAL INIT PROCEDURE           ! Set up control reg's!
                              ENTRY                           !  and both stacks.    !
0044 7600   00' 00'               LDA      RR0,|NEW_STATUS_AREA|
0048 7D0C                         LDCTL    PSAPSEG,R0
004A 7D1D                         LDCTL    PSAPOFF,R1
004C 760E   00' 44'               LDA      RR14,|SYS_STK + SIZEOF SYS_STK|
0050 7600   00' 3C'               LDA      RR0,|NML_STK + SIZEOF NML_STK|
0054 7D0E                         LDCTL    NSPSEG,R0
0056 7D1F                         LDCTL    NSPOFF,R1
0058 7F12                         SC       #%12               ! Trap into BREAKER    !
005A                          END INIT

005A                          INTERNAL BREAKER PROCEDURE      ! Demonstrate bus      !
                              ENTRY                           !  cycle types.        !
005A A9F7                         INC      R15,#SIZEOF SYS_STK ! Fix up system stack !
005C 670E   00' 3E'               BIT      |SYS_STK.OLD_FCW|,#14!Check previous mode !
0060 E604                         JR       Z,ELSE_            ! If mode was system   !
0062 7D02                             LDCTL   R0,FCW          !  set normal mode by  !
0064 A30E                             RES     R0,#14          !  clearing bit 14     !
0066 7D0A                             LDCTL   FCW,R0          !  of FCW;             !
0068 E808                             JR      FI_             !  else do I/O.        !
006A 2101   ABCD              ELSE_:  LD      R1,#%ABCD       ! Dummy port address.  !
006E 3D12                             IN      R2,@R1          ! I/O read             !
0070 3F13                             OUT     @R1,R3          ! I/O write            !
0072 3B05   1234                      SIN     R0,%1234        ! Special I/O read     !
0076 3B37   1234                      SOUT    %1234,R3        ! Special I/O write    !
                              FI_:                            ! Memory op's follow:  !
007A 7602   00' 30'               LDA      RR2,|PASS|         ! Internal operation   !
007E 2124                         LD       R4,@RR2            ! Data read            !
0080 93E4                         PUSH     @RR14,R4           ! Stk write            !
0082 29E0                         INC      @RR14              ! Stk read, stk write  !
0084 57E0   00' 30'               POP      |PASS|,@RR14       ! Stk read, data write!
0088 3304   FFA6                  LDR      LAST,R4            ! Code write           !
008C 7FEF                         SC       #%EF               ! Trap sequence        !
008E                          END BREAKER
                              END EXAMSEG
```

Figure 4-105. Z8001 Example Program

```
>
COPYRIGHT, ZILOG, INC. 1979
             All rights reserved.
             No part of this software may be copied or used without
             the express written consent of ZILOG, INC.

THURSDAY, NOVEMBER  1, 1979
RIO REL 2.2
%DATE 810424
FRIDAY, APRIL 24, 1981
%B;SET TABSIZE=4;EDIT EXAMSEG.S
B
EDIT 2.1
NEW FILE
INPUT
EXAMSEG MODULE
    $SEGMENTED
    $SECTION    EXAMSEG_P       ! Make imaging easy   !
    $REL        %0000
INTERNAL

        . . . . .

    SC      #%EF                ! Trap sequence       !
END BREAKER
END EXAMSEG

EDIT
QUIT
%Z8000ASM EXAMSEG
Z8000ASM 2.02
Pass 1 complete

   0 errors
Assembly complete
%IMAGER EXAMSEG.OBJ O=($=0000 EXAMSEG_P) {0000 0090} E=0044 O=EXAMSEG
IMAGER 2.0
7E BYTES LOADED
%EXTRACT EXAMSEG
RECORD COUNT = 0001  RECORD LENGTH = 0200   NO. OF BYTES IN LAST RECORD = 0090
ENTRY POINT = 0044  LOW ADDRESS = 0000  HIGH_ADDRESS = 0080  STACK SIZE = 0000
SEGMENTS:
0000 008F
%
```

NOTE

If the file EXAMSEG is created on a diskette, the first line of information output by the EXTRACT command will read as follows:

```
RECORD COUNT = 0002  RECORD LENGTH = 0080   NO. OF BYTES IN LAST RECORD = 0010
```

Figure 4-106. Z8001 Program Creation with RIO

Step	Key Sequence	Commentary
62.	BREAK	Return to the Z-SCAN monitor environment.

Step	Key Sequence	Commentary
63.	M, L, down, E, X, A, M, S, E, G, return	Set up and execute the Memory-io screen Load command. The program name is EXAMSEG (segmented example), and it is to be loaded into system code memory, segment 00. As the file is loaded, an incrementing number field appears toward the top left of the screen. This is a count of the number of records transferred from the host to target memory. Each record carries 30 or fewer bytes. When the loading is complete, the entry address of the program is displayed. If any error message appears, enter H and check the following: o Does the program file EXAMSEG exist? o Is its name correct? o Does the download utility LOAD exist? If no message appears when the command is executed, the host has not responded to the Load command sent by Z-SCAN. Terminate the load by entering BREAK, then type H and establish why this happened. When you have fixed the fault, return to the Z-SCAN monitor environment and type M, L, return.

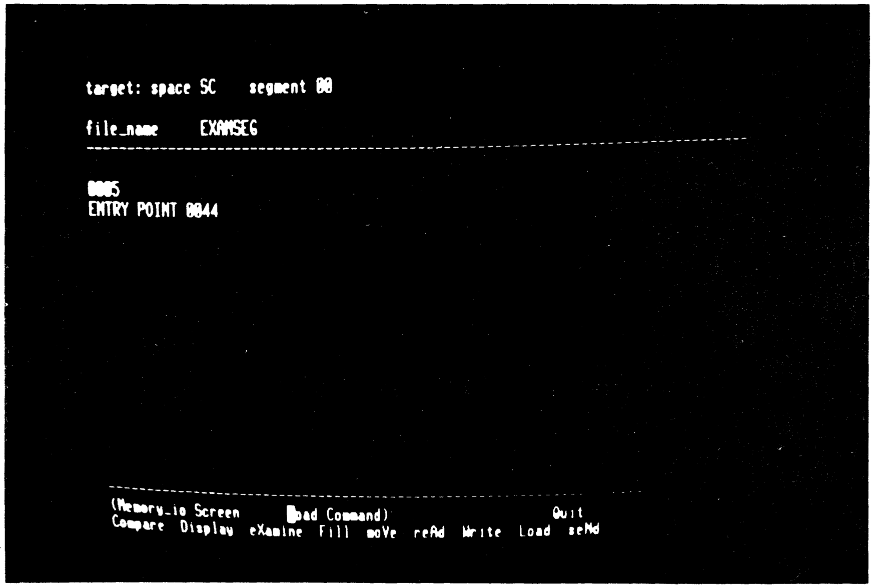


Figure 4-107. Loading of Z8001 Example Program

Step	Key Sequence	Commentary

If you have completed the four previous steps, skip the next one.

64.	M, V, 6, left, 0, 0, 9, 0, left, 5, E, 7, 0, RETURN	A copy of the example program shown in Figure 4-105 exists in the Z-SCAN monitor ROM. Use the Memory_io screen moVe command to copy it into the mappable memory.

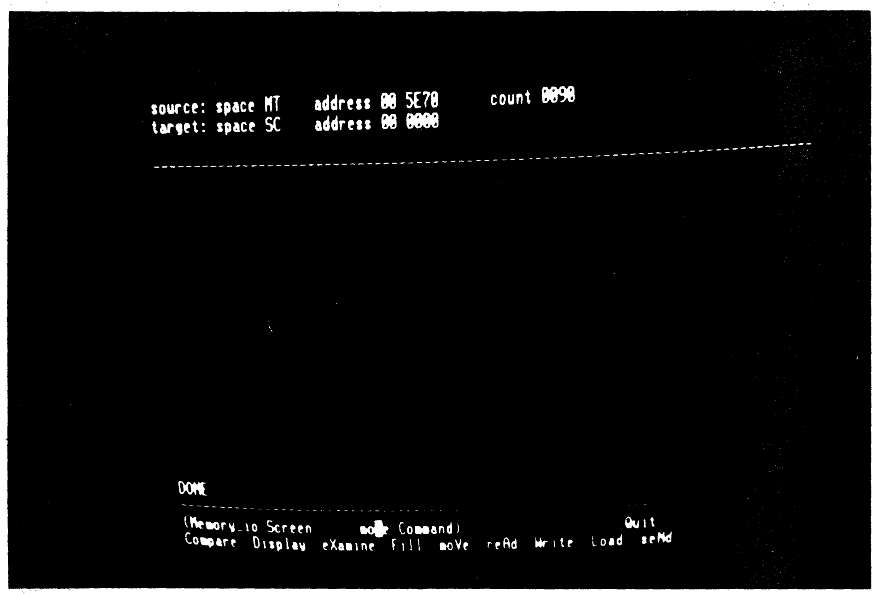


Figure 4-108. Copying Program with moVe

Step	Key Sequence	Commentary
65.	R, G, space, left space, left, 0, 0, 4, 4, RETURN	Set up the PC and FCW so that the program starts at location 0044 in system mode. At the same time, restore R0 to its default value.
66.	P, 1, left, 0, 0, 3, 0, down, 0, 0, 3, 4, right, 5, down, 2, left, 0, 0, 3, C, RETURN	The program has data, normal stack and system stack areas. Set up the Peek fields to monitor their contents before and after each emulation is run.

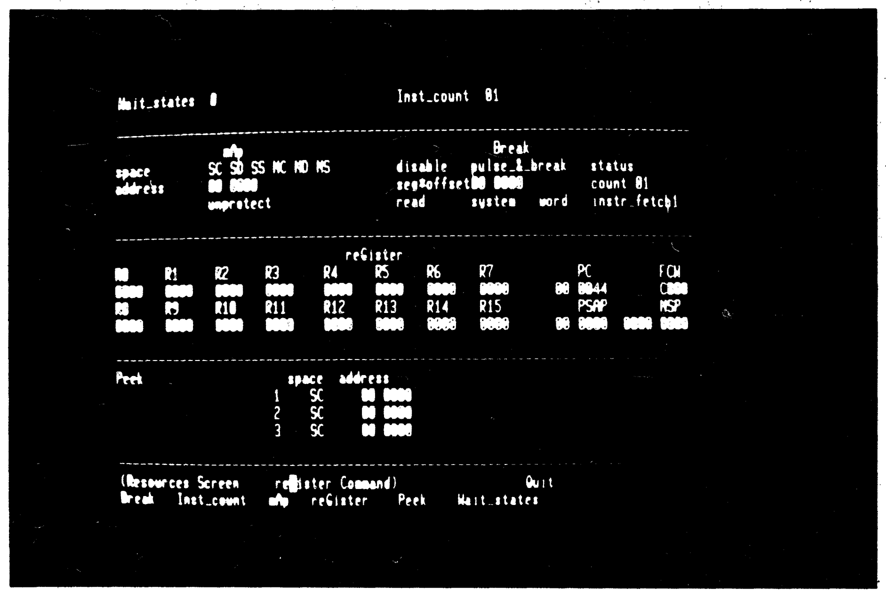


Figure 4-109. reGister Initialization

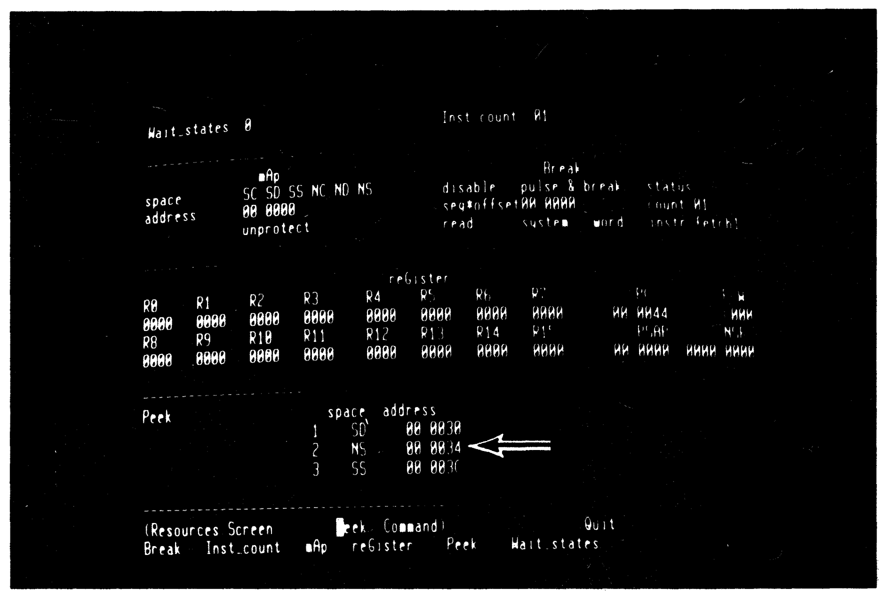


Figure 4-110. Set-up of Peek Parameters

Step	Key Sequence	Commentary
67.	B, 2, up, left, space, up, left, 0, 0, 4, 4, RETURN	Finally, set up a breakpoint on the first instruction of the initialization routine of the example program.
68.	E, G	The Z-SCAN is now ready to run the program. There is a trigger break after the first instruction is executed. At this point, the only change is in the PC value.

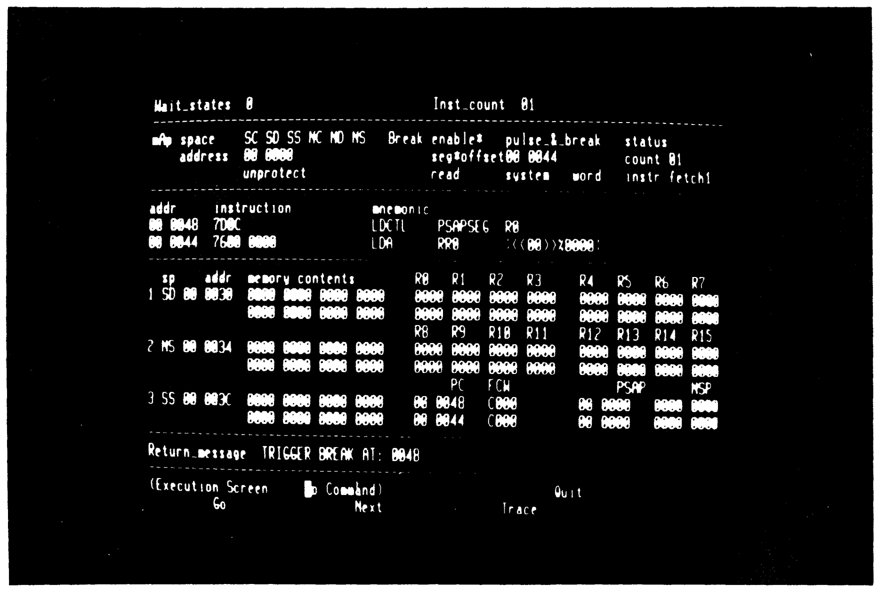


Figure 4-111. Emulation and Breakpoint

Step	Key Sequence	Commentary
69.	I, 6, down, 1, down	Trace the next five instructions, entering numbers to change the default Trace step count. At the end of the sequence, both system and normal stack pointers have been set up, changing the displays for the two **stack areas.** Notice that when the LDA instruction is used with a short offset address, the low byte of the address is loaded into both halves of the long word destination register. This is acceptable because the low eight bits of the segment register are ignored. Also, bit 7 of the PC segment number is a don't care. The final instruction, a System Call, pushes four words of data onto the system stack, producing a further change. The data also appears in the third Peek area.

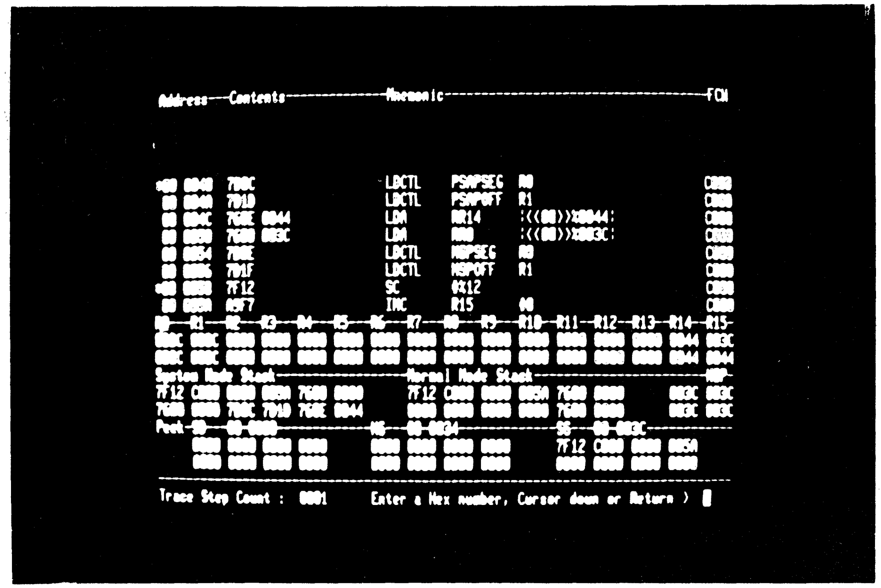


Figure 4-112. Tracing Initialization Routine

Step	Key Sequence	Commentary
70.	1, 0, 0, 0, down, BREAK	A large number of instructions can be traced by entering a hexidecimal number (botton right of menu area). Tracing begins when you enter cursor down. Let the display run for a while and observe that the program loops, alternately setting and clearing bit 14 of the FCW to move in and out of system mode. Tracing can be stopped at any time by entering the terminal BREAK key. The redisplayed memory content fields show that the contents of the data area and of both stacks have changed.
71.	RETURN, G, target RESET	Run the program. The emulation can be terminated with a target RESET because the status loaded from locations 0002 through 0006 in response to the input makes the CPU execute the instruction on which the breakpoint is set. The initial conditions of the program are not fully restored by the RESET because the data and stack areas may no longer be zero.

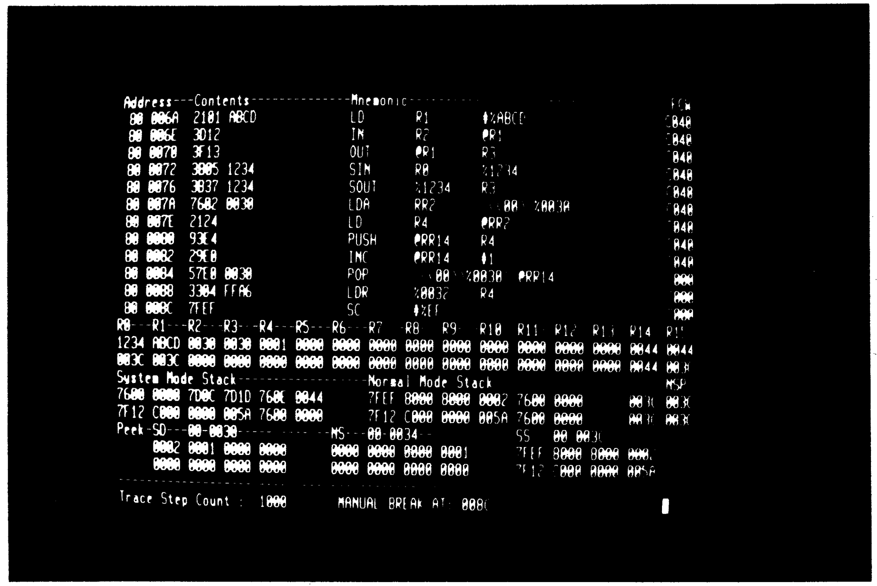


Figure 4-113. Trace of Main Routine

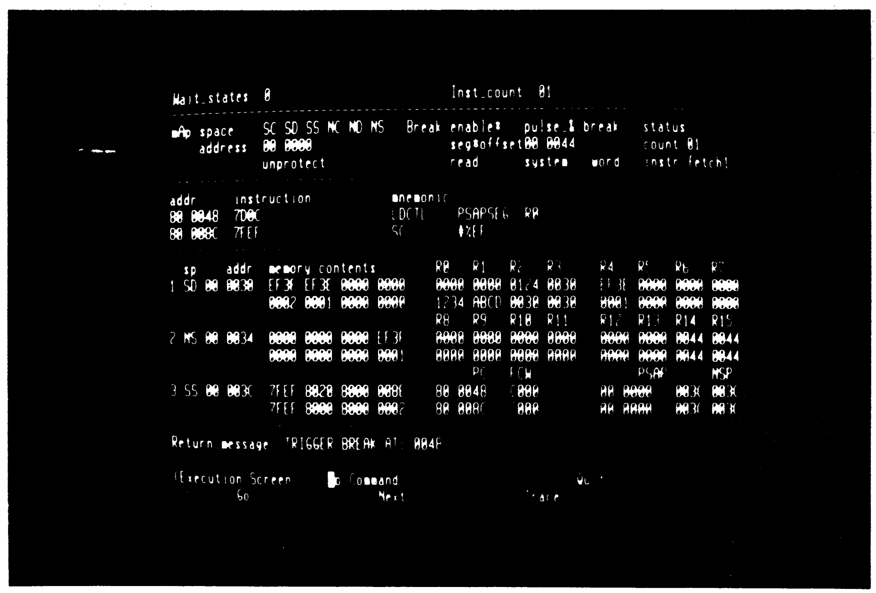


Figure 4-114. Trigger Due to Target Reset

Step	Key Sequence	Commentary
72.	RETURN, target NMI	A target NMI also terminates an emulation. Again, the initialization routine is entered in response to the input. The cause of the entry can be distinguished because the reset and NMI flag register values differ.
73.	R, B, down, 3, down, 1, left, A, RETURN	This tutorial does not explore the full possibilities of the program, which can generate a wide variety of bus cycle types in both system and normal modes. Experiment with it if you want to explore the Z-SCAN's features in more depth. As a start, set up a breakpoint on a system stack write of data pattern 0044.

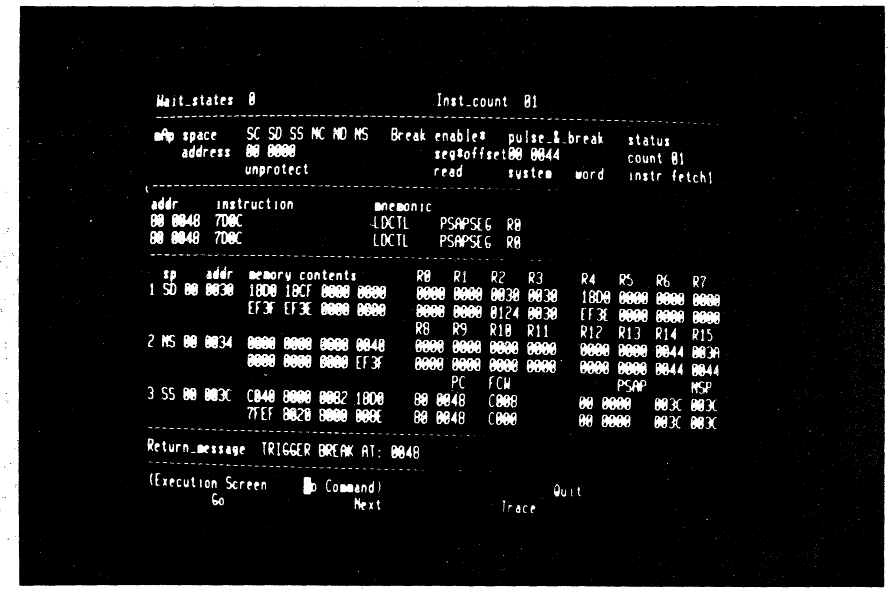


Figure 4-115. Trigger Due to Target NMI

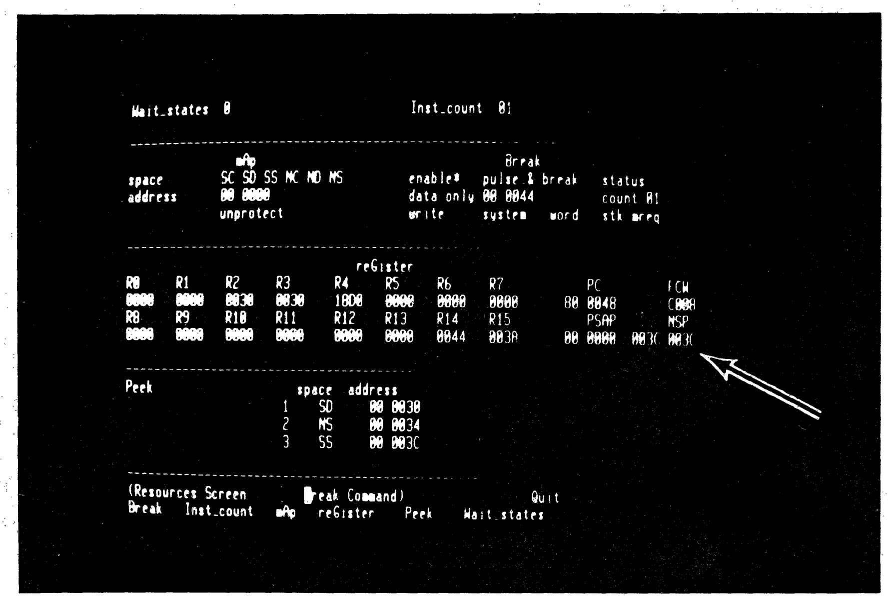


Figure 4-116. Set-up of Stack Write Break

Step	Key Sequence	Commentary
74. E, G		This last emulation can run for as long as four seconds before the instruction at address 0082 writes the data pattern matching the programmed break condition. The Z-SCAN may not stop the emulation before the next instruction is executed because the data match is detected only at the end of the last bus cycle of the INC instruction. Because of this, the next instuction, POP, is executed before the emulation terminates. This leaves the PC pointing to the LDR instruction.

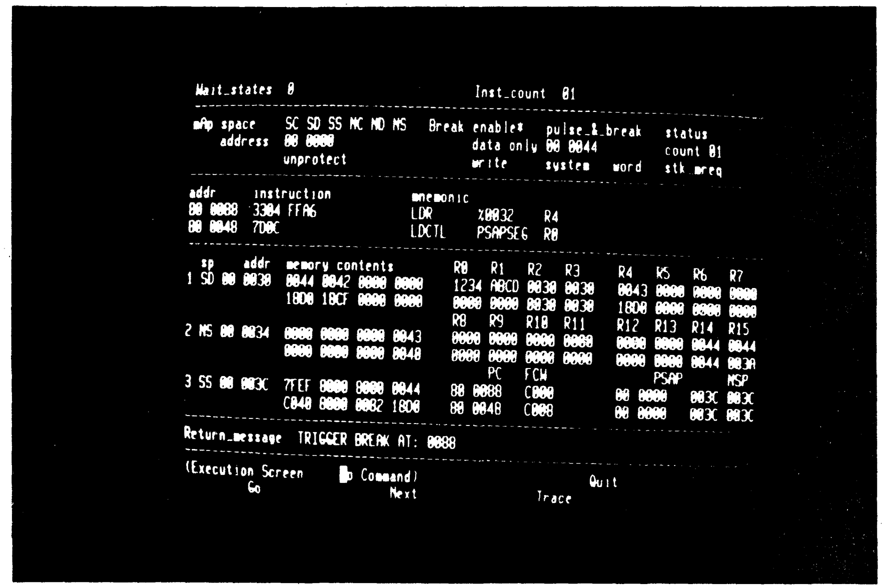


Figure 4-117. Break on Stack Write

4.9 CONCLUSION

This concludes the Z-SCAN monitor tutorials. They have shown many of Z-SCAN's features and most of its displays. A few commands have not been explored; these are discussed in Section 6. New users should now proceed to Section 5 which describes the connection of target hardware to Z-SCAN.

SECTION 5

TARGET HARDWARE CONNECTION

5.1 INTRODUCTION

The Z-SCAN 8000's major function is to replace a Z8001 or Z8002 microprocessor in a target system with an in-circuit emulator. This section details the method of connection. Readers are assumed to have some familiarity with the Z-SCAN monitor software. New users are advised to work through the tutorial in Section 4 before proceding to the connection of a target system.

While Z-SCAN is designed to mimic the Z8000 processors as accurately as possible, the characteristics of any microprocessor emulator inevitably differ slightly from those of the CPU it replaces. These differences and their impact on the behavior of Z-SCAN in certain types of target hardware are discussed. Designers of Z8000-based hardware should read this material. Users debugging existing designs may find that this section explains certain aspects of Z-SCAN's behavior in their target systems.

The combination of a Z-SCAN and a logic analyzer forms a powerful tool capable of real-time recording of logic signals in the target during emulations. The way Z-SCAN's break pulse output can be used to trigger the analyzer, or an oscilloscope, is described in the final part of the section.

5.2 USE OF THE EMULATOR CABLE

5.2.1 Clock Source

Z-SCAN is capable of operating either from its own 3.3 MHz internal clock or from an external clock supplied through the emulator cable from the target hardware. The external clock can have any frequency from 0.5 to 4.0 MHz.

If Z-SCAN is used without a target, as might be the case during the debugging of a non-hardware-dependent software module no larger than the 8K bytes of mappable memory, the internal clock source must be used. When Z-SCAN is connected to a target that has its own clock source, Z-SCAN must use the target's clock to ensure that its CPU operates at the same speed as synchronous logic elements in the target and to ensure successful emulation.

Changeover from internal to external clock is accomplished by moving a single jumper on the Z-SCAN printed circuit board. The jumper, designated E10, E11, E12, is located towards the front left of the board, as shown in Figure 5-1. The jumper is the only one on the board the user should alter, and it selects clock source as listed in Table 5-1.

Table 5-1. Clock Source Selection

Connection	Clock Source
E10 to E11	Internal
E11 to E12	External

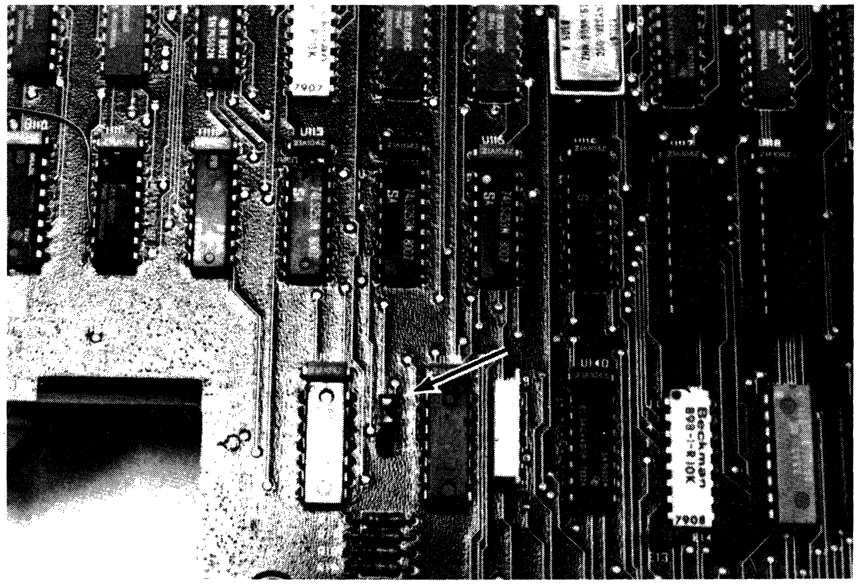


Figure 5-1. Clock Jumper Location

In order to select a new clock source, proceed as follows:

1. Switch the Z-SCAN power off by toggling the red power switch, located on the front panel, to the OFF position.

2. Remove the power cord from the socket on the rear of the unit.

--DANGER--

Failure to remove power from the unit prior to removal of the cover may result in exposure to hazardous voltages.

3. Remove the three screws and washers that secure the top cover of the unit at the top left, center, and right of the rear panel, as shown in Figure 5-2. Store the screws and washers in a safe place.

4. Grasping the rear of the top cover, lift it upwards and move it to the rear to release it from the front panel.

5. Locate the clock source jumper (see Figure 5-1) and move to the required position (see Table 5-1).

6. To replace the top cover, locate the front flange under the front bezel and swing the rear down. Make sure that the rear flange is inside the rear panel of the unit.

--DANGER--

Do not reconnect power to the unit until the top cover has been replaced and secured.

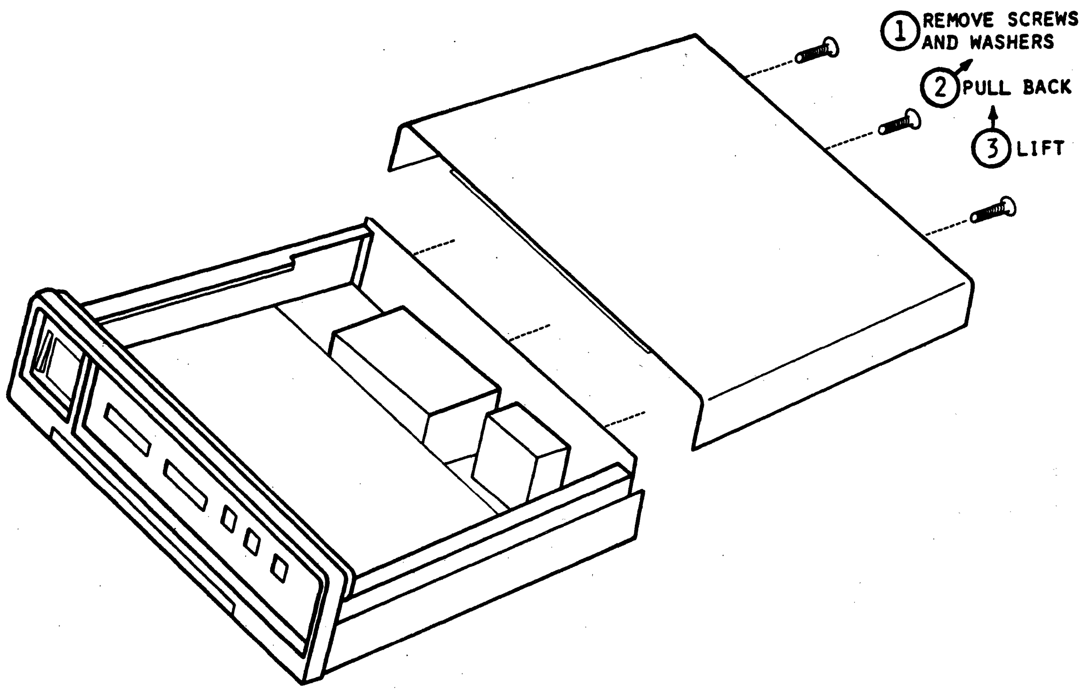


Figure 5-2. Z-SCAN Top Cover Removal

7. Replace the three screws and washers removed in step 3.

8. Reconnect the power cord to the rear of the unit, but **do not switch power on at this stage.**

9. If the external clock was selected in step 5, procede to Section 5.2.2 below, which describes the connection of the emulator cable. Z-SCAN requires the connection of a target in order to function when the external clock has been selected.

10. If the internal clock was selected in step 5, the unit can now be powered on by moving the front panel power switch to the on position. Correct operation can be verified by following the procedure described in Section 3.6.

5.2.2 Connection of the Emulator Cable

Two emulator cables are shipped with each Z-SCAN. The 40-way cable is used for Z8002 emulation and the 48-way for emulation of the Z8001. Before connecting either of the cables to the Z-SCAN, check that the correct processor is installed. The processor type is displayed on the System screen. Section 3.9 describes how to change the processor.

When the correct processor is installed, the target system can be connected to Z-SCAN with the emulator cable. To do this, procede as follows:

1. If Z-SCAN is not already switched OFF, switch it OFF using the front panel POWER/OFF switch.

2. Turn the target system OFF.

3. If the Z-SCAN unit is equipped with a Z8002 CPU, plug the 40-pin flat cable connector into the right-hand socket marked Z8002 on the front panel of the Z-SCAN. If a Z8001 is installed, plug the 50-pin flat cable connector into the left-hand socket marked Z8001. **The stripe indicating Line 1 should be to the right of the cable.**

--CAUTION--

> **It is possible to insert either connector upside-down. Incorrect connection can result in damage both to Z-SCAN and to the target system.**

4. Remove the plastic pin protector from the DIL header and store it in a safe place.

5. Plug the header into the CPU socket in the target system, making sure that the pin marked "1" on the header is mated with pin 1 of the socket. Figure 5-3 shows the Z-SCAN unit correctly connected to a target system.

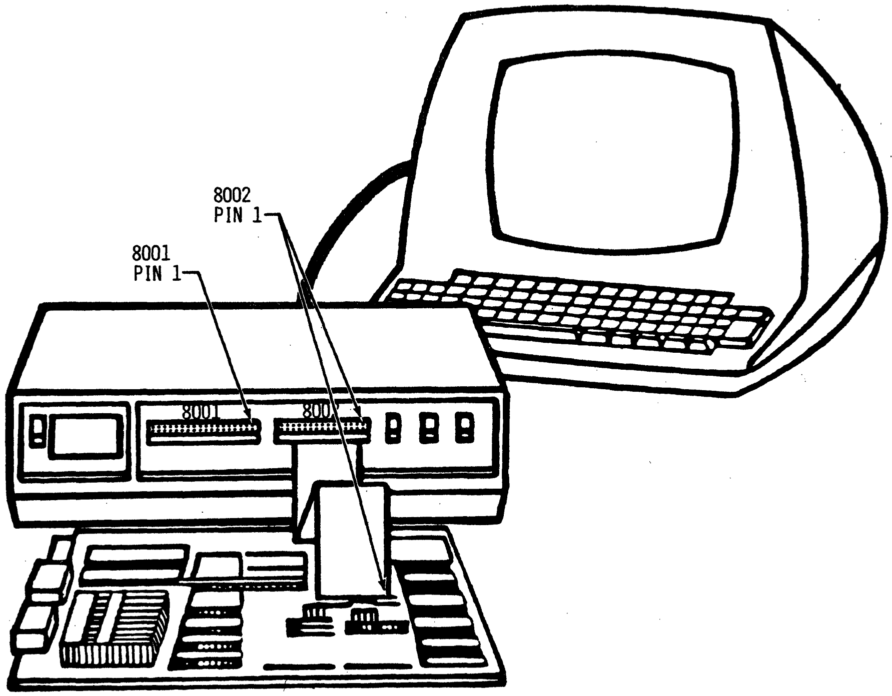


Figure 5-3. Z-SCAN and Z8002 Target System Connections

5.2.3 Checkout of Z-SCAN with Target System

To check that Z-SCAN can operate with the newly connected target system, the following test should be carried out. Note that this simple procedure only verifies that the target is correctly connected and is providing an adequate clock signal to the unit. It does not verify that the target is functional in any other respect.

1. Turn on the target system.

2. Power the Z-SCAN by moving the front panel Power/OFF switch to the POWER position.

3. Place the TARGET/MONITOR switch in the MONITOR position.

4. Toggle the RESET switch.

5. On the keyboard, enter RETURN once. The Z-SCAN sets its baud rate and displays the terminal menu.

   If the terminal menu does not appear, check the following:

   - Emulator cable is correctly connected.
   - Target system is powered.
   - Target clock circuitry is functioning properly.

- Target clock rate is within Z8000 specification (0.5 - 4.0 MHz).

- Target clock meets minimum high- or low-time requirements of Z8000 CPU (105 ns) and has proper rise time (less than 20 ns).

- Clock source selection jumper is correctly installed (see Section 5.2.1).

- Emulator cable assembly is not damaged (see Section 5.2.4).

After the problem has been identified and corrected, the Terminal Selection screen should appear after the checkout procedure. If problems persist despite the availability of an adequate clock from the target system, the Z-SCAN may require maintenance. In this event, the user should contact the nearest Zilog sales office.

6. Select a terminal number, then enter return. If the target system contains dynamic memory components, enter the key sequence:

RETURN, cursor down, 1, RETURN

This updates the status_to_target field on the System screen from internal_op to refresh. For further details see Section 5.4.3.

5.2.4 Care of the Emulator Cable

The emulator cable assembly is 18 inches (45.7 cm) long and is constructed from a special high-quality flat cable that has a ground wire adjacent to each signal wire for optimum transmission characteristics. Standard flat cable connectors cannot be used with this type of cable. If the assembly is damaged during use, a replacement must be obtained from Zilog. Z-SCAN's performance will be degraded if a substitute is constructed with standard cable and connectors.

While the assembly is quite sturdy, it can be damaged by incorrect handling. Observe the following precautions to minimize the possibility of damage:

- **Never pull on the cable. Use the procedures detailed below to remove the connectors from the Z-SCAN or from the target system.**

- When the cable is not plugged into a target system, cover the exposed pins on the emulator plug with the pin protector supplied with the unit. If the protector is lost, a small pad of conductive foam or styrofoam is an acceptable substitute.

- Once the cable has been connected to the Z-SCAN, do not remove it unless absolutely necessary. When removal is required, grip both sides of the cable and the connector between the thumbs and forefingers of both hands. Move the connector up and down slightly while gently pulling until it is free.

- To remove the emulator plug from the target system CPU socket, use a small screwdriver as a lever to lift each end of the Augat header from the socket in the target a little at a time. When the plug is free, cover the exposed pins with the pin protector supplied with the unit.

- If target hardware modifications are made, remove the cable from the target system to avoid contact with a hot soldering iron.

5.3 FRONT PANEL SWITCHES

The three switches at the right of the Z-SCAN front panel were described in Sections 3 and 4. This section describes the exact effect of each of the four types of input that these switches can generate to Z-SCAN. See Table 4-2 for details of how to generate each type. Z-SCAN's response to a particular input is determined primarily by the operating mode at the time the input is received (see Tables 5-2 through 5-5).

At no time does Z-SCAN drive the target system's RESET- or NMI- signals. Thus, while the Z-SCAN CPU responds correctly to a target RESET or NMI generated with the Z-SCAN front panel switches, circuitry in the target hardware that relies on these signals' being active does not respond. This makes it possible that the behavior of the target following a front panel reset or NMI will differ from that which occurs when either signal is generated by the target itself.

Table 5-2. Response to Monitor RESET Input

Operating Mode Before Monitor RESET	Operating Mode After Monitor RESET	Notes
Monitor	Monitor	The CPU and Z-SCAN hardware is RESET to its initial state. All information about the previous state of Z-SCAN is lost. Type RETURN to set baud rate.
Host	Monitor	
Target	Monitor	

Table 5-3. Response to Monitor NMI Input

Operating Mode Before Monitor NMI	Operating Mode After Monitor NMI	Notes
Monitor	Monitor	The input is ignored.
Host	Monitor	The input is ignored.
Target	Monitor	This is the Z-SCAN's manual BREAK.

Table 5-4. Response to Target RESET Input

Operating Mode Before Target RESET	Operating Mode After Target RESET	Notes
Monitor	Monitor	Returns Z-SCAN to its initial condition.
Host	Monitor	Has same effect as Monitor reset.
Target	Target	Has the same effect as the target system's RESET- input to Z-SCAN. The The Z8000 CPU in Z-SCAN is reset. All other Z-SCAN hardware is unaffected. See Section 7.4 of the <u>Z8000 CPU Technical Manual</u>.

Table 5-5. Response to Target NMI Input

Operating Mode Before Target NMI	Operating Mode After Target NMI	Notes
Monitor	Monitor	The input is ignored.
Host	Host	The input is ignored.
Target	Target	Has the same effect as the target system's NMI- input to the Z-SCAN. The Z8000 CPU in Z-SCAN will respond to an NMI-. All other Z-SCAN hardware is unaffected. See Section 7.6 of the <u>Z8000 CPU Technical Manual</u>.

5.4 HARDWARE DESIGN AND DEBUGGING WITH Z-SCAN

The Z-SCAN 8000 has been designed to emulate the Z8000 CPUs faithfully in both new and existing hardware designs. This means that any target that operates correctly when a CPU chip is installed should also operate correctly when a Z-SCAN emulator is used in place of the CPU. The converse is also true. However, the ac and dc characteristics of Z-SCAN, and under certain circumstances, its bus signal sequences differ slightly from those of an actual CPU. These differences arise from the buffering required to isolate the Z-SCAN CPU from possible faults in the target system and from the need to prevent execution of the Z-SCAN monitor software from affecting the target system.

The remainder of this subsection details the areas in which differences exist and describes their possible effects on emulation and debugging. Hints to designers allow potential problems to be avoided before they arise. The hints in general reflect conservative design practices and ensure that equipment can be produced reliably and repeatably once the design has been finalized. Additionally, each paragraph suggests ways in which small problems in existing target hardware designs can be overcome or circumvented.

5.4.1 Emulator DC Characteristics

The dc characteristics of the Z-SCAN emulator differ from those of an actual CPU in three respects:

- **Input Loading:** Z8000 CPUs load inputs very lightly (no more than 10 uA and, except in the case of CLK, less than 10 pF). The Z-SCAN, in contrast, loads each input with 30 pF and a low-power Schottky TTL buffer (200 uA). In addition, the NMI-, NVI-, SEGT-, WAIT-, RESET-, DS- and VI- inputs have 10k pullups for an additional load of 500 uA.

- **Output Drive:** Z8000 CPUs are specified with a load of 100 pF and 2 mA. Because it has low-power Schottky TTL drivers, Z-SCAN can drive a much greater load.

- **Input Levels:** The majority of Z8000 inputs are completely TTL compatible. Two are not: CLK has more stringent high- and low-level requirements, and RESET- requires a slightly greater input high level. In contrast, all inputs to Z-SCAN are TTL compatible.

The electrical differences between a Z-SCAN and the Z8000 CPUs make it possible (though unlikely) that a target system could work with Z-SCAN but not with a CPU, or vice-versa. Such problems can easily be avoided at the design stage by adopting a few simple standards:

- **Clock Driver:** Never attempt to drive the CLK pin of the CPU directly from a TTL output. A special drive circuit capable of meeting the stringent requirements of the Z8000 is required. The Zilog application note A Small Z8000 System (document #03-8060) details a suitable design. A TTL output with a pullup resistor is not a satisfactory alternative.

- **Reset Driver:** If RESET- is driven by a TTL output, add a pullup resistor. The value is not critical: 4.7 K will do.

- **Bus Loading:** Do not attach too many loads directly to the bus signal pins of the CPU. As a rule of thumb, Z8000 processors can accommodate up to ten NMOS loads plus one low-power Schottky TTL load on each bus signal line, provided that the total length of the line is not greater than 8 in (20 cm) of printed circuit track. Greater loading is likely to exceed the capacitive drive capability of the CPU, even if dc loading limits are not exceeded. If there is any doubt about loading levels, or if bus signals are to be carried between circuit boards, use buffers.

Emulation problems arising from the differences between the dc characteristics of Z-SCAN and those of a Z8000 CPU are likely to show one or more of the following symptoms:

- **Intermittency:** The symptoms appear and disappear unpredictably.

- **Temperature Sensitivity:** The symptoms are seen only when Z-SCAN or the target system is warm and can be removed by cooling a particular component in the target system.

- **Voltage Sensitivity:** Raising or lowering the supply voltage in the target system affects the symptoms.

- **Locality:** Z-SCAN is able to access all features of the target system except those associated with a particular component or logic block.

If it is established that the Z-SCAN capacitive loading is increasing access times in the target to an unacceptable level, and that the target is capable of meeting the worst case ac specification of the Z8000, a temporary solution is to replace the target memory or I/O components with faster parts. Alternatively, provided that full-speed emulation is not required, the Z-SCAN Wait_states command can be used to relax access time requirements. Section 6.10.6 gives more details.

In general, quick fix solutions to such problems are not recommended because they probably indicate a marginal hardware design which, even if it works correctly with a CPU in prototype form, could suffer from repetitative or reliability problems when it is moved into production. The user is urged to determine the source of the problem and incorporate a permanent solution into the target hardware.

5.4.2 Emulator AC Characteristics

The ac characteristics of Z-SCAN differ from those of a Z8000 CPU because of delays introduced by signal buffering. The differences are minimized by using a factory-selected CPU in Z-SCAN.

Problems might also occur in synchronous logic in the target. Typically, such logic uses the system clock to latch signals coming from the CPU. For example, a dynamic memory controller might latch MREQ- on the rising edge of CLK-. The Z8000 has been designed to allow comparatively long set-up times in such cases, so it is unlikely that the slight reduction of time that results when Z-SCAN is used will cause problems. If a problem does occur because the timing skew introduced by the Z-SCAN is unacceptable to the target, it can usually be solved by introducing extra delay in the clock path to the synchronous logic in the target. Designers should be aware that such a solution may affect the access time requirements of memory or I/O components controlled by the synchronous logic.

5.4.3 Dynamic Memory Refresh

Z8000 microprocessors have a feature that allows them to refresh dynamic memory components automatically with a minimum of external logic. This is described in Chapter 8 of the Z8000 CPU Technical Manual (document #00-2010-C). Zilog's application note A Small Z8000 System suggests a suitable logic design and shows the relationship between the contents of the upper byte of the refresh register and refresh rate.

Z-SCAN supports automatic refresh before, during, and after emulations to preserve the integrity of the contents of dynamic memory in the target system. Z-SCAN itself contains no dynamic memory and so does not require refresh to be enabled in order to operate correctly.

Refresh is controlled by the Z8000 refresh register. Z-SCAN does not exercise as close control over the contents of this register as it does over those of others, partly because changes in its contents are largely independent of the code which is being executed, and also because some of its bits are write-only bits. For these reasons it does not appear on the Resources or Execution screens (Sections 6.10 and 6.11). Z-SCAN only alters the contents of the refresh register when the status_to_target field on the System screen is changed by the user. When refresh is selected, the register is loaded with %9E00; when internal_op is selected, a value of %0000 is loaded. The System screen is described in Section 6.8.

The hexadecimal value %9E00 causes the CPU to perform a refresh operation every 60 clock cycles. At a 4 MHz clock rate, this results in 128 refresh cycles every 1.92 ms, sufficient to satisfy the worst case requirements of typical dynamic RAM components. For targets in which the clock rate is significantly lower or in which a higher refresh rate is required, the user must change the value in the refresh register if worst case requirements are to be met. The alteration can be accomplished in one of two ways:

- Run an emulation of the initialization portion of the target application software. This should contain code that loads the refresh register with the value required by the target.
- Using the eXamine command on the Memory_io screen, load the opcode %7D0B (LDCTL REFRESH,R0) into any available RAM location. Use mappable memory if all the target memory is dynamic. Then use the Resources screen reGister command to load the required refresh register value into R0, the location of the instruction into PC, and %4000 into FCW. Finally, step through the instruction with the Execution screen Next command.

It is worthwhile to consider the exact effects of the two possible status_to_target values on the bus signals in the target system (Table 5-6).

Table 5-6. Monitor Mode Target Signals

status_to_target	ST_{0-3}	AD_{0-15}	AS-	MREQ-	DS-
internal_op	internal op (0000)	active	active	3-stated	inactive*
refresh	internal op (0000) or refresh (0001)	active	active	active	inactive*

*DS- is held high by a 10 kilo ohm pullup resistor

When internal_op is selected, every bus transaction generated by the monitor appears to be an internal operation to the target system. Because AS- is still active, self-refreshing (pseudo-static) memories in the target retain their contents provided that they use AS- and not MREQ- as a clocking signal.

When refresh is selected, most Z-SCAN monitor bus transactions appear as internal operations to the target system. They differ from the internal operations generated by an actual CPU because MREQ- may be active. Refresh cycles, generated when the CPU refresh rate counter times out, present refresh status to the target.

5.4.4 Target Memory and I/O Access

When Z-SCAN accesses target memory during emulations, the transactions it generates are identical to those that would be generated by a Z8000 CPU. However, when such accesses are made by the Z-SCAN monitor software, there may be differences. All the commands available on the Memory_io screen (see Section 6.9) cause Z-SCAN to perform memory or I/O operations in the target system. The Peek and current instruction fields on the Execution screen also require target memory accesses in order to be updated.

The Memory_io screen reAd and Write commands access target byte and word standard and special I/O ports using operations exactly like those produced when I/O instructions are executed by a program running under emulation. For target memory accesses, however, Z-SCAN always uses the same width of data (word or byte) for each type of operation, independent of the width (long word, word, or byte) selected by the user for the display of information. Table 5-7 lists the operation types, together with the transactions generated by Z-SCAN.

Table 5-7. Z-SCAN Target Memory Access Transactions

Operation	Transaction	Notes
Memory read	Byte read	
Memory write	Byte read, word write	A read-modify-write sequence for each byte to be written accommodates memories that do not support byte writes.

Z-SCAN's choice of transaction types should cause no problems in most target systems. There are, however, some unusual design configurations where target accesses may produce unexpected results:

- **Memory that does not support byte reads:** A correctly designed Z8000 memory control circuit does not need to distinguish between byte and word reads and hence implicitly supports both (see A Small Z8000 System). If, for any reason, a target's memory does not respond to byte read transactions, the Z-SCAN monitor will not be able to access that memory.

- **Systems using B/W- as a memory bank select signal:** The Z-SCAN monitor expects the same memory space to be accessed by both byte and word transactions. The Z-SCAN memory modification commands cannot be used in unorthodox target systems that use the B/W- (Byte/Word) signal to choose between two separate memory banks.

- **Word-wide, memory-mapped I/O:** The Z-SCAN memory modification commands cannot operate on write-only, word-wide memory-mapped I/O, nor do they give correct results on memory-mapped ports that require all 16 bits to be written or read in a single operation.

5.4.5 Interrupts and Traps

Z-SCAN terminates all emulations by giving the CPU a non-maskable interrupt. This interrupt has a higher priority than the three other external interrupts supported by the Z8000--segment trap, vectored and non-vectored. This means that all break conditions--step, manual, trigger and write protect --have priority over interrupts generated by the target hardware, including target NMI (see Section 6.10.1).

The Trace command (see Section 6.12.3) and the Next command (see Section 6.12.2), if used with an instruction count of one, give the CPU an NMI after each user instruction. This prevents the acknowledgement of any target interrupt, even though the target program is being executed. This does not prevent interrupt service routines from being traced, but it does mean that they cannot be entered while another part of the program is being traced.

The Z8000 CPU always fetches one instruction that it does not execute after it has accepted an interrupt or trap. See Section 9.4.5 of the Z8000 CPU Technical Manual for further details. The aborted operation is indistinguishable from any other instruction fetch, first word, bus cycle, and so can trigger the Z-SCAN breakpoint logic if suitable conditions have been set up. When this happens, the state of the user program following the breakpoint will differ from the state expected.

Unexpected breaks of this type do not corrupt the target program: if emulation is resumed, the instruction is fetched and executed normally after the target handler interrupt is executed.

5.4.6 Memory Management Considerations

Many applications use a Z8001 CPU in conjunction with a Z8010 Memory Management Unit, or some other memory management hardware. These external components increase system reliability and flexibility by serving two functions:

- Translation of logical address information from the CPU to physical addresses for memory components.

- Protection of memory from invalid access by the CPU.

When the Z-SCAN mappable memory is used in conjunction with a target system that has a memory manager, the user should be aware that the mappable memory is fixed in logical address space during emulations. It responds directly to the untranslated addresses output by the CPU. It cannot be relocated or protected by the action of the memory manager.

The protection attributes of target system memory may be violated by a program running under emulation. In such cases, the MMU activates the SEGT- signal and the user's trap handler is entered to deal with the problem. It is also possible for target memory accesses generated when Z-SCAN is in monitor mode to violate protection attributes. For example, the user could use the Fill command on a write protected memory area. An external memory manager responds to these violations by asserting SEGT-, but the monitor does not acknowledge the signal, and so it remains outstanding until the next emulation starts. The user's trap handler is then entered to deal with a violation that is not caused by the user program.

A message displayed at the lower right of the System and Trace screens warns the user if SEGT- is asserted before an emulation is started (see Section 6.12). If the user does not want the trap routine to be entered, the Memory_io screen Write command must be used to reset the memory management hardware before emulation.

5.4.7 Direct Memory Access

Z-SCAN fully supports Direct Memory Access (DMA) in the target system while emulations are running. Note that if a trigger or manual break condition occurs during a DMA operation, emulation does not stop until the DMA controller releases the bus, because the CPU cannot service the break request until it has regained control of the bus. Also, because BUSACK- is not a term in the breakpoint equation, Z-SCAN cannot distinguish between transactions generated by the CPU and those generated by another bus master.

DMA controllers (or any other types of bus masters) are able to read and write the Z-SCAN mappable memory once they have gained control of the bus during an emulation. A prerequisite for such operations is that the alternative bus master and the CPU socket must be connected to the same bus. Any buffering between the socket and the alternative bus master probably prevents access to mappable memory because the buffers may be 3-stated during DMA operations.

Z-SCAN does not acknowledge bus requests from the target when no emulation is in progress. This means that DMA devices that repeatedly request the bus once it is enabled (for example, CRT refresh controllers) do not have their requests honored after an emulation has terminated. For problem-free operation with Z-SCAN, such controllers should be designed to accommodate the possibility that the bus may not be granted an indefinite period following a request. Designs that abort or shut down if a request is not honored within a certain time may not restart correctly when emuation is resumed.

5.4.8 Termination of Emulation

The previous section mentioned that a breakpoint, whether manual or programmed, is not activated until the end of a bus acknowledge state. The same is true of wait and stop states: an emulation cannot terminate unless RESET-, BUSACK-, WAIT-, and STOP- are all inactive. The only certain method of returning control to the Z-SCAN monitor if any of these signals is stuck in the active state is to apply a monitor reset using the Z-SCAN front panel switches. This action, which must be followed by RETURN, reinitializes the monitor, meaning that Z-SCAN must be completely reprogrammed. This can be avoided by first resetting the target with its own reset logic (a Target mode reset from Z-SCAN front panel switches may not have the required effect; see Section 4.3) and then entering a manual break with a Monitor mode non-maskable interrupt (NMI).

There are occasions when it appears that a programmed breakpoint should have been encountered--execution has proceeded past the point at which the breakpoint was expected to occur--but emulation continues. This is likely to happen when the bus transaction on which the breakpoint was set does not completely match the conditions set up with the Breakpoint command. The mismatch can arise from the fact that the Z-SCAN address/data bus breakpoint comparator is a full 16 bits wide and so does not detect byte data or interrupt vector matches (eight bits) or refresh address matches (nine bits). Problems can often be circumvented either by eliminating the offending term from the break condition or by choosing an alternative bus transaction as the trigger condition. This topic is discussed in more detail in Section 6.10.1.

Unless one or more of the signals mentioned in the first paragraph is active, it should always be possible to stop an emulation with a front panel monitor NMI. If emulation cannot be stopped this way, it is possible that the CPU has entered an illegal state as a result of being driven by target-generated signals that are out of specification. The most likely culprit is the clock. Another possibility is that substantial ground currents are flowing in the emulator cable and disrupting dc levels. This can happen when Z-SCAN and the target system are connected to different power ground distributions. For this reason it is recommended that Z-SCAN and the target system are both connected to the same power receptacle.

5.5 USE OF THE HARDWARE TRIGGER

The rear panel BNC connector which carries the Z-SCAN break pulse output signal allows the unit to be used in conjunction with other test instruments such as oscilloscopes and logic analyzers. Section 6.10.1 details the programming of the break pulse logic. This section discusses the use of the break pulse in general terms only because of the wide variety of equipment with which Z-SCAN can be used.

5.5.1 Break Pulse Characteristics

The pulse from the rear panel is positive going. This means that the rising edge of the pulse signals the time when the programmed bus condition is detected. Detection occurs shortly after the rising edge of clock in T2, and the pulse is one clock cycle long. The pulse appears in the same cycle that causes the match when the programmed condition is address, control/status, or control/status with address. For data or control/status with data, the pulse is output during the bus cycle that follows the programmed condition causing the match. A separate pulse is produced for each cycle which results in a match, even when a number of consecutive cycles all match the programmed condition.

The pass counter logic can be used in conjunction with the pulse feature. For a pass count of n, a pulse is output every nth time the programmed condition is satisfied. There is no output at any other time.

The Z-SCAN pulse logic is inhibited when no emulation is in progress, so there is no danger of spurious triggering of external equipment, even if the programmed match condition is satisfied during the execution of the monitor software.

5.5.2 Connection of External Equipment

Connection of an oscilloscope or logic analyzer to the Z-SCAN break pulse output is a simple matter, typically accomplished with a coaxial cable terminated with BNC connectors.

--CAUTION--

The break pulse output is driven directly by the output of a low-power Schottky TTL gate, which may be damaged if subjected to a sustained short circuit.

Due to its short duration, the pulse should be used as an external trigger when Z-SCAN is used in conjunction with a logic analyzer. It is not suitable for use as a gating, qualifying, or enabling signal. The analyzer should be set to expect a positive-going TTL-level trigger pulse.

The pulse can also be used as an external trigger for an oscilloscope. The time base should be set to trigger on the positive edge of the dc-coupled input signal. AC coupling is not recommended because the mean level of the signal--and hence the trigger point--changes according to the frequency of the pulses. If the break pulse is displayed on an oscilloscope for any reason, it may have a slight but inconsequential ringing on transitions, because the output is not terminated.

5.5.3 Choice of Logic Analyser Recording Window

When a logic analyzer receives a trigger, one of three things happens:

- **The logic analyzer stops recording,** and its memory holds information about events prior to the trigger.

- **It starts recording,** stopping when its memory is full. Here, events after the trigger are recorded.

- **It continues recording** but stops after a certain number of sample clock cycles. Information about events before and after the trigger is recorded.

All three options can be used in conjunction with the Z-SCAN break pulse output when the pulse_only option is selected with the Break command (see Section 6.11.1).

Z-SCAN also outputs a break pulse when the pulse_&_break option is selected. This allows a logic analyzer to be used to record events in the target system prior to the break if the analyzer is set to stop recording when it receives a trigger pulse.

5.5.4 Clocking of Logic Analyzers

With most logic analyzers, users have a choice of two sources for the clock that determines the rate at which logic inputs are sampled. The analyzer's internal clock, which typically allows sampling rates of between tens of nanoseconds and milliseconds, can be used. Alternatively, the user can provide an external clock signal from the system under test.

Both sources have their uses in the development of Z8000-based designs. The internal clock is used both for simplicity of set-up and for maximum resolution. The sample rate should be such that the sample period is shorter than the duration of the shortest pulse being monitored; this avoids the possibility of pulses being missed. The shortest pulse generated by the Z8000 is Address Strobe (AS-), which has a minimum duration of 20 ns less than the CPU's clock high width (85 ns with a 4 MHz clock). In order to reliably capture AS-, a sample period of 50 ns is required in a 4 MHz system with an internally clocked logic analyzer. Still shorter periods may be required to capture glitches of short duration in the target hardware.

The disadvantage of internal clocking is that the analyzer must sample the input signals at a higher rate than is strictly necessary to record events occurring in a system that is driven synchronously by its own clock source. As a result, the number of events that can be recorded by the analyzer before its memory becomes saturated is less than optimal. For example, with a 50 ns sample period it takes only about 50 Z8000 bus cycles to saturate a 1024-word sample memory. A greater number of cycles could be accommodated if a clocking signal precisely matched to the Z8000 system were used. Of course, such a signal is readily available in the CPU clock signal.

When the CPU clock is used as an external clock for a logic analyzer, its negative edge (high-to-low transition) should be treated as active. It is on this edge that Address Strobe is active, that address and status are valid in T1 of any bus cycle, that WAIT- is sampled during T2, and that data is sampled in T3 during read operations. For further details, see Section 9.4 of the Z8000 CPU Technical Manual. Using the clock, a 1024-word sample memory in an analyzer can record about 250 Z8000 bus cycles, five times more than is possible with the analyzer's own clock.

In certain situations it may be desirable to use a more selective clock source than the clock signal. For example, the user may wish to record only I/O transactions or only refresh cycles. This can be accomplished in one of two ways, depending on the amount of information required about the transactions:

- **Address and Status Only:** When just one sample occurring shortly after Address Strobe is sufficient, the break pulse output of Z-SCAN can be used directly as a positive-going clock signal. Use the Break command (see Section 6.11.1) to select pulse_only on the selected status condition, with ignore_AD and a count of 01.

- **Full Record:** When complete information about each transaction is required, the analyzer should be clocked either from its internal clock source or from the target clock, as described above, but the clock must be qualified by the Z8000 status signals in order to select particular types of transactions. Most analyzers have a clock qualifier input that can be used to gate the clock in this way. The qualifying signal must come from the target, from a proprietary qualifying probe pod available from the supplier of the analyzer, or from a temporary circuit constructed from logic components and attached to the target system. Most Z8000 systems fully decode the status lines ST_{0-3}. An output of the decoder can often be used directly as a clock qualifier, avoiding the requirement for additional logic.

5.5.5 Break Pulse Demonstration

The wide variety of equipment that can be used with Z-SCAN makes it impossible to give a step-by-step tutorial. It is suggested that users having logic analyzers and wishing to become familiar with the break pulse should procede as follows:

1. Connect the emulator cable to the Z-SCAN as described in Section 5.2.2, steps 1 through 4.

2. Attach logic analyzer signal probes to signal pins of interest on the header at the target end of the cable. Suggested signals are ST_{0-3} (pins 21-18 on the Z8002, 23-20 on the Z8001), AS- (pins 29-34), DS- (pins 17-19), MREQ- (pins 16-18), and R/W- (pins 25-30). Consult the logic analyzer manual to find out how to arrange these so ST_{0-3} can be interpreted as a hex digit with ST_3 as the most significant bit.

--WARNING--

Take care not to bend or break the pins on the header.

3. Connect the Z-SCAN rear panel break pulse output to the external trigger input of the analyzer. On analyzers without such an input, connect the break pulse to an unused signal input. The position of the connector is shown in Figure 3-2.

4. Ensure that the analyzer is on, then turn on the Z-SCAN.

5. Set up the analyzer for pre-trigger recording with an internal clock period of 50 ns. Select a positive-going trigger from the external trigger input rather than from the analyzer's internal word recognizer. For analyzers without an external trigger, select the signal input

carrying the break pulse as the only significant term in the trigger equation: all other inputs should be "don't cares." Consult the logic analyzer manual for its set-up procedure.

6. Prime the analyzer manually so that it starts to record information.

7. Work through the tutorial of Section 4 of this manual. Experienced users can ignore some of the redundant keystrokes in arriving at the steps that perform emulations.

Each time an emulation terminating with a trigger break is run, the analyzer should capture information about the transactions preceding the break. If it does not, it is probably not set up or primed properly. Check the manual again. Once the error has been corrected, use the Z-SCAN Register command (see Section 6.10.4) to reset the PC register to the value it held before emulation, then rerun the emulation with the Go command (Section 6.11.1). Before each emulation, reprime the analyzer so that it can record the new information.

The format in which the recorded information is displayed depends both on the particular analyzer used and on the display mode selected. The signals suggested above in step 2 are best suited to the Timing Diagram Display mode supported by most analyzers. These signals give insight into the way the processor uses the bus while executing programs.

SECTION SIX

MONITOR SOFTWARE DESCRIPTION

6.1 INTRODUCTION

The main functions of the monitor program software are to monitor the interaction between the Z-SCAN system and the target system during emulation, to supervise the changeover from Target mode to Monitor mode, and to control the passing of program files between a host system and Z-SCAN. A secondary but more visible aspect of the program is its user interface, which controls, monitors, and acts upon user input from the terminal keyboard. Section 6.2 describes the Z-SCAN operating modes in more detail.

The user interface controls the commands available to the user throughout the set-up and execution phases of emulation. It also checks the syntax and sequence of user input and acts upon correct sequences by updating the screen display and, if necessary, the values of operating parameter fields. The Z-SCAN screen displays are introduced in Section 6.3. If user input is invalid, the monitor ignores it completely. Valid inputs are discussed in Sections 6.4 through 6.7.

Table 6-1 is an alphabetical summary of the software commands available in the monitor program and the keys that must be entered to access them. Sections 6.8 through 6.12 describe each command, its parameters, and its usage in detail.

Table 6-1. Software Monitor Commands

Command	Key	Command	Key	Command	Key
Break	B	Inst_count	I	reAd	A
Compare	C	Load	L	reGister	G
Display	D	mAp	A	Resources	R
eXamine	X	Memory_io	M	seNd	N
Execution	E	moVe	V	System	S
Fill	F	Next	N	Trace	T
Go	G	Peek	P	Wait_states	W
Host	H	Quit	Q	Write	W

6.2 Z-SCAN 8000 OPERATING MODES

The Z-SCAN 8000 system can operate in any of three modes: Monitor mode, Transparent mode and the Target Mode. One of these can be used only in configurations that include a host system for software development. The three modes are:

1. **Monitor Mode**

- The Z-SCAN terminal is logically connected to the Z-SCAN unit, giving the user access to the monitor's commands.
- In this mode, the user has access to five distinct displays on the terminal's CRT. These screens allow the examination, enabling, and modification of resources belonging to the Z-SCAN or to the target system.
- If the Z-SCAN system configuration includes a host system, program files can be passed between the host file system and Z-SCAN's memory when the Z-SCAN Monitor mode is in effect.

2. **Transparent Mode**

- This mode logically connects the Z-SCAN terminal to the host system, giving the user unrestricted access to host system resources.
- The user has no access to the Z-SCAN monitor commands during Transparent mode, but can enter host system commands just as if the terminal were directly connected to the host.
- Transparent mode can be entered from Monitor mode at any time and does not affect any parameters the user has set up to control the debugging process.

3. **Target Mode**

- This mode dedicates Z-SCAN resources to running either Z8001 or Z8002 emulation on the target system, depending on the CPU installed in Z-SCAN.
- While Target mode is in effect, the user cannot enter keyboard commands either for Z-SCAN or for the host system.

Following a RESET and baud rate synchronization (see Section 3.5), the Z-SCAN software enters Monitor mode. Transition to either Transparent mode or Target mode occurs in response to specific commands entered on the terminal keyboard. The reverse transitions take place in response to generation of the Z-SCAN break signals. Alternatively, transition from Target mode to Monitor mode can take place if the program running during emulation generates a condition that triggers the Z-SCAN breakpoint logic.

Figure 6-1 diagrams the allowed transitions and their causes. Note that direct transitions between Transparent and Target modes are not allowed.

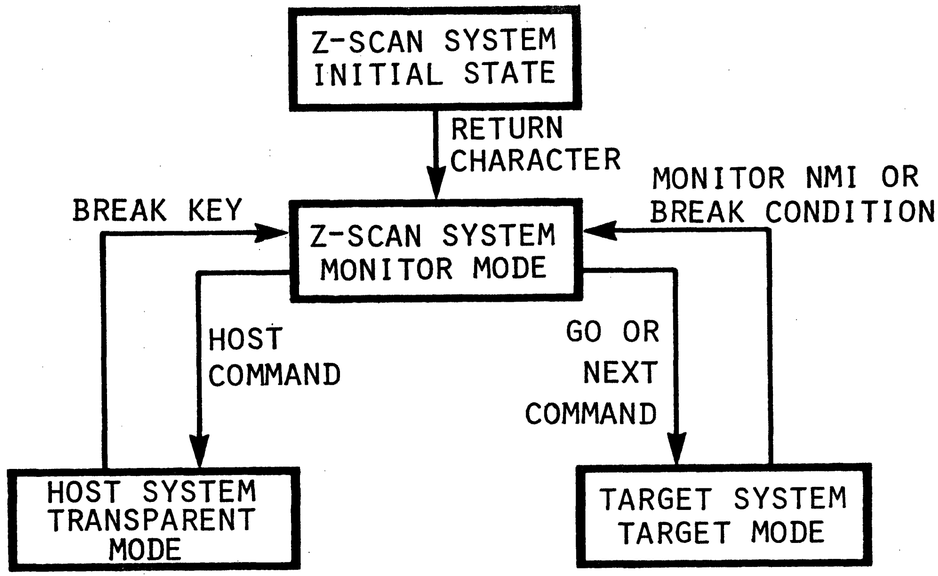


Figure 6-1. Z-SCAN 8000 Operating Modes

6.3 MONITOR MODE OVERVIEW

Almost all interaction between the user and Z-SCAN monitor software occurs in Monitor mode. During Transparent and Target modes, the user cannot give commands to the monitor. For this reason, the remainder of this section describes Monitor mode commands and displays.

The commands available while Z-SCAN is in Transparent mode are determined by the host system and its operating software and are not described here. Refer to the host system's documentation for further details.

While operating in Target mode, Z-SCAN 8000 emulates either the Z8001 or the Z8002, so its behavior is described in the Z8000 CPU Technical Manual (document #00-2010-C). Further information can be found in Section 5.4 of this manual.

In Monitor mode, Z-SCAN gives the user access to five screens, each allowing the user to select from a menu of commands and other screens in order to set up and control the emulation process. The basic functions of each screen are listed below. In keeping with the format used by the Z-SCAN CRT terminal displays, the capital letter in each screen or command name indicates the key entered to activate the screen or command. Sections 6.8 through 6.13 give a complete description of each screen and how its associated commands can be used.

The seven screens available in Monitor mode and the functions of each are:

1. **Terminal Selection screen**

   o Displayed following monitor RESET and entry of RETURN.

   o Allows the user to select from a choice of terminal cursor control protocols.

- Provides access to the System screen.

2. **System screen**

- Informs the user of the monitor software release level and of the baud rate selected for communications between the terminal, Z-SCAN's monitor and the target system.

- Allows the user to select the appropriate baud rate for the host system communication link.

- Allows the user to select one of two values, internal operation or refresh, for the CPU status that is sent to the target system while the monitor software is running.

- Allows other screens to be selected.

3. **Memory_io screen**

- Presents a choice of commands that allow manipulation of target system memory: Compare, Display, eXamine, Fill and moVe.

- Allows target system input and output ports to be manipulated with the reAd and Write commands.

- Permits program files to be loaded from the host system into memory. This function is controlled by the Load command.

- Permits program files or information to be sent to the host system from Z-SCAN via the seNd command.

- Allows other screens to be selected.

4. **Resources screen**

- Gives the user access to the Z-SCAN emulation control resources through the Break, Inst_count, mAp, reGister, Peek and Wait_states commands.

- Allows other screens to be selected.

5. **Execution screen**

- Displays the conditions selected for the execution and control of emulations and the state of the processor before and after each emulation.

- Allows emulations to be started with the Go and Next commands.

- Upon termination of each emulation begun by either the Go or Next commands, displays a message stating the reason for termination.

- Allows access to the Trace screen.

- Allows other screens to be selected.

6. **Trace screen**

- Accessible from Execution screen.

- Provides an instruction-by-instruction analysis of the execution of the user program and its effect on registers and memory.

- Allows return to the Execution screen.

7. **Host screen**

- Displayed when Z-SCAN enters the Transparent mode.

- Indicates that control of the display has passed from the Z-SCAN monitor to the host system's software.

- Does not allow access to Z-SCAN monitor commands or to other screens.

6.4 Z-SCAN SCREEN LAYOUTS AND COMMAND DISPLAYS

With the exception of the Terminal Selection screen and the Host screen, each of the Z-SCAN displays is divided into two or more areas by horizontal rows of dashes. Each area has one of three distinct functions:

- **Menu area.** This area appears below the bottom row of dashes on the System, Memory_io, Resources and Execution screens. It lists the name of the screen, the commands available on the screen, and the active command. If no command is active, the names of alternative screens are listed.

- **Window area.** This type of area appears on the Memory_io screen only, directly above the menu area. It displays variable amounts of information during the execution of commands available on the Memory_io screen.

- **Command area.** Any area above both the menu area and (on the Memory_io screen only) the window area is a command area. These areas display fixed amounts of information. Much of the information displayed in command areas is invariant, consisting of headings or of Z-SCAN parameters that cannot be changed on the particular screen displayed. The contents of certain fields in command areas can be altered by the user. This process is described in Sections 6.5, Cursor Manipulation, and Section 6.6, Variable Fields.

  Command areas are divided into subscreens. Each subscreen is controlled by one of the Z-SCAN monitor commands. On the Memory_io screen, only the subscreen associated with the active command is displayed, whereas on other screens, all subscreens are displayed at all times.

6.4.1 The Menu Area

The menu area appears on all screens except the Terminal Selection screen and the Host screen. The format and content of other areas varies from screen to screen and is discussed in following sections that relate to specific screens.

Examples of menu displays can be found at the bottoms of Figures 4-5 and 4-9. Moving through the two lines of the menu area from left to right and top to bottom, the first items encountered are enclosed in parentheses. The item on the far left is the name of the screen itself--for example "Resources screen." If there is another item inside the parentheses, it is the name of an active command, which is a command the user has selected from those displayed on the lower line of the menu.

Outside the parentheses, the contents of the upper line vary according to whether or not a command has been selected. If no command is active, the line names alternative screens. To select another screen the user enters the initial letter of the desired screen name. When a command is active, the only information outside the parentheses is the name of any command supplementary to the active command. To execute a supplementary command, the user enters the letter in its name that is capitalized. The only supplementary command is Quit, which deactivates the current command.

The contents of the lower line of the menu area are fixed for each particular screen; they do not change in response to user input. The lower line lists the commands available on a particular screen. The user can activate any one of these commands by entering the letter in its name that is capitalized, for example, "A" for the mAp command.

6.5 CURSOR MANIPULATION

When a screen is first displayed, the cursor rests on the initial letter of the name of the screen, which is inside the parentheses at the top left of the menu area. When a command is activated, its name is added to the information inside the parentheses, and the cursor automatically moves to the first variable field associated with the active command. For commands controlling more than one field, the "first" field is that nearest the top left of the subscreen associated with the command.

Once the cursor has been positioned on the first variable field in a subscreen, the contents of that field can be altered by user input. This process is described in Section 6.6. To allow other fields on the same subscreen to be altered, the cursor must be moved into those fields. The four cursor control keys, ↑ , ↓ , <--, and --> (cursor up, cursor down, cursor left, and cursor right) are used for this purpose.

Each cursor control key moves the cursor into a variable field that is logically adjacent to the current field. Logical adjacency requires the destination field to be part of the subscreen associated with the active command. A field that is physically adjacent (for example, on the same display line as the current field) but is not part of the subscreen associated with the active command cannot be entered through the use of the cursor control keys alone.

For consistency, the top and bottom variable fields in a given subscreen column are considered to be logically adjacent, as are the far left and far right fields on a given subscreen line. This means that if, for example, the cursor is moved up from the top field, it appears on the bottom field. Sometimes there is only one variable field in a particular subscreen row. In such cases, the cursor left and cursor right keys cannot move the cursor out of the field, because there is no field logically adjacent to it in those directions (in fact, the implementation considers the field to be logically adjacent to itself). Similar reasoning applies to fields without a logically adjacent field above or below them.

After the values held by the fields in a given subscreen are updated, the cursor must be returned to the menu area before a new command be activated or an alternative screen selected. Entering RETURN achieves this, moving the cursor from the current field to the name of the active command. The command itself is not deactivated until a new command is selected or the Quit command is executed.

Table 6-2 shows how the cursor can be manipulated to access parameter fields in a typical subscreen. The example shows the effect of the cursor movement keys when the Resources screen is active. The cursor position in each step is at the left of the field shown in boldface type. Refer to Section 6.10.5 for further details of the Peek command.

Table 6-2. Effect of Cursor Control Keys
(Peek command, Resources screen)

Step	Display	Keystroke	Notes
1.	SC 00 0000 SC 00 0000 SC 00 0000		Cursor is on screen name (Resources screen) prior to selecting Peek command.
		P	The Peek command is activated by keying P. This moves the cursor to the first variable item in the Peek command area. (Cursor position is indicated by bold characters of the following display.)
2.	**SC** 00 0000 SC 00 0000 SC 00 0000	-->	Move right. This key positions the cursor at the beginning of the second variable item in this command area.
3.	SC **00** 0000 SC 00 0000 SC 00 0000	↓	Move down to vertically adjacent field.
4.	SC 00 0000 SC **00** 0000 SC 00 0000	<--	Move left.
5.	SC 00 0000 **SC** 00 0000 SC 00 0000	↑	Move up to vertically adjacent field.
6.	**SC** 00 0000 SC 00 0000 SC 00 0000	↑	Move directly down to bottom row of this field.
7.	SC 00 0000 SC 00 0000 **SC** 00 0000	RETURN	Remove cursor from subscreen.

6.6 VARIABLE FIELDS

As mentioned above, command areas display information the user is allowed to alter. There are two main types of modifiable fields: hexadecimal and multiple choice. The next two subsections discuss these in detail. A third subsection mentions two additional types of fields, the file name and memory content fields.

6.6.1 Hexadecimal Fields

A hexadecimal field may contain two, four or 16 hexadecimal digits and can be modified, once the cursor has been positioned in the field, by entering new digits (0-9 and upper case A-F). A space restores the default value of a hex field whereas entering the control (CTRL) and R keys returns it to the value it held when the cursor was last moved into it. CTRL R is entered by holding down the control key while pressing the R key.

If the user does not want to modify the particular digit on which the cursor rests, "<" or ">" can be entered to move the cursor left or right within the field. The SHIFT key must be held down while these characters (< or >) are entered.

None of the keys mentioned above is able to move the cursor out of the variable field. When the cursor is moved to the right of the far-right character, it wraps around to the far-left character. The reverse is also true. The cursor can be moved out of the field by using the cursor control functions described in the previous section.

Table 6-3 gives examples of the effects of valid keys on a four-digit hexadecimal field. Although the count field in the Compare command (Memory_io screen) has been used for this example, the effects of keystrokes as shown in Table 6-3 apply to all similar hexadecimal fields. The position of the cursor is shown in Table 6-3 by a vertical arrow.

Step 7 of the key sequence shown in the table gives the result shown only if the field holds the value 0018 when the cursor is moved into it. If you wish to examine the effect of the sequence yourself, you must first set up the field by selecting the Memory_io_screen, then enter the following keystrokes:

C, left, 0, 0, 1, 8, left, right

Table 6-3. Effect of User Entry on Hexadecimal Field (Compare command, Memory_io screen)

Step	Keystroke	Contents	Notes
1.	(see note)	0018	Not default value. Use instructions in previous paragraph to start with 0018 in the count field and with the cursor on the first 0.
2.	1	1018	Changes digit that cursor is on to 1.

Table 6-3. Effect of User Entry on Hexadecimal Field--Continued
(Compare command, Memory_io screen)

Step	Keystroke	Contents	Notes
3.	SHIFT >	1018	Moves cursor right one position. (Repeated use of the SHIFT and > keys steps the cursor forward through each position. This is useful in moving the cursor to a particular position without having to rekey any of the other digits.)
4.	G	1018	Z-SCAN ignores this input because it is not a valid hex digit.
5.	F	10F8	Hex digit "F" is inserted and the cursor automatically moves to the next position.
6.	SHIFT >	10F8	These keys are used to move forward through each position in the field. If the cursor is on the last position of a hexadecimal field, SHIFT > moves the cursor back to the first position of the field.
7.	space	000C	Entering a space always restores the field to the default value. Also, the cursor returns to the first position of the field each time a space is used.
8.	SHIFT <	000C	These keys (SHIFT and <) move the cursor backwards through each position of the field. If the cursor was on the first position before using these keys, then the use of these keys moves the cursor to the last position of this field.
9.	CTRL R	0018	CTRL R is used to restore the contents of a field to the value it held when the cursor firts moved into the field.

All hexadecimal fields are initialized to meaningful default values by a monitor reset or power-up sequence. The range of values that can be held in some hexadecimal fields is constrained: for example, a count field cannot contain zero. If the user attempts to move the cursor out of a field that contains an illegal value, the monitor automatically restores the contents of the field to what it held when the cursor last entered it.

6.6.2 Multiple-Choice Fields

The other main type of modifiable field is the multiple-choice field. As the name suggests, such fields allow the user to choose a value from a fixed number of alternatives.

An example of such a field is the data type associated with the Display command (see Section 6.10.2). It has five valid values: word, byte, long (for 32-bit long words), nseg (for nonsegmented disassembly) and seg (for segmented disassembly). Rather than displaying the value selected as a single-digit code, the monitor displays a descriptive string. Single digit codes are used internally by Z-SCAN as indexes into tables of possible values for each multiple-choice field. Consequently, when the cursor is positioned on a multiple-choice field, the user can select a particular value (table entry) by entering the single hexadecimal digit that indexes that choice. Valid indexes range from zero to one less than the number of entries in the table. If the user inputs a digit outside the allowed range, the last entry in the table is used.

It is unreasonable to expect all users to learn the index numbers for each possible value in each of the Z-SCAN multiple choice fields. Consequently, an alternative method of selecting values is provided: when the cursor rests on a multiple choice field, SHIFT and > can be entered to select and display the next possible value. The preceding table entry is selected by entering SHIFT and <. As previously indicated, the SHIFT key must be depressed during the entry of either of these characters (< or >). This feature allows the user to step forward or backward through the list of available values until the desired value is found. Stepping forward from the last available value reselects the first possible value. The reverse is also true.

Besides the hexadecimal digits, the "greater than" and the "less than" keys, two other keys are accepted as valid inputs for multiple choice fields. The space character is considered equivalent to zero, and so it selects the first of the possible choices. Entering the CTRL and R keys returns the contents of a multiple choice field to what it held when the cursor last entered it.

Like hexadecimal fields, multiple choice fields are initialized to meaningful default values by a monitor reset or power-up sequence. The default corresponds to the first entry in the table of possible values. This makes it possible to restore the default value with a single keystroke: zero or space, either of which selects the first value in the table.

Table 6-4 lists the hexadecimal digits that can be used to make particular choices in one of the multiple choice fields; this particular field is the "type" field of the Memory_io Screen's Display command.

Table 6-4. Multiple Choice Field Indexes (Display command, Memory_io screen)

Index	Choice	Selected by
0	word	0, space, Monitor reset
1	byte	1
2	long	2
3	nseg	3
4	seg	4, or any of the hex digits greater than 4 (i.e., 5 thru F)

Entering an F always selects the last possible choice in any of the multiple choice fields, whereas entering a 0 or space always selects the first possible choice in any multiple-choice field.

Table 6-5 shows the effect of user input including some key sequences other than Index entries on the same example field. These examples show how user input is handled on all multiple choice fields by the Z-SCAN monitor. If you wish to perform the sequential steps in this table, select the Memory_io screen, then enter the following keystrokes:

D, left, 4, left, right

Table 6-5. Effect of User Input on Multiple Choice Field (Display command, Memory_io screen)

Step	Keystroke	Contents	Index	Notes
1.	(none)	seg	4	The field holds this value when the cursor is moved into it. Step 6 recalls the same value.
2.	1	byte	1	Select second table entry by the index value of 1.
3.	SHIFT <	word	0	Step back to preceeding table entry via SHIFT and < keys.
4.	SHIFT >	byte	1	Step forward to next table entry via SHIFT and > key.
5.	2	long	2	Move directly to third entry of table using index value.

Table 6-5. Effect of User Input on Multiple Choice Field
(Display command, Memory_io screen)

Step	Keystroke	Contents	Index	Notes
6.	CTRL R	seg	4	Restore field to value that existed at the time the cursor was moved into the field. This is the value shown in step 1 above.
7.	F	seg	4	Move directly to last entry in table.
8.	SHIFT >	word	0	Step forward and loop back to first entry in table.
9.	6	seg	4	Any hex number larger than number of items in table always selects the last table entry.
10.	3	nseg	3	Selects fourth entry in table.
11.	space	word	0	Restores field to the default value.

6.6.3 Other Field Types - File Name and Memory Content

In addition to hexadecimal and multiple choice fields, the Z-SCAN monitor uses two other types of fields. Each is used in only one situation.

- **The file name field** appears only in the Load and seNd command subscreens (see Sections 6.10.8 and 6.10.9). This field can hold up to 32 non-space characters. Z-SCAN makes no check on input because it has no knowledge of the file-naming conventions of the host system. It is the responsibility of the host system's LOAD command to check the file name for validity. For further details see Section 7, Interface to Non-Zilog Hosts.

  The SHIFT >, SHIFT <, and control (CTRL) and R keys are used in the same way in the file name field as in a hexadecimal field (see Section 6.6.1). Hence, Z-SCAN does not allow these characters to be sent to the host as part of a file name. If you put a space in a file name, Z-SCAN treats it as a terminator and shows this by erasing all input to the right of the inserted space.

- The Memory Content field is used exclusively by the eXamine command and is tailored to the special requirements of that command. Its behavior is similar to that of a hexadecimal field with two, four or eight digits. The differences are summarized in paragraph seven of the next section.

6.7 SUMMARY OF VALID USER INPUT SEQUENCES

The three preceding sections discussed the keystrokes used to achieve the following objectives:

- o Moving from one screen display to another.
- o Activating commands available on the current screen.
- o Moving the cursor into and out of modifiable fields in parameter areas.
- o Updating the contents of modifiable fields.

This section summarizes the keys that can be entered, which generally depend on the position of the cursor on the screen. Refer to Section 6.8 for complete information on executing commands.

Many of the terminals supported by the Z-SCAN monitor offer features which are not needed by the monitor. In many cases, Z-SCAN ignores data received when keys associated with such features are entered. Some keys, however, may send data that is considered to be valid. The function keys on some terminals do this, while others affect the display on the terminal screen without sending data to the Z-SCAN. Local editing keys such as line delete and character insert have this effect.

If the display is corrupted by the entry of an incorrect key, the user should activate another screen (the following paragraphs explain how to do this), then reactivate the corrupted screen. If any of the variable fields on the screen contains an incorrect value, they should be corrected before the user continues.

1. Following a power-up or monitor RESET from front panel:

- o To synchronize Z-SCAN's baud rate with that of your terminal, enter RETURN. The terminal selection screen is displayed. Entry of an incorrect character may result in garbage being displayed. To correct this, RESET Z-SCAN, then enter RETURN.
- o To select a terminal type, enter one of the numbers listed on the screen. Invalid entries are ignored. Incorrect entries can be replaced by entering another selection. Appendix A details the terminals supported by the Z-SCAN monitor.
- o When the terminal type has been selected, enter RETURN to activate the System screen. If the display is corrupted, the terminal selection is incorrect. RESET Z-SCAN and repeat the steps above.

2. Cursor on current screen name inside menu area parentheses:

- o To select another of the five screen displays, enter the appropriate capital letter from the screen name.

- To activate a command on the current screen, enter the appropriate capital letter from the command name. This automatically moves the cursor to the first position of the first item in the associated command area.

- System screen only: Enter RETURN to move cursor into the host baud rate field.

All other input is considered invalid and does not change the state of the monitor software. Screen editing keys must not be used as they can cause unpredictable results.

3. **Cursor on a hexadecimal variable field in a command area:**

- Enter hex digits (0-9, capital A-F) or spaces to modify the value in the field.

- Enter SHIFT > or SHIFT < to move the cursor forward or backward inside the current field without modifying its contents.

- Enter space to change the contents of the field to the default value.

- Enter CTRL R to restore the field to the value it held when the cursor was moved into it.

- Use the cursor control keys (cursor up, cursor down, cursor right or cursor left) to move the cursor to other fields in the subscreen associated with the active command.

- To return the cursor to the menu area (state six below) without executing the active command, enter BREAK.

- To execute the current command, enter RETURN.

Any other key is considered invalid and is ignored by the Z-SCAN monitor software. Screen editing keys must not be used as they can cause unpredictable results.

4. **Cursor on a multiple choice, variable field in a command area:**

- Enter a single hexadecimal digit (0-9, A-F) to select a corresponding value to a particular entry in the Z-SCAN internal table of choices.

- Enter SHIFT > or SHIFT < to select the next or previous value from the Z-SCAN internal table of choices.

- Enter CTRL R to restore the field to the value it held when the cursor was moved into it.

- Enter space to restore the field to the default value.

- Use the cursor control keys (cursor up, cursor down, cursor left or cursor right) to move the cursor to other fields in the subscreen associated with the active command.

- To return the cursor to the menu area (state six below) without executing the active command, enter BREAK.

- To execute the current command, enter RETURN.

Any other key is considered invalid and is ignored by the Z-SCAN monitor software. Screen editing keys must not be used as they can cause unpredictable results.

5. **Cursor on file name field** (Memory_io screen Load and seNd commands only):

- Enter any keys not mentioned below to update the value of the field

- Enter SHIFT > or SHIFT < to move the cursor forward or backward inside the current field without modifying its contents.

- Enter CTRL R to restore the field to the value it held when the cursor was moved into it.

- Enter space to terminate the file name.

- Use the cursor control keys (cursor up or cursor down) to move the cursor to other fields in the subscreen associated with the active command.

- To return the cursor to the menu area without excecuting the command, enter BREAK.

- To execute the command, enter RETURN.

Any other key is considered invalid and is ignored by the Z-SCAN monitor software. Screen editing keys must not be used as they can cause unpredictable results.

6. **Cursor on active command name inside menu area parentheses:**

- To move the cursor to the first variable field associated with the active command, enter RETURN.

- To select another command on the same screen, enter the appropriate capital letter from the command name. The cursor moves automatically from the command name to the first position of the first field associated with the new command.

- To deactivate the current command, enter Q. The upper menu line is rewritten to display the names of alternative screens.

- To display another screen, enter the appropriate capital letter from the screen name. It is not necessary to enter Q prior to displaying another screen when the cursor is on an active command name inside the menu area parentheses. (The Terminal Selection screen cannot be called up in this manner and the Trace screen can only be selected from the Execution screen.)

Any other key is considered invalid and is ignored by the Z-SCAN monitor software. Screen editing keys must not be used as they can cause unpredictable results.

7. **Cursor in window area** (Memory_io screen eXamine command only):

   - Enter hex digits (0-9, A-F) to modify the value in the memory location currently open. After sufficient digits are entered to fill the current location, the next location is automatically opened.

   - Use the <-- (cursor left) key or SHIFT < to backspace over and delete incorrectly entered digits.

   - The and (cursor up and cursor down) keys can be used to open the previous and next locations respectively.

   - To return the cursor to the menu area, enter RETURN.

   Any other key is considered invalid and is completely ignored by the Z-SCAN monitor software. Screen editing keys must not be used as they can cause unpredictable results.

8. **Cursor resting in window area** (Memory_io screen only):

   - Enter cursor down to clear the window area and display the next block of data.

   - Enter cursor up to clear the window and display the previous block of data (Display command only).

   - Enter BREAK to abort the Load, seNd, Fill or moVe commands.

   - Enter RETURN to move the cursor to the menu area.

   Any other key is considered invalid and is completely ignored by the Z-SCAN monitor software. Screen editing keys must not be used as they can cause unpredictable results.

9. **Cursor resting at bottom right of Trace screen:**

   - Enter cursor down to execute the number of instructions shown in the step count field.

   - Enter hex digits (0-9, A-F) to alter the value in the step count field. Use cursor left or SHIFT < to backspace over and delete incorrect entries. Use cursor down to begin executing the number of instructions indicated by the modified step count.

- Enter RETURN to move to the Execution screen.

Any other key is considered invalid and is completely ignored by the Z-SCAN monitor software. Screen editing keys must not be used as they can cause unpredictable results.

10. **Cursor writing data onto screen:**

    - Enter CTRL S to force a temporary pause in data output to the terminal. Enter CTRL Q to resume output. Some terminals have a scroll control key which sends CTRL S and CTRL Q alternately.

    - Enter BREAK to ensure that output stops when the screen is full and before any part of the screen is overwritten with new data. Break achieves this by aborting execution on the Trace screen and by flushing the type ahead input buffer. Any commands entered but not yet executed are lost.

    Any other key (with the exception of screen editing keys) can be entered while Z-SCAN is sending data to the terminal or while output is paused. The input is buffered and is not acted upon until output is complete.

11. **Cursor resting in Execution screen return message area** (Z-SCAN in Target mode):

    - To force a return to Monitor mode, use the Z-SCAN front panel switches to enter a Monitor NMI.

    Input from the keyboard is not accepted while Z-SCAN is in Target mode.

12. **Transparent mode** (all cases not mentioned above):

    - Enter data in the format required by the host system.

    - To return to Monitor mode from Transparent mode, use the BREAK key. If the host and terminal baud rates differ, the user is prompted to set the baud rate of the terminal to the Monitor mode value. The System screen is displayed when the terminal baud rate is correctly set up.

6.8 THE TERMINAL SELECTION SCREEN

Following a power-up or monitor RESET sequence, Z-SCAN must establish two characteristics of the terminal being used:

- Baud rate.

- Cursor addressing protocol.

To set up the baud rate, the user enters a RETURN. Z-SCAN measures the width of the start bit of the character, then programs a baud rate generator to match the calculated speed. Fourteen rates, listed in Table 6-7, are supported.

When the terminal baud rate is established, the Terminal Selection screen shown in Figure 6-2 is displayed. If the terminal is running at a baud rate not supported by Z-SCAN or if an incorrect character is entered, the display will be corrupted or will not appear. Z-SCAN must be RESET before the baud rate can be set up correctly.

The display prompts the user to enter a terminal selection digit. Valid choices are listed on the screen. Invalid entries are ignored. If an incorrect choice is entered, it can be replaced by entering the correct choice.

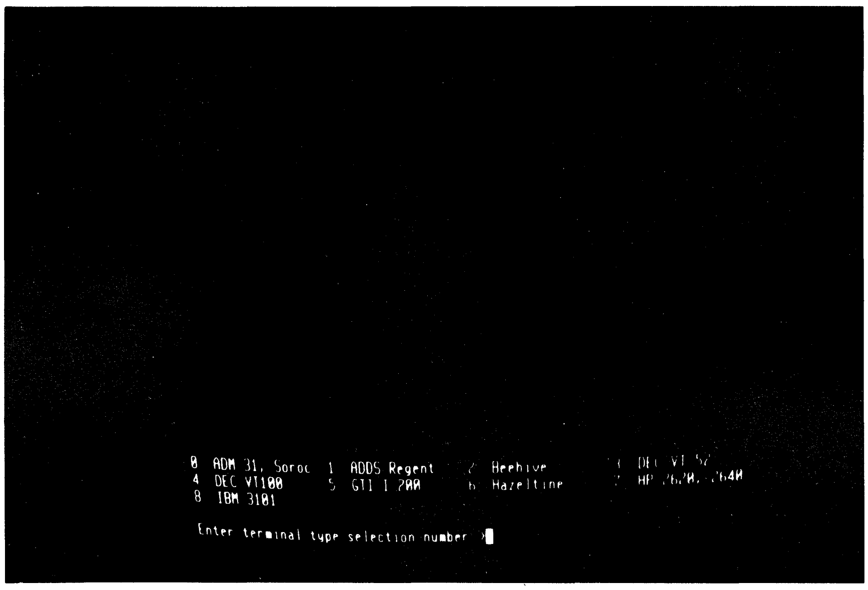


Figure 6-2. The Terminal Selection Screen

Table 6-6 lists a number of terminals supported by the Z-SCAN monitor and the corresponding selection digits. Appendix A gives further information about the supported terminals and describes how to find out whether an unlisted type can work with Z-SCAN.

After the selection digit has been entered, the user must enter RETURN to move to the System screen (Section 6.9). If garbage appears instead of the System screen, the terminal selection digit is incorrect. Z-SCAN must be RESET and baud rate synchronization re-established before the correct digit can be entered.

Table 6-6. Terminals and Terminal Type Selection Numbers

Supplier	Model	Selection Number
ADDS	Regent 20, 40 or 60	1
Beehive	B-100 or Bee-1	2
DEC	VT52	3
DEC	VT100	4
General terminals	I-200, I-400	5
Hazeltine	1420 or 1500 series	6
Hewlett Packard	2620 or 2640 series	7
IBM	3101	8
Lear Siegler	ADM 31	0
Soroc	IQ 120 or IQ 135	0
Televideo	TVI 912 or TVI 920	0
Zentec	Zephyr	0

6.9 THE SYSTEM SCREEN

Figure 6-3 shows the System screen in the default state. The display area is divided into three parts by rows of dashes. The bottom is a Menu area (see Section 6.4.1) and the other two are Command areas. Since there is no choice of commands on this Screen, the second line of the menu area is blank.

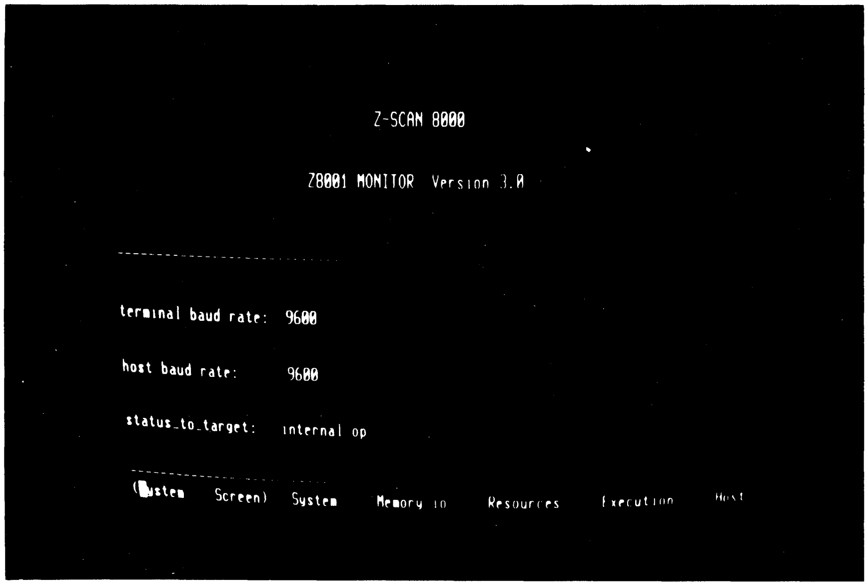


Figure 6-3. The System Screen

The top Command area identifies the Z-SCAN monitor software. It shows the release level of the software and identifies the CPU type (Z8001 or Z8002) installed. It does not contain any user modifiable fields. The release level may differ from that shown in Figure 6-3.

The first field in the center command area specifies the terminal baud rate. This is not a variable field (as discussed in Section 6.6) and cannot be changed by the user on the System screen. The terminal baud rate is used between Z-SCAN and the terminal during Monitor mode.

The second field in the center command area specifies the host baud rate. This variable field allows the user to select the baud rate used between the terminal and a host system (if connected). Following a RESET, its value defaults to the same value as the terminal baud rate.

- o To move the cursor to the host baud rate field from the menu area, enter RETURN.
- o To step forward through the 14 selections available for the host baud rate use SHIFT >.
- o To step backward through the various selections available for this field, use SHIFT <.

Table 6-7 lists the Host baud rates supported by Z-SCAN. Many terminals do not support all the rates supported by Z-SCAN. Some terminals support rates not available on Z-SCAN. These speeds must not be used. The effects of the terminal baud rate selection are discussed in Section 6.13.

Table 6-7. Host Baud Rate Values

Field name	Type	Values (M-C) Range (hex)	Index/ Default	Notes
Host baud rate	mult choice	19200	0	See text
		9600	1	
		4800	2	
		2400	3	
		1800	4	
		1200	5	
		600	6	
		300	7	
		200	8	
		150	9	
		134.5	A	134.5 baud
		110	B	
		75	C	
		50	D	

--NOTE--

In this and subsequent tables, mult choice stands for multiple choice. Hex-N, where N is 2, 4 or 16, indicates a hexadecimal field with the given number of digits. The fourth column lists indexes for multiple-choice fields. The default is always the choice with an index of zero. A default value is given for hexadecimal fields.

The third field in the center command area specifies the status_to_target. This variable field determines the status code sent to the target system during Monitor mode and Transparent mode. It is a multiple-choice field with two possible values: internal op (internal operation, the default value) or refresh. The implications of each possible status are covered in Section 5.4.3.

- To move the cursor from the host baud rate field to the status_to_target field, enter cursor down (↓).
- To move the cursor back to the host baud rate field, enter cursor up (↑).
- To move the cursor to the menu area from either of the two variable fields, enter RETURN.

Table 6-8 summarizes the behavior of the status_to_target field.

Table 6-8. Status_to_target Values

Field Name	Type	Values (m-c) Range (hex)	Index/ Default	Notes
status_to_target	mult choice	internal_op refresh	0 1	see Section 5.4.3

6.10 THE MEMORY_IO SCREEN

The Memory_io screen (Figure 6-4) supports nine commands that can manipulate the contents of memory and I/O ports in the target system. The memory commands can also operate on the Z-SCAN mappable memory (see Section 6.11.3) and on the Z-SCAN monitor memory. Monitor commands and emulations are prohibited from operating on the Z-SCAN I/O ports.

--NOTE--

To set up conditions for emulation, it should not be necessary for the user to operate on the contents of monitor memory with Memory_io screen commands. If the commands are used to operate on monitor memory, the user should exercise great care, since changes to the contents of the memory could prevent the monitor from functioning correctly.

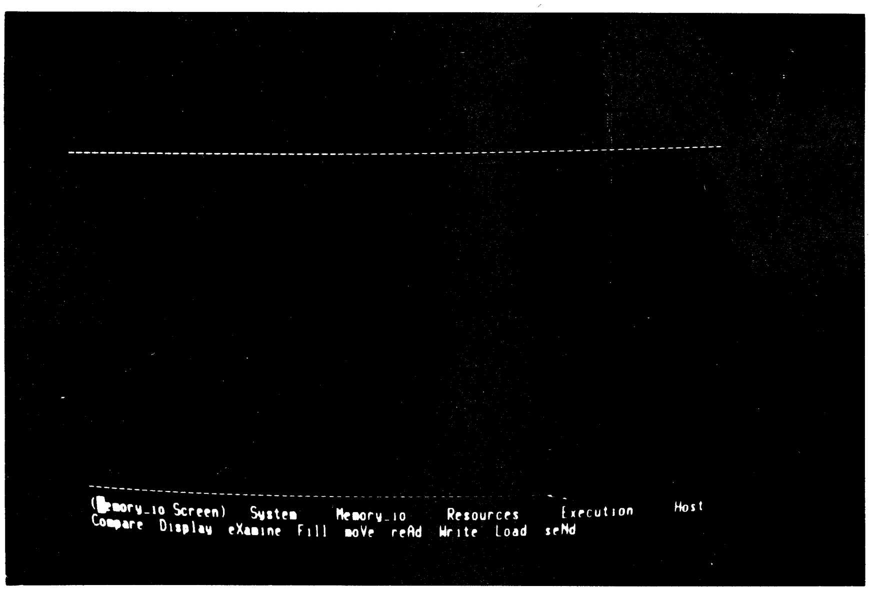


Figure 6-4. The Memory_io Screen

The Memory_io screen display is divided into three areas by lines of dashes. The bottom area is the menu area. Above it is a window area, used primarily for the display of data during the execution of commands. The window is initially empty and remains so until a command is executed. The command area at the top of the screen is also initially blank but is used to display a subscreen associated with the command as soon as any command is activated. Refer to Section 6.4 for further information on screen layouts and command displays.

Once a command has been activated, the cursor automatically moves up into the first variable field in the newly displayed subscreen, allowing the contents of the various fields to be updated as described in Section 6.6.

Simply updating the values held by variable fields does not result in the actual performance of an active command. For example, changing the start address field in the memory display command does not result in the immediate display of the contents of the newly addressed block of memory. In order for the action defined by the command and its parameters to be performed, the command must be executed. Execution is accomplished by entering a RETURN after all the command parameters are set to the desired values. The cursor is in the command parameter area just before execution takes place.

Execution of any command clears the window area (which may contain data resulting from a previous execution) and then performs the command. As execution proceeds, data or messages are displayed in the window. If a command requires the display of more data than will fit in the window, the window is repeatedly filled from top to bottom as many times as are necessary to display all the data. The user is given the opportunity to abort or continue with the command after each block of data is displayed.

Execution of commands that operate on large areas of memory may take a long time because Z-SCAN runs one or two separate emulations for each byte in the block. Section 5.4.4 discusses the hardware implications of these memory accesses.

Execution can be suspended temporarily by entering CTRL S to stop the display of further data. CTRL Q terminates the suspension.

When execution is complete, all commands except Display and eXamine give a closing message, usually DONE. The cursor returns to the menu area, and the command is still active. The state of the software is as described in state seven of Section 6.7.

In summary:

- To abort the current command, enter BREAK. This moves the cursor to the menu area without executing the command.
- To execute the current command, enter RETURN.
- To temporarily stop the execution process, enter CTRL S. To continue, enter CTRL Q.
- To continue when the window is full, enter (cursor down); to abort the execution, enter RETURN.

6.10.1 The Compare Command

This command performs a byte-by-byte comparison of the contents of two areas. Any differences between corresponding bytes in the source and target blocks are reported. The terms source and target have no significance; the result of comparing two blocks of memory does not depend on which is named in the source field and which is in the target.

Execution of the command clears the window area and displays a heading at its top left. Bytes from the source and target areas are then compared. Each time a difference is found the two bytes and their addresses are displayed on a new line. This continues either until the examination is complete or until the window area is full. In the first case, a count of differences is displayed at the bottom left of the window, and the cursor returns to the menu area. When the window is full no message is displayed. The cursor rests at the bottom right of the area. Two user input characters are valid:

- Cursor down clears the window and redisplays the heading, allowing more data to be displayed.

- RETURN terminates the command, moving the cursor to the menu area. The message ABORTED appears, indicating that not all the differences between the blocks were displayed.

If there are no differences between the source and target areas, the headings and the message NO DIFFERENCES appear before the cursor returns to the menu area.

Figure 6-5 is an example of a display produced by the Compare command.

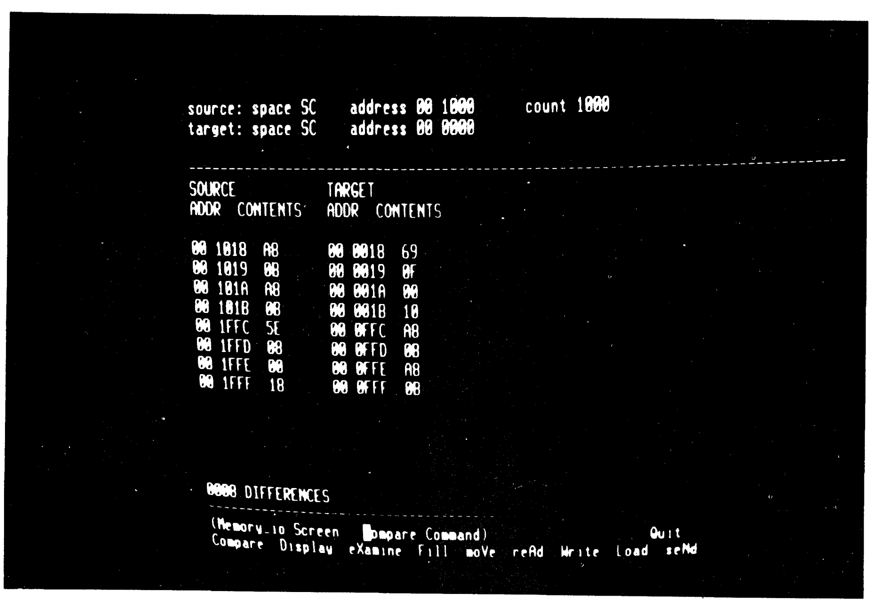


Figure 6-5. The Compare Command

Additional examples of the Compare command are included in the tutorials in Section 4.

Table 6-9 summarizes the Compare command's fields and parameters.

Table 6-9. Compare Command Fields

Field name	Type	Values (M-C) Range (hex)	Index/ Default	Notes
source space	mult choice	SC SD SS NC ND NS MT	0 1 2 3 4 5 6	System code space System data space System stack space Normal code space Normal data space Normal stack space Monitor memory space
source address (segment number) (offset)	Hex-2 Hex-4	00-7F 0000-FFFF	00 0000	Does not appear if Z8002 is installed This is the only address field on the Z8002
count	Hex-4	0001-FFFF	000C	This is a byte count
target space	mult choice	--same as for source space--		
target address (segment number) (offset)	Hex-2 Hex-4	00-7F 0000-FFFF	00 0000	See notes for source address

6.10.2 The Display Command

The Display command allows the contents of blocks of memory to be displayed on the screen. It does not allow the user to modify those contents; that function is handled by the eXamine command, described in Section 6.10.3. The Display command has two operating modes: memory dump and disassembly. In the first, sixteen bytes are displayed per screen line. The hexadecimal representation of their contents follows the address of the first byte, which appears at the left of the screen. The data can be formatted on the display as bytes, words, or long words (two, four or eight hex digits per data item, respectively). The right of the screen is used to display an ASCII representation of the same data, delimited at each end by asterisks (*). Non-printing characters (00-1F and 7E-FF) are represented by periods (.). There are 17 (decimal) display lines giving a total of 110 hex bytes per display.

The Display command can also disassemble the contents of memory in segmented or nonsegmented mode. The format of the disassembled instruction mnemonics and operands is as described in the Z8000 PLZ/ASM Assembly Language Programming Manual (document # 03-3055-02). Addresses and most immediate operands are presented as hexadecimal values. Decimal representation is used for immediate values represented by four or fewer bits.

The display is presented in listing format with addresses at the left of the screen, mnemonics in the center and operands at the right. The memory words disassembled from the decoded instructions appear between the address and the mnemonic.

If the disassembler encounters one of the Z8000's extended instructions, the message * * UNIMPLEMENTED INSTRUCTION * * is displayed in the place of the mnemonic and operands, but the correct number of disassembled words appear to the right of the address. Extended instructions, which may be two, three or four words in length depending on addressing mode and address format, are described in Section 6.8 of the Z8000 CPU Technical Manual (document # 00-2010-C).

When the disassembler finds that the first word of an instruction does not correspond to an operation available on the Z8000, the message * * INVALID OPCODE * * appears and just one word appears to the right of the address. The same message is shown if an illegal register designator--for example, an odd valued long word designator--appears anywhere in the instruction. The number of words disassembled is determined by the opcode and addressing mode of the first instruction word. No error message is generated if invalid constant or opcode fields appear in the second word of an instruction. The invalid instruction message is likely to appear when data areas are disassembled, or when an incorrect disassembly mode (segmented or nonsegmented) is used.

Seventeen instructions are displayed in the window area by disassembly. The number of bytes displayed depends on the instructions disassembled.

--NOTE--

> The Z-SCAN memory target access method allows the Display command to address words at odd memory addresses. Individual Z8000 instructions address words only at even addresses. See Section 5.4.4 for more details.

After 110 (hex) bytes or 17 (decimal) instructions have been decoded, the cursor rests at the bottom right of the window area. One of three characters must be entered:

- **Cursor down** clears the screen, then displays the next block of memory. For word, byte and long display types, the next block starts at the address shown at the bottom left of the display before the screen is cleared, giving an overlap of 16 (decimal) bytes. For disassembly, the next block starts at the address following that of the last word displayed before the screen is cleared.

- **Cursor up** clears the screen, then displays the previous block of memory. For word, byte and long display types, the final line of the block is the same as the first line of data on the screen just erased. Again, this gives an overlap of 16 (decimal) bytes. For disassembly, the first word disassembled is fetched from an address 44 hex bytes below that which follows the last word displayed before the screen is cleared.

- **Return** moves the cursor to the menu area. The command remains active and the window area is not cleared.

Figure 6-6 shows a display produced by the execution of the Display command.

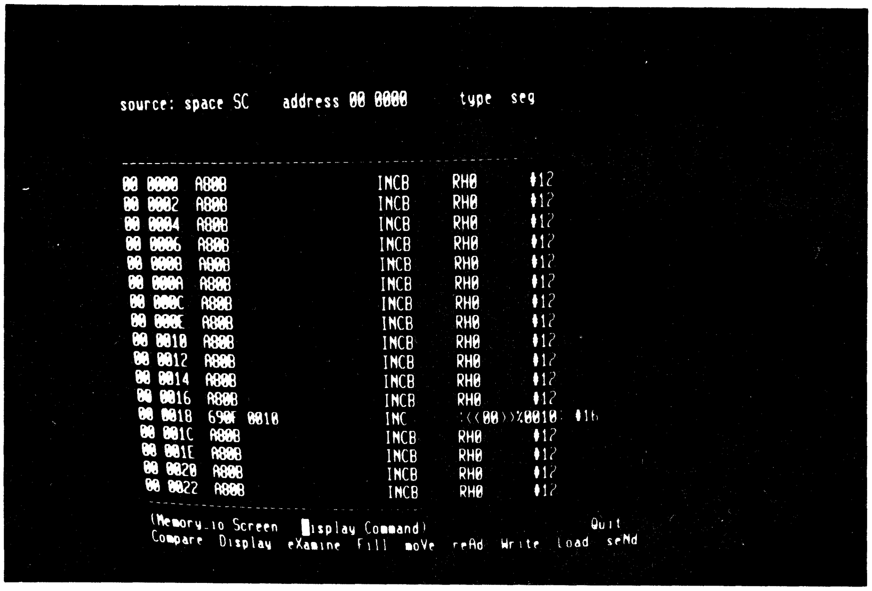


Figure 6-6. The Display Command

Table 6-10 details the variable fields in the Display Command.

Table 6-10. Display Command Fields

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
source space	mult choice	SC SD SS NC ND NS MT	0 1 2 3 4 5 6	System code space System data space System stack space Normal code space Normal data space Normal stack space Monitor memory space
source address (segment number)	Hex-2	00-7F	00	Segment number does not appear if Z8002 is installed
(offset)	Hex-4	0000-FFFF	0000	This is the only address field on the Z8002. It should be even except when the type field is set to byte.
type	mult choice	word byte long nseg seg	0 1 2 3 4	 Non-segmented disassembly Segmented disassembly

6.10.3 The eXamine Command

The function of this command is to allow the user to examine, and optionally to modify, the contents of individual bytes, words, or long words in memory. When the eXamine command is executed, the contents of the location specified in the command area at the top of the screen are displayed in the window area, and the user is prompted for a new value that will replace those contents.

Figure 6-7 shows the initial screen obtained by the eXamine command. Additional examples of this command are included in the tutorials of Section 4.

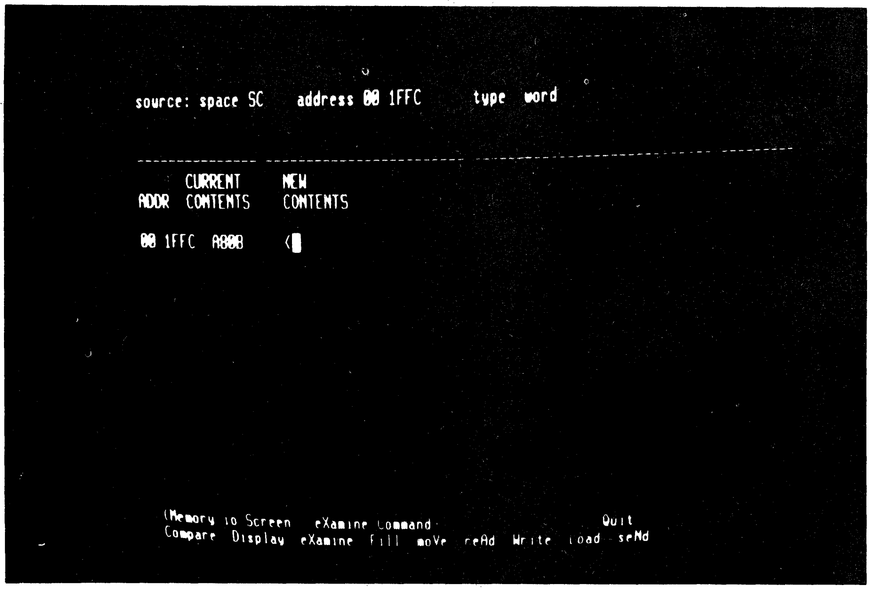


Figure 6-7. The eXamine Command

Table 6-11 lists the parameters of the eXamine command. After setting up the parameters, one of two keys must be entered:

- BREAK returns the cursor to the menu area without executing the command. The command remains active.
- RETURN clears the window area then displays the address of the location selected by the parameters, its contents, and a prompt for a new value.

At this stage, the following keys are valid:

- Hex digits can be used to alter the value held in the open location.
- Cursor up and cursor down keys open the previous and the next memory locations, respectively.
- The next location (in the new contents field) is automatically opened after enough hex digits have been entered to fill the location.
- Erroneous input can be deleted with the cursor left or SHIFT < keys.
- Execution of this command is terminated by entering RETURN. The cursor moves to the menu area and the command remains active.

The eXamine command is capable of writing into the Z-SCAN mappable memory. A write protection feature is available that protects the contents of this memory (see Section 6.11.3). Write protection applies only during Target mode; Memory_io screen commands can still write to the memory. The user should exercise caution to avoid overwriting data that should be preserved.

Table 6-11. eXamine Command Fields

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
source space	mult choice	SC SD SS NC ND NS MT	0 1 2 3 4 5 6	System code space System data space System stack space Normal code space Normal data space Normal stack space Monitor memory space
source address (segment number) (offset)	Hex-2 Hex-4	00-7F 0000-FFFF	00 0000	Segment number does not appear if Z8002 is installed This is the only address field on the Z8002. It should be even except when the type field is set to byte.
type	mult choice	word byte long	0 1 2	

6.10.4 The Fill Command

The Fill command allows the user to replicate a specified string one to eight bytes in length throughout a specified memory area. If the length of the memory area in bytes is not exactly divisible by the length of the string in bytes, the final copy of the string is truncated. The length of the memory area is defined as end_address- (begin_address + 1). This implies that the last byte filled is the one located at the end address. If end_address is less than begin_address, filling continues from the bottom of memory after the top of memory has been passed.

The string is specified as a sequence of up to 16 (decimal) hex digits. The string used to fill memory always consists of a whole number of bytes. Thus, if the user enters a string that has an odd number of digits, a leading zero is assumed. For example, a user input of 12345 is interpreted as a three-byte string: 01, 23, 45.

The default string is empty; that is, it has no digits and its length is zero. It is equivalent to a field of spaces. Z-SCAN automatically aborts the Fill command if it is executed with such a fill string.

The area to be filled may include the Z-SCAN mappable memory, which is described in Section 6.10.3. The Fill command can alter its contents even if it is write protected.

The only display produced by this command in the window area is the termination message DONE. See Figure 4-20 for an example. Table 6-11 lists the command's variable fields.

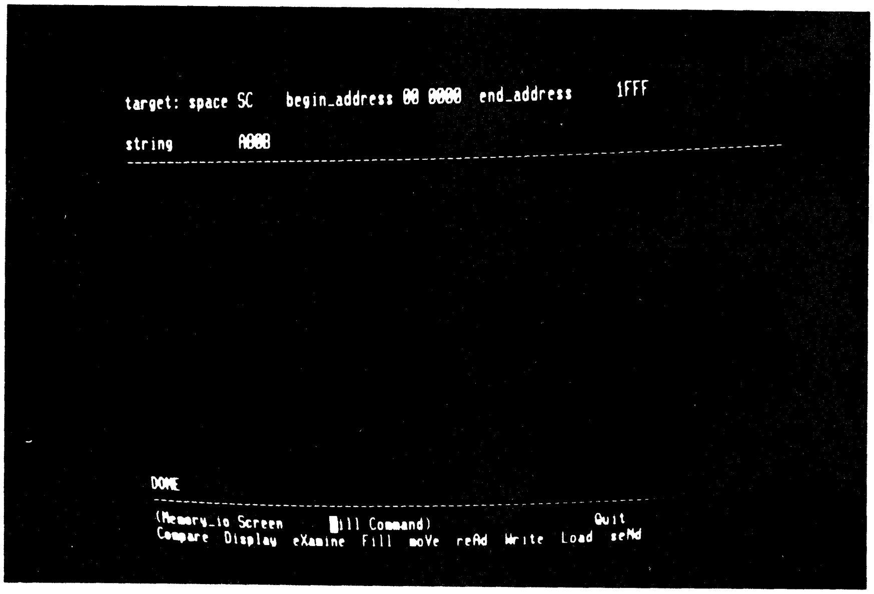


Figure 6-8. The Fill Command

Table 6-12 lists the variable fields for the Fill command.

Table 6-12. Fill Command Fields

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
target space	mult choice	SC SD SS NC ND NS MT	0 1 2 3 4 5 6	System code space System data space System stack space Normal code space Normal data space Normal stack space Monitor memory space
begin_address (segment number) (offset)	Hex-2 Hex-4	00-FF 0000-FFFF	00 0000	Segment number does not appear if Z8002 is installed This is the only address field on the Z8002.
end_address	Hex-4	0000-FFFF	0000	On the Z8001, the same segment number is used for both address fields.
string	Hex-16	empty - FFF...FFF	empty	Leading zero is implied if length is odd.

6.10.5 The moVe Command

The moVe command allows the user to copy the contents of one area of memory into another area. The contents of the source area are not altered by this command. The source and target areas can be in the same or in different address spaces, and they can overlap. Thus, it is possible for the target address to be within the block specified by the start address and by the length. Similarly, the source address can be within the block specified by the target address and the length. Note that the length is always specified in bytes.

As with the eXamine and Fill commands, the target memory area for the moVe command may include the Z-SCAN mappable memory (see Section 6.11.3). The moVe command can alter its contents even if it is write protected.

Figure 6-9 shows an example of the moVe command. This display results from setting parameters to move the contents of 10 bytes starting at location 0000 in system code space to the 10 bytes starting at location 2000 in system code space. A RETURN is entered after setting up the parameters. This executes the command and displays the DONE message when complete.

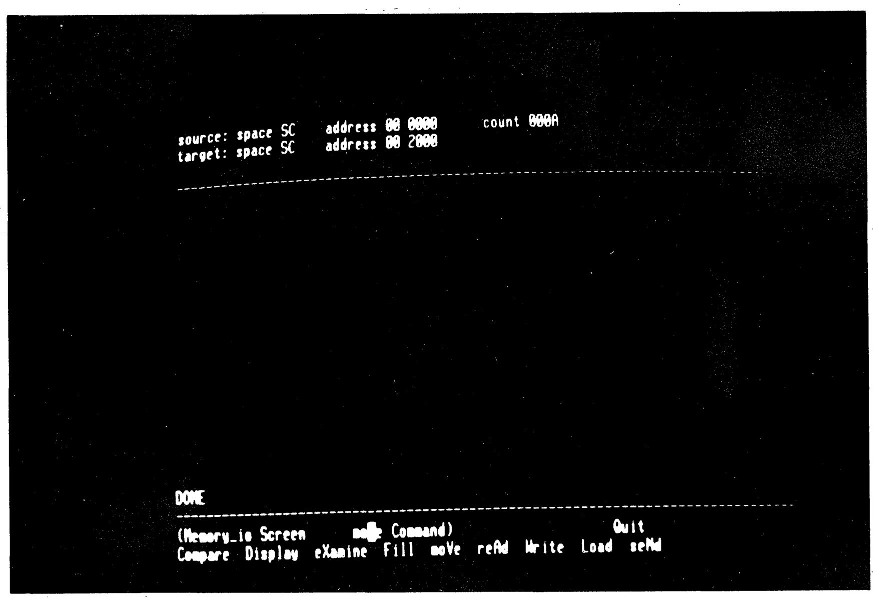


Figure 6-9. The moVe Command

Table 6-13 summarizes the moVe command's parameters.

Table 6-13. moVe Command Fields

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
source space	mult choice	SC SD SS NC ND NS MT	0 1 2 3 4 5 6	System code space System data space System stack space Normal code space Normal data space Normal stack space Monitor memory space
source address (segment number) (offset)	Hex-2 Hex-4	00-FF 0000-FFFF	00 0000	Segment number does not appear if Z8002 is installed This is the only address field on the Z8002.
count	Hex-4	0001-FFFF	0001	This is a byte count.
target space	mult choice	--same as for source space--		
target address (segment number) (offset)	Hex-2 Hex-4	00-7F 0000-FFFF	00 0000	See notes for source address.

6.10.6 The reAd Command

The user can read byte- or word-wide ports in the target system through the use of this command. Up to FFFF read operations can be performed and their results displayed. Both standard and special operations are supported by the reAd command.

Note that Z-SCAN's own ports cannot be accessed by this command, just as they cannot be accessed during emulations.

Execution of the reAd command first clears the window area, then displays the data read, one word or byte per line. An ordinal number appears to the left of each value.

If the number of operations specified by the count field in the command area is not complete when the bottom of the window is reached the cursor waits at the bottom right of the area. Two keys are valid in this context:

- Cursor down clears the screen and displays more data.
- Return aborts the command, moving the cursor to the menu area. The message ABORT is displayed.

The message DONE is displayed when the requested number of reads has taken place.

Figure 6-10 is an example of a display following execution of the reAd command.

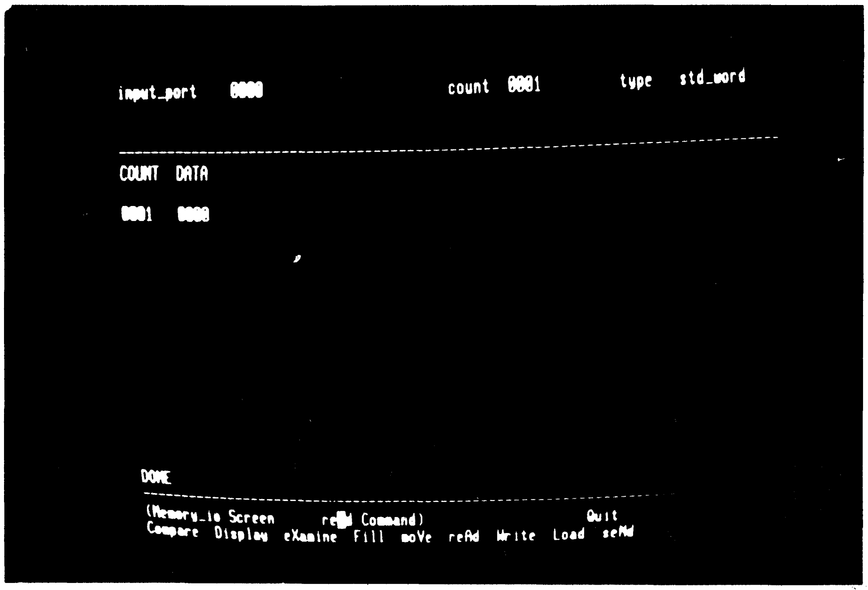


Figure 6-10. The reAd Command

Table 6-14 details the reAd command parameters.

Table 6-14. reAd Command Fields

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
input_port	Hex-4	0000-FFFF	0000	See Section 5.4.4.
count	Hex-4	0001-FFFF	0001	Counts words or bytes.
type	multi choice	std_word std_byte spl_word spl_byte	0 1 2 3	Reads whole bus. Reads bits 0-7 of bus if address is odd, or bits 8-16 of bus if address is even. Reads whole bus. Reads bits 0-7 of bus if address is odd, or bits 8-16 of bus if address is even.

6.10.7 The Write Command

With this command, a string of up to eight bytes can be written to a single byte or word-wide I/O port in the target system. The Write command also supports both standard and special operations. The output string can contain up to 16 hex digits. If the string contains an odd number of digits and the destination is a byte port, a leading zero is assumed. Thus, the high nibble of the first byte transmitted is zero. Similarly, if the destination is a word port, up to three leading zeros can be assumed to ensure that the string fills a whole number of words. For example, the user input 12345 could be interpreted as three bytes (01, 23, 45) or as two words (0001 and 2345).

As each word or byte is output, it is displayed in the window area to the right of an ordinal number. No output occurs if the command is executed with the default string, which is empty (equivalent to a user input of spaces). Note that this command cannot access Z-SCAN's own output ports. The message DONE indicates completion.

An example of the Write command's default screen display is shown in Figure 6-11.

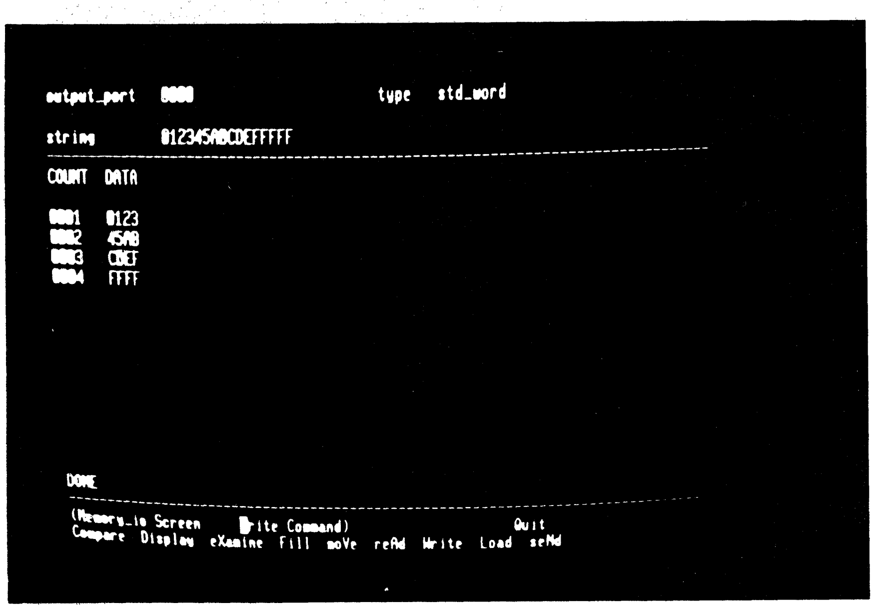


Figure 6-11 The Write Command

Table 6-15 lists the parameters for the Write command.

Table 6-15. Write Command Fields

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
output_port	Hex-4	0000-FFFF	0000	
type	mult	std_word	0	16 bits of data appear on bus
	choice	std_byte	1	Data duplicated on high and low halves of bus
		spl_word	2	16 bits of data appear on bus
		spl_byte	3	Data duplicated on high and low halves of bus
string	Hex-16	empty - FFF...FFF	empty	Leading zeros may be implied See above.

6.10.8 The Load Command

The Load command allows the downloading of executable files (procedure files) from a host system into memory controlled by Z-SCAN. This section describes the user interface to the command. Section 7 of this manual details the download transactions between Z-SCAN and the host system.

Z-SCAN requires three items of information in order to load a program. It needs to know which address space it will be loaded into. A load address is not required, since the file itself contains this information. If Z-SCAN is using the Z8001 CPU, the second item of information needed is the segment number. The last item of information required is the file name, which may contain up to 32 arbitrary characters. Refer to Section 6.6.3 for further information on file names.

When the Load command is executed, Z-SCAN requests the host system's load utility to open the requested file. It is possible that the host cannot run the load utility or that the requested file cannot be opened for some reason. In either of these cases, an error message is displayed in the window area and execution is terminated. If the host does not respond to the request at all, Z-SCAN waits indefinitely for a response. In this case, the user must abort the command by entering a BREAK.

More often, the load utility runs successfully and the contents of the file are loaded into the target memory area. As this happens, a record count is displayed at the top left of the window area. Each record contains about 30 (decimal) bytes of data. When loading is complete, the entry point of the program just loaded is displayed before execution of the command terminates. The user can abort loading at any time by entering BREAK.

Load can be used to write the contents of a program file into the Z-SCAN mappable memory, which is described in Section 6.11.3. As with other monitor commands that can write to this memory, Load is able to alter its contents even if it is write protected.

Figure 6-12 shows an example screen for the Load command. The tutorials in Section 4 include other examples of the Load command.

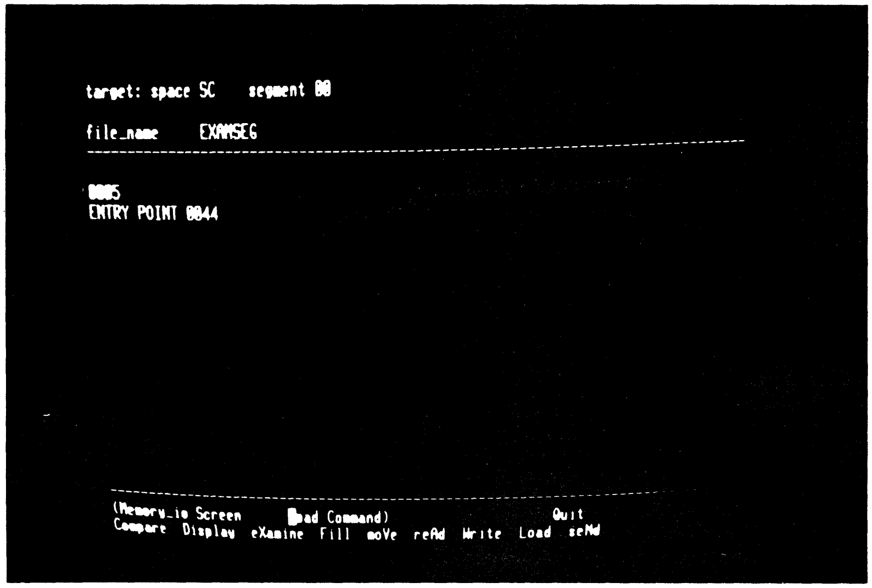


Figure 6-12 The Load Command

Table 6-15 lists the parameters for the Load command.

Table 6-16. Load Command Fields

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
space	mult choice	SC SD SS NC ND NS MT	0 1 2 3 4 5 6	System code space System data space System stack space Normal code space Normal data space Normal stack space Monitor memory space
segment number	Hex-2	00	00-7F	Segment number does not appear if Z8002 is installed
file_name	file name	any	blank	32 arbitrary characters terminated by space.

6.10.9 The seNd Command

The seNd command allows the uploading of information or procedure files contained in memory controlled by Z-SCAN. This section describes the interface to the command. Section 7 of this manual details the upload transactions between Z-SCAN and the host system.

Z-SCAN requires several items of information in order to seNd (upload) information or files to the host system. The begin_address and end_address identify the block of data to be sent from the source address space. The Z8001 requires the segment number to be stated. Both begin_address and end_address are offsets in the same segment. The number of bytes sent is (end_address - begin_address + 1), so that the last byte in the block is that at end_address.

The seNd command is intended mainly for saving patched programs. For this reason, it has an entry point field to define the address at which execution of the program should begin. The entry point must be greater than or equal to the begin_address and less than the end_address. seNd can be used to save the contents of data areas provided that a dummy entry point is supplied.

When the command is executed, the validity of the addresses is checked. If the check fails, BAD ADDRESS is displayed and the command aborts. If the parameters are acceptable, the host's SEND utility is activated. An error message is displayed if the host cannot load the utility or if the file name is unacceptable--for example, the file already exists with the same file name as the one given in the seNd command. More normally, the host program is activated without error and records are sent to the host by Z-SCAN. An incrementing number at the top left of the screen counts the records, which each contain 30 or fewer bytes. Execution terminates when the transfer is complete.

Figure 6-13 shows an example of the initial display obtained by accessing the seNd command.

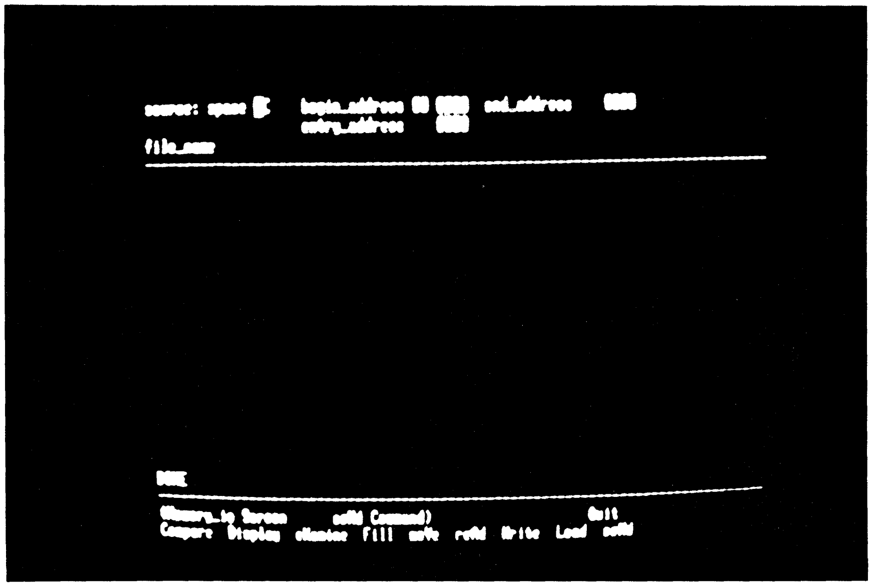


Figure 6-13 **The seNd Command**

Table 6-17 describes the parameters for the seNd command.

Table 6-17. seNd Command Fields

Field Name	Type	Values (M-C) Range (hex)	Index/ Default	Notes
source space	mult choice	SC SD SS NC ND NS MT	0 1 2 3 4 5 6	System code space System data space System stack space Normal code space Normal data space Normal stack space Monitor memory space
begin_address (segment number) (offset)	Hex-2 Hex-4	00-7F 0000-FFFF	00 0000	Segment number does not appear if Z8002 is installed. This is the only field on the Z8002. Must be less than end_address.
end_address	Hex-4	0000-FFFF	0000	Must be greater than begin_address.
entry_address	Hex-4	0000-FFFF	0000	Must be in range of begin_address to end_address -1.
file_name	file_name	any	blank	32 non-blank characters

6.11 RESOURCES SCREEN

Before an emulation can be run, a number of parameters must be set up to define the emulation starting conditions, its environment and the constraints placed on it. Without such controls, the emulator would have little advantage over a CPU for debugging purposes.

The Resources screen allows the user to enter control information of this type. A set of six commands is available on this screen. Some of these configure the Z-SCAN resources, for example, its mappable memory and breakpoint logic, whereas others affect the CPU's state and behavior during emulations.

The Resources screen is shown in Figure 6-14. Above the Menu area there are three command areas which are further divided into six subscreens, one for each command. Unlike the Memory_io screen, the Resources screen displays all its subscreens continually, whether the associated command has been activated or not. Commands are activated by entering the capital letter from their names when the cursor is in the menu area.

Once a command has been activated, the cursor automatically moves into the first variable field in the associated subscreen, which allows the contents of the fields to be updated as described in Section 6.6. Entering a RETURN or BREAK moves the cursor back to the menu area.

Figure 6-14. The Resources Screen

Each of the following subsections describes one of the six commands available from the Resource screen (Break, Inst_count, mAp, reGister, Peek and Wait_states).

6.11.1 The Break Command

The Z-SCAN breakpoint logic is very flexible, allowing BREAKs to be set or pulses to be output in response to a wide variety of conditions. Consequently, the monitor must maintain a large number of variables in order to control its functions. The Break command provides a comprehensive and comprehensible interface to those functions. The default parameters for the Break command fields are included in the Resources Screen, Figure 6-14.

One of the two main inputs to the Z-SCAN breakpoint logic is the address/data bus and, in the case of the Z8001 only, the segment number. The logic can be programmed to search for address matches or data matches, but not both at once. Hence, it is possible to request a break to occur when a particular location is read, but it is not possible to simultaneously request that the break take place only when a particular data pattern is read from that location. Instead, a break can be programmed when that data pattern is read from any location. On the Z8002, the only address field is the 16-bit address/data bus contents. On the Z8001, it may be the segment number, the offset, or both. The segment field is ignored when searching for data matches.

The Z-SCAN address/data comparator is 16 bits wide. This must be taken into account when setting up breaks conditioned on byte data. With byte write operations, the user can take advantage of the fact that the Z8001 duplicates the eight bits of data on the upper and lower halves of the bus during all such operations. For example, to break when an ASCII A (code 41 hex) is written, load the match field with 4141. During byte read operations, it is difficult to predict the data that will be seen on the unused part of the bus (upper byte for odd addresses, lower byte for even), so it is seldom possible to program a break conditioned on byte read data. For similar reasons, the user is cautioned against the setting of breakpoints on specific 8-bit interrupt vectors or 9-bit refresh addresses.

The other input to the breakpoint logic is the set of seven signals that constitute the Z8000's status: read/write, normal/system, byte/word and the four status code lines, ST_{0-3}. Users can select any of 128 possible combinations, although some of these are meaningless because the CPU never generates them. An example of such a meaningless status is a normal mode I/O operation, a transaction that is specifically prohibited by the processor architecture.

The design of the breakpoint logic makes it possible to break following a non-maskable interrupt acknowledge cycle. However, because the logic that controls the change of mode from Target to Monitor traps this particular status code, an NMI acknowledge that triggers the breakpoint is prevented from reaching the target system. This is likely to prevent the target's service routine from being entered correctly, even when emulation is resumed. For this reason, breaks on NMI acknowledges are not allowed and the corresponding status code appears as "reserved".

When some characteristic of either Z-SCAN or the target system makes it undesirable to select a certain set of breakpoint conditions, it is almost always possible to program an alternative condition that ends the emulation under identical or very similar circumstances. Such alternatives take advantage either of program flow or of CPU characteristics. In the case of NMI acknowledge, the operating sequence of the CPU ensures that the new program status area in system code space is referenced soon after such a cycle. Thus, a breakpoint placed on the NMI status entry is an acceptable substitute for a break on NMI acknowledge. Similarly, it may be possible to pick out a certain instruction that will be executed if and only if a particular byte data value is read.

The outputs of the address/data and status comparators feed the trigger logic. Either of these inputs can be masked out (a "don't care" condition). The trigger logic can be fired either when both of its inputs are true (the logical AND condition represented by enable*) or when either of the inputs is true (the logical OR condition, enable+). If either of the inputs is a "don't care" when a logical OR trigger is selected, the break condition is always satisfied, and any emulation stops at once if the trigger logic is enabled (pulse_&_break selected).

The trigger can be further conditioned by a pass counter, which can count up to FF pulses (255 decimal) before providing a trigger output. Values other than 01 should not be used in conjunction with the Execution screen Trace command (Section 6.12.3) because this command loads the pass counter with the count value before each instruction is emulated. This usually prevents a multiple pass break condition from being satisfied.

Figure 6-15 is a conceptual diagram of the breakpoint logic chain. It is provided to help users visualize the action of the logic, not to illustrate the actual implementation. The instruction counter shown on the diagram is discussed in connection with the Inst_count command in the Section 6.11.2.

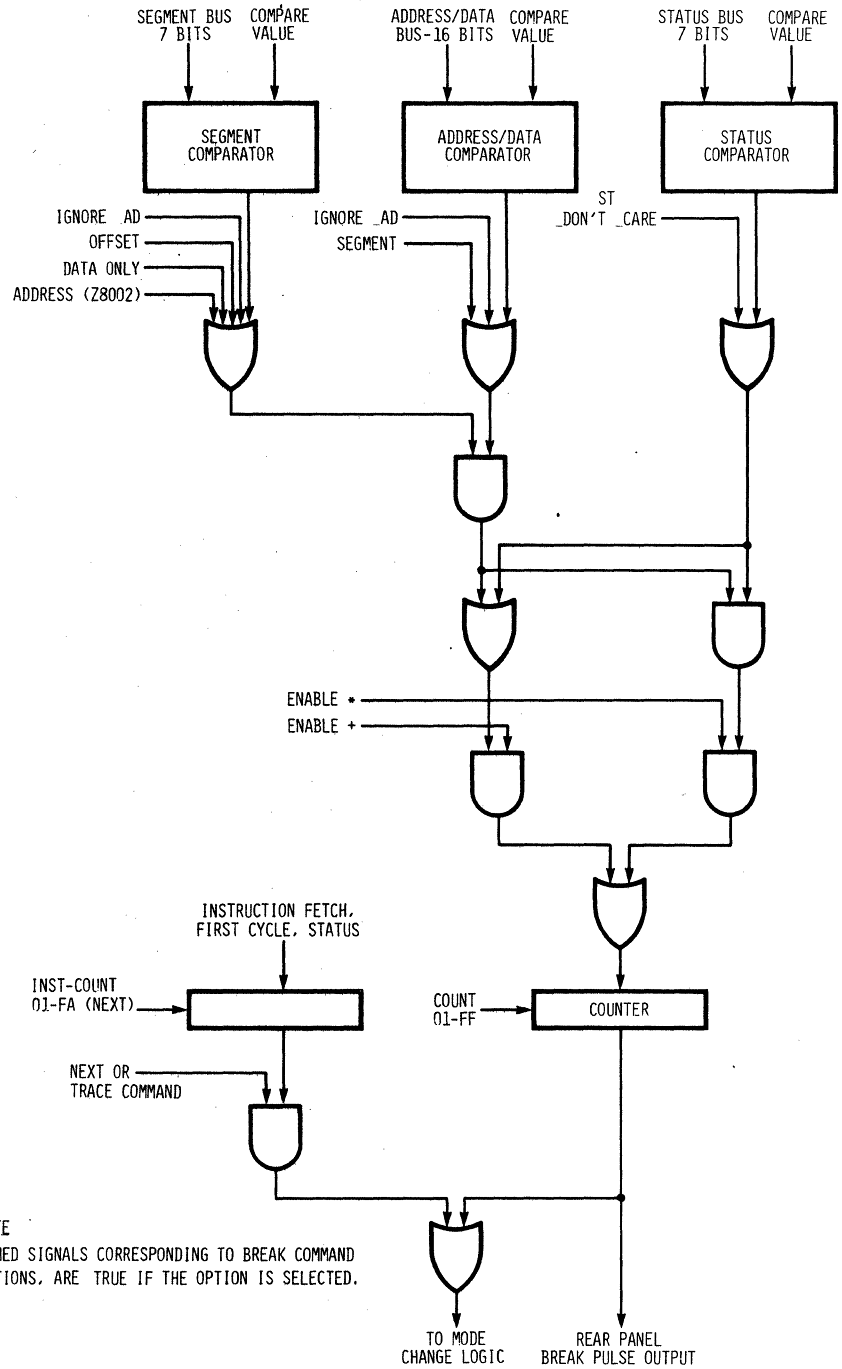


Figure 6-15. Z-SCAN Breakpoint Logic (Conceptual Diagram)

Table 6-16 lists the Break command fields and their possible values. Steps 19 and 20 of the tutorial in Section 4.5, together with their associated Figures 4-14 and 4-15, show the setting of a breakpoint that uses the enable* option. The alternative, enable+, is shown in step 55 and Figure 4-43.

Table 6-18. Break Command Fields

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
master enable	mult choice	disable enable+ enable*	0 1 2	Inhibit pulse and break Enable, OR condition Enable, AND condition
effect	mult choice	pulse_&_break pulse_only	0 1	Trigger ends emulation Trigger does not stop emulation
status select	mult choice	status ST_dontcare	0 1	Status affects break Status ignored
address/data select (Z8002 installed)	mult choice	address data ignore_AD	0 1 1	Address affects break Data affects break Address/data ignored
address/data select (Z8001 installed)	mult choice	seg*offset offset segment data ignore_AD	0 1 2 3 4	Segment and offset must match for break Offset affects break Segment affects break Data affects break Address/data and segment ignored
segment number		00-7F	00	Used for address matches only with Z8001
match pattern	Hex-4	0000-FFFF	0000	See above notes
count	Hex-2	01-FF	01	Pass count field
read/write	mult choice	read write	0 1	
normal/system	mult choice	system normal	0 1	
byte/word	mult choice	word byte	0 1	

Table 6-18. Break Command Fields--Continued

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
status code	mult choice	instr_fetch1	0	Instruction fetch, 1st word
		internal_op	1	Internal operation
		mem_refresh	2	Memory refresh
		io_reference	3	Standard I/O
		special_io	4	Special I/O
		seg_trap_ack	5	Not produced by Z8002
		reserved	6	NMI acknowledge. See above
		nvi_ack	7	Non-vectored int. ack.
		vi_ack	8	Vectored interrupt ack.
		data_mreq	9	Data memory access
		stk_mreq	A	Stack memory access
		EPU_data_mrq	B	Extended Processing Unit
		EPU_stk_mrq	C	Memory operations
		code_sp_access	D	Code space access, Nth word
		EPA_transfer	E	CPU to EPU transfer
		reserved	F	Not used by CPU

6.11.2 The Inst_count Command

As well as the breakpoint logic described in the previous section, Z-SCAN contains an instruction counter that can be programmed to stop an emulation after a given number of instructions have been executed. The user must start the emulation with the Execution screen Next command (see Section 6.12.2) if the instruction counter is to affect the trigger condition. Figure 6-5 shows the relationship between the instruction counter and the breakpoint logic.

The instruction counter can be used in conjunction with a programmed break condition to run emulations that stop either when a specified number of instructions is executed or when a particular condition is detected on the bus. This feature further increases the flexibility of Z-SCAN.

Up to FB (251 decimal) instructions can be counted. The difference between this maximum and that of the breakpoint pass counter (FF) arises because four instructions are fetched between the time that the instruction counter is enabled and the time at which the next emulation actually begins. Table 6-19 summarizes the range of values for the field.

Table 6-19. Inst_count Command Field

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
Inst_count	Hex-2	01-FB	01	

6.11.3 The mAp Command

Z-SCAN contains an 8K byte (8192 decimal bytes) block of mappable memory. This feature facilitates the development and debugging of target hardware, because the mappable memory can be used as a substitute for memory in the target system. This feature is generally used in one of two situations:

- **Target memory not implemented.** Often, when a prototype is made, it does not include as much memory as is provided by the final design. Alternatively, it may be found that the preliminary memory design does not function correctly. In either of these cases, Z-SCAN mappable memory can substitute for the missing or faulty memory, allowing software debugging to proceed even before the target hardware is fully functional. The memory can also be used for the loading of test programs that exercise the target hardware, thus speeding hardware development.

- **Target memory is read-only.** Many applications, particularly those addressed by small, dedicated systems, require software that is totally ROM (Read-Only Memory) based. This non volatile method of program storage allows systems to operate without requiring backup storage devices (floppy disks, for example) to save and protect the software when power is removed from the equipment. The production advantages of ROM-based software (also known as firmware) are balanced by a development disadvantage: it is difficult to make changes in firmware in order to debug applications programs. The non volatile nature of the memory means that it must be removed from the prototype in order to modify its contents--if they can be modified at all. Z-SCAN circumvents such problems by allowing the mappable memory to substitute for target system ROM. Because the memory can be written as well as read, the user can easily make changes to its contents as debugging proceeds. An additional feature protects the mappable memory against write accesses, permitting it to simulate read-only memory.

The mAp command allows the user to define the types of memory accesses to which the mappable memory responds. The address space (or spaces) in which the memory appears must be selected, then the ranges of addresses within those spaces. Finally, write protection can be enabled or disabled, as required.

Mappable memory can appear in any combination of the Z8000's six memory address spaces (system code, system data, ..., normal stack) and can be mapped as a single block onto any 8K byte boundary within those spaces. Such boundaries are multiples of 2000 hex. It is not possible for the memory to appear at different addresses in different spaces, nor is it possible for it to be write-enabled in one space and write-protected in another. When the memory is mapped at a particular address in a particular block, it responds to all CPU accesses in the range between the base address and base address + 1FFF hex. Memory in the target may respond in the same address range. Even so, CPU read accesses read data from the mappable memory, not from the target. CPU writes are to both the mappable memory and the target memory.

The user may sometimes want to develop ROM-based applications that require more memory than is provided by the Z-SCAN mapping feature. The recommended approach in such cases is to develop and debug the software as a number of separate pieces, consigning each piece to ROM when it is considered fully functional. This allows the amount of code in the mappable memory to be kept within the allowed limits. The Z8000 support software available on many host systems permits newly written routines to be linked to existing procedures in ROM.

Table 6-20 details the mAp command's parameter fields. The tutorials in Section 4 include examples of operations using the mAp command.

Table 6-20. mAp Command Fields

Field name	**Type**	**Values (M-C) Range (Hex)**	**Index/ Default**	**Notes**
system code select	mult choice	— SC	0 1	Mappable memory does not respond to system code accesses. Mappable memory responds to system code accesses.
system data select	mult choice	$\overline{\text{SD}}$	0 1	Similar to first field
system stack select	mult choice	$\overline{\text{SS}}$	0 1	Similar to first field
normal code select	mult choice	$\overline{\text{NC}}$	0 1	Similar to first field
normal data select	mult choice	$\overline{\text{ND}}$	0 1	Similar to first field
normal stack select	mult choice	$\overline{\text{NS}}$	0 1	Similar to first field
segment	Hex-2	00-7F	00	Segment number does not appear if Z8002 is installed.
address	mult choice	0000 2000 C000 E000	0 1 6 7	This is NOT a hex field , because the index required by the choice table.
protection	mult choice	unprotect protect break	0 1 2	Mappable memory can be written. Mappable memory cannot be written, but emulation continues if an attempt is made to do so. Mappable memory cannot be written, and emulation terminates if an attempt is made to do so.

6.11.4 The reGister Command

By default, Z-SCAN saves the contents of the CPU registers at the end of each emulation and retores the same values at the start of the next emulation. The reGister command allows the user to alter register contents between emulations so that the next emulation starts with modified values. For example, the Program Counter's contents can be changed so that the next emulation does not start where the previous one finished, or in order to set up the entry point of a newly loaded file.

Changes made to register contents on the Resources screen are reflected in the upper rows of register values on the Execution screen (see Section 6.12). Table 6-21, parts 1 and 2, details the reGister command's variable fields. The tutorials in Section 4 demonstrate the use of the reGister command.

Some of the control registers are used only on the segmented Z8001. They do not appear when a Z8002 is installed in Z-SCAN. These registers are the Program Counter and new program status area segment numbers and normal mode register 14 (NSPSEG).

Table 6-21. reGister Command Fields

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
R0	Hex-4	0000-FFFF	0000	Word register zero. High byte is byte register RH0, low byte is RL0. Can be used as more significant part of long word reg. RR0 or as most significant part of quad reg. RQ0.
R1	Hex-4	0000-FFFF	0000	Also RH1, RL1, RR0, RQ0
R2	Hex-4	0000-FFFF	0000	See notes above
R3	Hex-4	0000-FFFF	0000	See notes above
R4	Hex-4	0000-FFFF	0000	See notes above
R5	Hex-4	0000-FFFF	0000	See notes above
R6	Hex-4	0000-FFFF	0000	See notes above
R7	Hex-4	0000-FFFF	0000	See notes above
PC Seg. no.	Hex-2	00-FF	00	Segment number for Program Counter. This does not appear if Z8002 CPU installed. MSB may be set as a result of program execution, but cannot be set by user.

Table 6-21. reGister Command Fields
(continued)

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
PC (offset for Z8001)	Hex-4	0000-FFFF	0000	Program Counter. Action of CPU undefined if contents are odd.
FCW	Hex-4	0000-FFFF	C000	Flag and Control Word. See Z8000 Technical Manual to determine legal values. Default is segmented system mode with interrupts disabled. The Z8002 ignores the fact that the segmentation flag is set.
R8	Hex-4	0000-FFFF	0000	Also RR8, RQ8
R9	Hex-4	0000-FFFF	0000	See notes above
R10	Hex-4	0000-FFFF	0000	See notes above
R11	Hex-4	0000-FFFF	0000	See notes above
R12	Hex-4	0000-FFFF	0000	See notes above
R13	Hex-4	0000-FFFF	0000	See notes above
R14	Hex-4	0000-FFFF	0000	See notes above. Also segment number of system stack pointer for Z8001.
R15	Hex-4	0000-FFFF	0000	System mode Stack Pointer, must contain an even value if used for this function. Also less significant half of RR14 in system mode only
PSAP Segment no.	Hex-2	00-7F	0000	Program Status Area Pointer segment number. Not displayed if Z8002 is installed.
PSAP (offset for Z8001)	Hex-4	0000-FFFF	0000	Program Status Area Pointer. Low byte is ignored by CPU.
NSP Segment Segment no.	Hex-4	0000-FFFF	0000	Normal mode Stack Pointer Segment no. (R14). Not displayed if Z8002 is installed.
NSP offset	Hex-4	0000-FFFF	0000	Normal mode stack pointer (offset on Z8001) R15.

6.11.5 The Peek Command

Z-SCAN automatically captures and displays the contents of the CPU registers at the end of an emulation. This information appears on the Execution screen (Section 6.12) and can be updated with the reGister command (Section 6.11.4). Often, the user wants to know the contents of selected areas of memory at the end of each emulation as well as register contents. In order to satisfy this request, Z-SCAN provides three windows, each four words in length, into target memory and displays the contents of these windows alongside the register information. Note that this usage of the word window is different from that used in connection with the window area on the Memory_io screen.

The Peek command allows the user to select the memory space and start address for each of the three windows. Table 6-22 lists its variable fields. An example of its use is given in step 66 of the tutorial, Section 4.6.

Table 6-22. Peek Command Fields

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
space #1	mult choice	SC SD SS NC ND NS	0 1 2 3 4 5	System code space System data space System stack space Normal code space Normal data space Normal stack space
address #1 (segment number)	Hex-2	00-7F	00	Window segment number. Segment number does not appear if Z8002 is installed.
(offset)	Hex-4	0000-FFFF	0000	This is the only address field on the Z8002.
space #2	mult choice	--same as space #1--		
address #2 (segment number) (offset)	Hex-2 Hex-4	00-7F 0000-FFFF	00 0000	See notes for address #1.
space #3	mult choice	--same as space #1--		
address #3 (segment number) (offset)	Hex-2 Hex-4	00-7F 0000-FFFF	00 0000	See notes for address #1

6.11.6 The Wait_states Command

Not all memory components meet the maximum access time requirement of 350 ns required by the Z8000 to run at full speed with a 4 MHz clock frequency. A common example of such a memory component is the EPROM (Erasable Programmable Read Only Memory), which has a typical access time of 450 ns. If the Z8000 is used with slow memory or with slow I/O, its access time requirement must be increased. This function is handled by the WAIT- input to the CPU, which must be driven by an external wait state generator in systems that cannot meet full speed access time requirements.

To eliminate the need for each user to implement a wait state generator at an early stage in development, Z-SCAN provides its own generator. This can insert between zero and eight wait states in each memory transaction. It also affects I/O and interrupt acknowledge cycles, as detailed in Table 6-23. The differences between the three types of transactions arise because the CPU samples the WAIT- signal at a different time, relative to Address Strobe, in each type of transaction. Note that refresh operations are not affected because the CPU does not sample the WAIT- line during these cycles. Table 6-24 summarizes the choice of values available in the single variable field controlled by this command.

Table 6-23. Effect of Wait States

Wait_states Field Value	Memory Reference Cycles Added	Memory Reference Total Length	I/O Reference Cycles Added	I/O Reference Total Length	Interrupt Acknowledge Cycles Added	Interrupt Acknowledge Total Length
0	0	3	0	4	0	10
1	1	4	0	4	0	10
2	2	5	1	5	0	10
3	3	6	2	6	0	10
4	4	7	3	7	1	11
5	5	8	4	8	2	12
6	6	9	5	9	3	13
7	7	10	6	10	4	14
8	8	11	7	11	5	15

Table 6-24. Wait_states Command Field

Field name	Type	Values (M-C) Range (Hex)	Index/ Default	Notes
Wait_states	mult choice	0	0	No waits
		1	1	
		2	2	
		3	3	
		4	4	
		5	5	
		6	6	
		7	7	
		8	8	

6.12 THE EXECUTION SCREEN

The Execution screen differs from those discussed previously in that it has no variable fields. Instead, it performs the following functions:

- Display of Z-SCAN parameters pertinent to emulation--for example, the setting of the breakpoint logic.
- Display of information on the status of the processor and selected target memory areas before and after each emulation.
- Display of a message that indicates the reason for termination of each emulation.
- Provision of commands to start emulations (Go and Next).
- Access to the Trace screen for detailed program analysis.

Figure 6-16 shows the screen as it appears between emulations. It is shown in this state rather than in its default state to better illustrate the functions of the various fields. Note that the screen is divided into five command areas (although they are not used for the entry of command parameters) and a menu area.

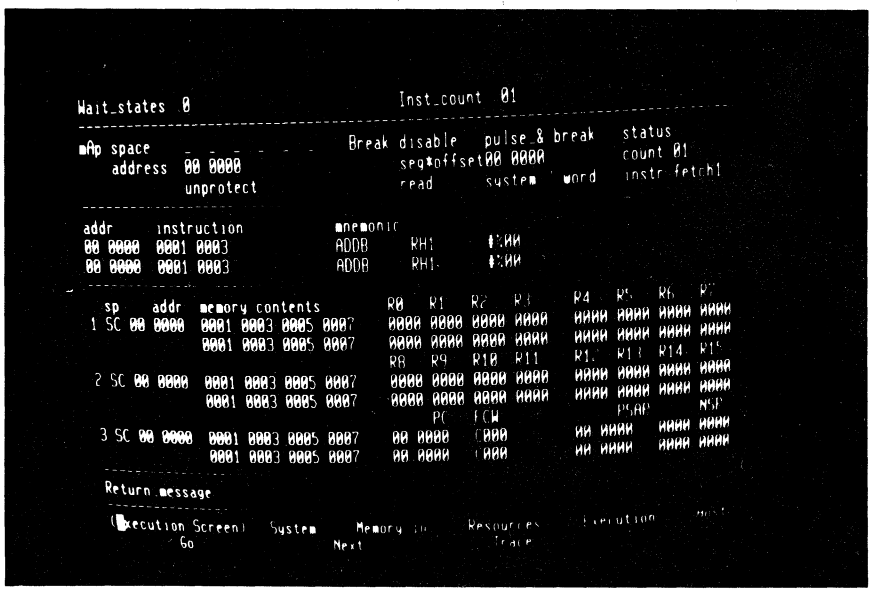


Figure 6-16. The Execution Screen

The two top command areas contain Z-SCAN status information that describes the state of the Wait_states, Inst_count, mAp and Break variable fields. These displays are for reference only, since the fields can be modified only by Resources screen commands. For further details, see Sections 6.11.1 through 6.11.3 and 6.11.6.

Moving toward the bottom of the screen, the next command area contains two lines of data that correspond to the Program Counter contents before and after the last emulation and to the instructions at those locations. The current information, captured after the emulation stopped, appears as the upper row of data, and the status from before the emulation is displayed on the lower line. The display lines are generated by the disassembler described in Section 6.10.2. Disassembly is always nonsegmented if a Z8002 is installed in Z-SCAN. For the Z8001, the segmentation bit (bit 15) of the FCW controls disassembly segmentation mode.

Below the instruction information is a large command area that contains information about register and memory contents. The memory locations displayed to the left of this area are selected by the Resources screen Peek command (see Section 6.11.5). There are two rows of data for each location, again corresponding to the status before (lower row) and after (upper row) the last emulation.

To the right of the memory trace information is a display of register contents. As with the instruction and trace fields, the two rows of data represent the status before and after the last emulation. As with the Resources screen reGister command, three control registers do not appear if a Z8002 is installed in Z-SCAN. Refer to Table 6-21 for details. It is possible for bit seven of the Program Counter segment number to be set by program execution on the Z8001, for example, when a long offset direct address mode call instruction is executed. This has no consequence: a segment number of 80 is equivalent to 00, 81 to 01 and so on.

Register contents and traced addresses can be changed using the Resources screen reGister and Peek commands, respectively (see Sections 6.11.4 and 6.11.5). Any such change is reflected in the top entry for the corresponding field on the Execution screen. In other words, it is the status captured after the last emulation that is modified. It is this status that is used when the next emulation begins.

The bottom Command area on the Execution screen is headed Return_message. It is here that the monitor writes a short message stating where ($\overline{PC}$ contents) and why (cause of transition from Target to Monitor mode) the previous emulation terminated. This field is blank when the Execution screen is first displayed unless a segment trap is outstanding on the Z8001. Section 5.4.6 discusses this situation. Table 6-25 lists the four possible causes of break conditions. One or more of these messages appears each time an emulation terminates.

Table 6-25. Termination Messages

Message	**Notes**
TRIGGER BREAK	The condition set up by the Break command was satisfied.
MANUAL BREAK	The user generated a monitor NMI with the Z-SCAN front panel switches.
WRITE PROTECT BREAK	The executing program attempted to write into the Z-SCAN mappable memory when a write-protect break was enabled.
STEP BREAK	The number of instructions defined by the Inst_count field was executed during an emulation started with the Next command.

The menu area of the Execution screen lists two commands that are specific to the screen: Go and Next. These are described in Sections 6.12.1 and 6.12.2. There is a third command, Trace, which generates its own screen. The Trace command is described in Section 6.12.3.

6.12.1 The Go Command

This command starts an emulation at the address held in the PC register. The emulation continues until one of three conditions occur:

- The breakpoint condition selected with the Break command is met.
- The user generates a Monitor NMI with the Z-SCAN front panel switches.
- The program running under emulation attempts to write into the Z-SCAN mappable memory when a break is set on write protect violation.

If any of the above conditions is detected, the emulation either stops at once or executes a maximum of one more instruction in Target mode before Z-SCAN switches back to Monitor mode. This implies that the CPU must be executing instructions so that the transition between modes can occur. This topic is discussed in Section 5.4.8. The tutorials in Section 4 include examples of the Go command.

6.12.2 The Next Command

The Next command allows the user to start an emulation that executes a given number of instructions before it is automatically terminated. The number of instructions is determined by the contents of the Inst_count field, which can be modified on the Resources screen (see Section 6.11.2). This command is useful for stepping through programs one or more instructions at a time.

It is possible for an emulation started by the Next command to terminate before the instruction count is exhausted if an alternative break condition arises. The Next command can terminate when:

- The programmed instruction count is exhausted.
- The breakpoint condition selected with the Break command is met.
- The user generates a monitor NMI with the Z-SCAN front panel switches.
- The program running under emulation attempts to write into the Z-SCAN mappable memory when a break has been set on write-protect violation

If any of the above conditions is detected, the emulation either stops at once or executes a maximum of one more instruction in Target mode before Z-SCAN switches back to Monitor mode. This implies that the CPU must be executing instructions so that the transition betweeen modes can occur. This topic is discussed in Section 5.4.8 (Termination of Emulation). The tutorials in Section 4 include examples of the use of the Next command.

The user should be aware of the effect of stepping through the Z8002's block instructions, for example LDIR or OTDRB. When Inst_count is set to one, each step results in a single operation on one word or byte, and the Program Counter value is not changed by the operation unless the Block Count register named in the instruction is decremented to zero. If, on the other hand, Inst_count has a value greater than one, the block instruction executes in its entirety and counts as just one instruction as far as the Z-SCAN instruction counting logic is concerned. This is a consequence of the interaction between the interruptable block instructions and Z-SCAN's use of non-maskable interrupts to terminate emulation (see Section 5.4.6). The Z8000 CPU Technical Manual provides more information about the operation of block instructions.

6.12.3 The Trace Command

The Trace command provides a more detailed picture of program execution than either the Go or Next command because it disassembles and displays each instruction before it is executed. Disassembly incurs a time penalty, so unlike Go or Next, Trace cannot run emulations in real time. The monitor traces the user program by forcing a non-maskable interrupt after each instruction. This prevents acceptance of any interrupts or traps generated by the target hardware. Section 5.4.6 discusses this behavior in detail.

Figure 6-17 shows the Trace screen in its default state. The contents of all memory data fields in the figure is arbitrary and has no significance in the following discussion.

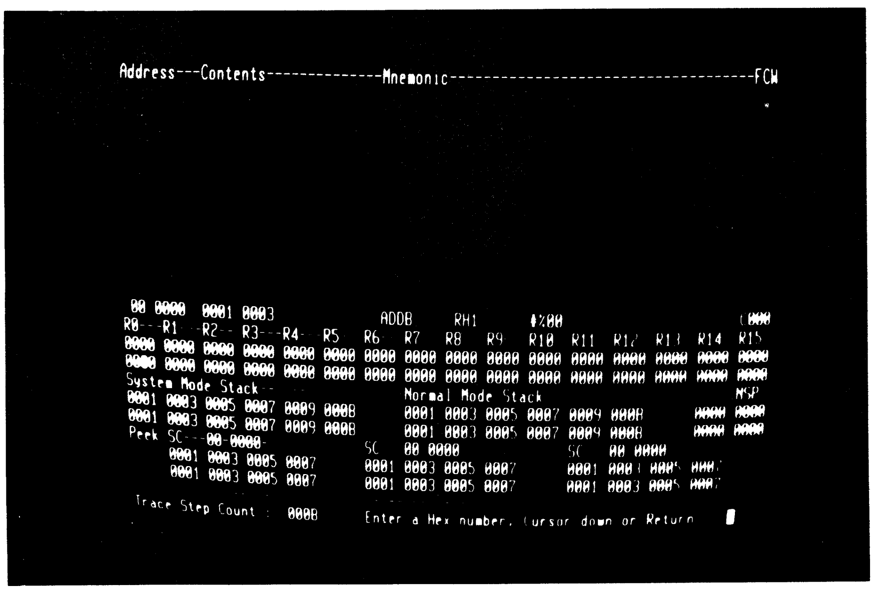


Figure 6-17. Default Trace Display

In its default state, the screen displays a heading followed by ten blank lines. Below these, a single disassembled instruction appears in the format described in Section 6.10.2, followed by the FCW value. Disassembly mode is always nonsegmented if a Z8002 is installed in Z-SCAN. The segmentation bit (bit 15) of the FCW controls the mode on the Z8001. The instruction will be executed when emulation starts.

Below the instruction, two lines of general-purpose register contents are shown. The values in the top line are loaded before the next emulation starts; those in the lower line were used when the last emulation started. Space limitations prevent the display of the normal mode stack pointer (NSP) in this area. It appears elsewhere on the screen. The program status area pointer (PSAP) is not shown on the Trace screen.

Two stack area displays appear below the register contents. Again there are two lines of data, corresponding to the current and previous stack states. The left hand area displays the 12 (decimal) bytes at the top of the system stack, which starts at the address in RR14 (Z8001 segmented mode) or R15 (Z8001 nonsegmented mode or Z8002). Thus, the address of the first byte displayed varies as data is added to or removed from the system stack.

To the right, the normal stack is displayed in a similar format. The normal stack pointer register contents appear at the far right. The segment number register (NSPSEG) is not displayed if a Z8002 is installed in Z-SCAN.

The final data area displays the contents of the memory areas defined by the Resources screen Peek command (Section 6.11.5). Current and previous contents appear on the upper and lower lines respectively.

The field at the left of the bottom line defines a step count; that is the number of instructions to be executed when emulation starts. The default value is 000B (11 decimal), sufficient to fill the upper half of the screen with disassembled instructions. The value can be changed as described below. A prompt filling the remainder of the line invites the user to enter hex digits, cursor down or RETURN. The effect of these keys are as follows:

- Enter cursor down to execute the number of instructions shown in the step count field.

- Enter hex digits (0-9, A-F) to alter the value in the step count field. Use cursor left or SHIFT < to backspace over and delete incorrect entries. Use cursor down to begin executing the number of instructions indicated by the modified step count.

- Enter RETURN to move to the Execution screen.

When tracing starts, the bottommost instruction on the screen is redisplayed. An asterisk in column one shows that it is the first instruction executed in the series of traced instructions.

Tracing can be stopped by four events:

- The number of instructions defined by the step count has been traced. This is the normal termination. The prompt is redisplayed on the bottom screen line.

- The condition set up by the Resources screen Break command (Section 6.11.1) is satisfied. Tracing terminates at once and the message TRIGGER BREAK replaces the prompt.

- o A traced instruction attempts to write to write protected mappable memory. Tracing terminates at once and the message WRITE PROTECT BREAK replaces the prompt.

- o The user enters the terminal keyboard BREAK key. Tracing terminates at once and the message MANUAL BREAK replaces the prompt.

When tracing terminates, the register, stack and peek memory contents fields are updated. All instructions shown in the top half of the screen have been executed except the bottommost. The next emulation starts by executing this instruction. The FCW values at the right of the screen show the state of the CPU **before** the execution of the instruction to the left. Figure 6-18 shows the Trace screen after execution.

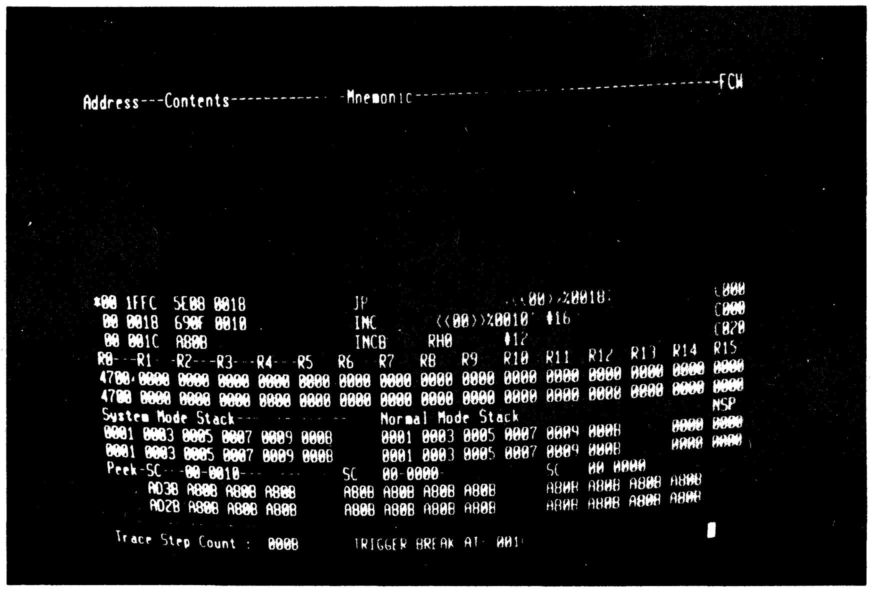


Figure 6-18. Trace Screen after Execution

Three termination conditions described above replace the prompt on the bottom screen line with an informative message. The user input required is not changed: hex digits, cursor down and RETURN are still valid keys.

As discussed in Section 5.4.7, the Z-SCAN monitor may violate protection attributes set up in the memory manager of a Z8001-based target system. The user is warned of this condition by the appearance of a warning message near the bottom right of the screen. Segment traps cannot be serviced during tracing for the reasons outlined at the beginning of this section. Other screens and commands must be used to correct the condition.

6.13 The Host Screen

The characteristics of the host screen are defined almost entirely by the host system rather than by Z-SCAN. The initial screen, which is displayed in response to the Host command, is shown in Figure 6-19. It is clear, except for the message "Host" at the top left. When the contents of the host baud rate and terminal baud rate fields set up on the System screen (Section 6.9) differ, a supplementary message, "Set baud rate of terminal to (speed), then enter RETURN and ">", appears. The speed is taken from the host baud rate field. The monitor waits until a RETURN character is received at the host baud rate before enabling communication between the terminal and the host system. Consult the terminal documentation for details of its baud rate setting procedure.

When the baud rate is correct, Transparent mode is entered. This mode is terminated when the user enters BREAK. If the System screen host and terminal baud rates are the same, the System screen is displayed at once. If they differ, the procedure described in the previous panel is repeated, telling the user to set the baud rate of the terminal to that required by the monitor. The System screen is displayed after a RETURN has been entered at the correct baud rate.

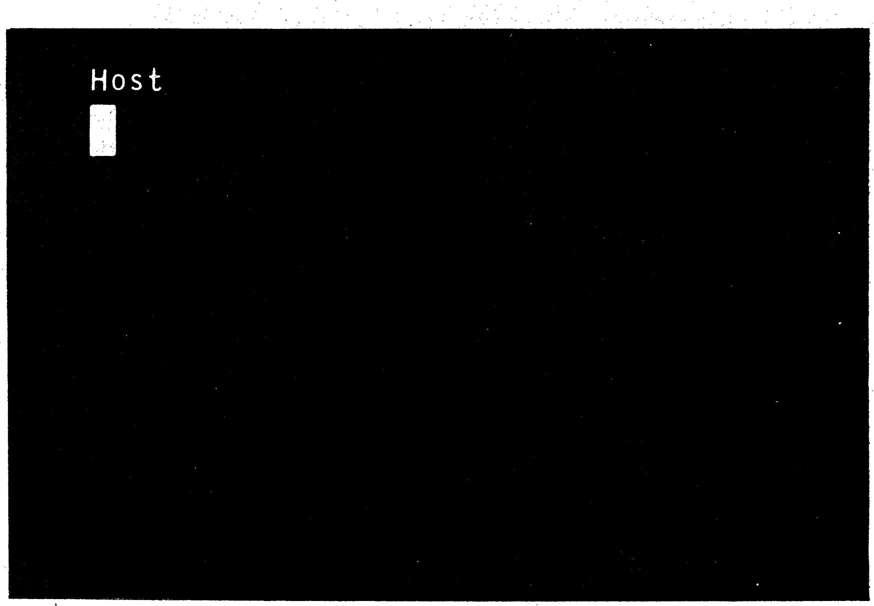


Figure 6-19. The Host Screen

SECTION SEVEN

INTERFACE TO NON-ZILOG HOSTS

7.1 INTRODUCTION

This section describes the overall communications protocol of the Z-SCAN monitor so that custom host software can be designed to interact successfully with the Load/seNd utility built into Z-SCAN.

Load/send communication between a Zilog (or other) host system and the Z-SCAN monitor is accomplished by exchanging messages containing printable ASCII characters. Message types are:

- o Single-character, data-block acknowledgement
- o Error text
- o Data block

All messages exchanged during a Load or seNd command are text lines, each ending in RETURN (carriage return). Memory and other data are converted into hexadecimal numerals for transmission, and the resultant message is readable left-to-right, high-order digit first, as it is transmitted over the RS-232 link.

The following illustrates a simplified form of Z-SCAN's Load/seNd transmission protocol:

Mode	Host Message	Message Path	Z-SCAN Message
either mode	-- acknowledge	<---- ---->	start command --
Load	data --	----> <----	-- acknowledge
seNd	-- acknowledge	<---- ---->	data --

In this illustration, data/acknowledge message pairs are continually sent until the desired amount of data is transmitted successfully, or a fatal error or user abort occurs.

7.2 DATA ACKNOWLEDGEMENT MESSAGES

Of the three types of Load/seNd messages, data-block acknowledgements are the simplest. A data-block acknowledgement must be sent each time a data message is received. It consists of exactly one of three characters--0, 7, or 9--followed by a RETURN. The characters have the following meaning:

0 Data block received with valid checksums.

7 Transmission error, please resend last block

9 Bad load address or error message received, abort process.

Thus, the sender (either the host system or Z-SCAN) simply places a data message on the RS-232 link and after one echoed line, receives one of these three characters. If the host does not echo the input characters, it must at least precede any acknowledgement with RETURN or RETURN and linefeed. This happens if, for example, the host is half duplex.

If a 7 is persistently returned to the host system because of checksum errors, the sender must decide when to stop trying to send a message. In the Z-SCAN seNd command, this occurs after a total of 10 trys. A custom host Load command can use some other value, if desired. In any case, once the sender stops re-sending data, it notifies the receiver that it is giving up the whole communication effort by sending one of the error messages (// RETURN) described below.

Z-SCAN Load/seNd acknowledgement logic allows for don't-care regions in acknowledgement lines, such as:

    xxxx 0 xxxx RETURN xxxx ...

Here, Z-SCAN's seNd command examines all characters returning from the host after a data message has been sent. It throws out all data until a 0, 7 or 9 is found. The data preceding the acknowledgement character must not contain 0, 7 or 9. It also throws away the rest of the line. Z-SCAN does not currently use these regions, so the host receives 0, 7 or 9 and RETURN in unadulterated form, even if it chooses to send "dirty" acknowledgements.

7.3 DATA TRANSMISSION MESSAGES

All memory or other numerical data sent over the host/Z-SCAN link is formatted as ASCII characters that represent each data byte as two hexadecimal digits. Error-message text can also be sent. Since each transmission is terminated by a RETURN, it appears to the host as if the source were simply a standard ASCII terminal. This formatting allows Z-SCAN to be connected to a standard terminal port on the host, which, in turn, makes possible its transparent (user-to-host) mode of operation.

Data and error messages begin with a slash (/) to distinguish them from acknowledgements. The basic format for a message is:

    / printable_ASCII_characters RETURN

7.3.1 Error Messages

To distinguish error messages from data blocks, an error message has the form:

    / / error_text RETURN

where the degenerate case, //, is used by the Z-SCAN seNd command to respond to the last 7 in a failed sequence of sending retries (see Section 7.2). Z-SCAN thus signals the host to abort the entire communication process. When sent by the host, error messages cause the Z-SCAN Load and seNd commands to abort, but any error text is first copied to the user's terminal to indicate that an error has occurred.

The error-message facility thus allows the sender to force the receiver to abort, while optionally providing an informative message of up to 40 characters. Error messages can only be sent in place of a data message--that is, only when a data message might otherwise by sent. The appropriate acknowledgement of any error message is 9. Z-SCAN supplies but does not expect such closure, since all error messages from Z-SCAN correspond to abort conditions.

7.3.2 Data Messages

A data message begins with /, ends with RETURN, and contains printable characters that are translated from hexadecimal numerals into memory bytes, checksum bytes, or address words. The sender's message encoding proceeds byte by byte, nibble-by-nibble, building a string of hexadecimal text digits 0 through F. The receiver must then reverse the translation to obtain binary byte values. The basic format of a data message is:

	/	address	byte_count	checksum1	data	checksum2	RETURN
char:	1	2 3 4 5	6 7	8 9	10...	2xbyte_count+10	2xbyte_count+11

where the items sent are defined as:

address — 4-digit hexadecimal (0-F) representation of a 16-bit destination address for data, highest-order digit first. If the byte count is zero, it corresponds to a program file's entry address and must be even.

byte_count — 2-digit hexadecimal, 8-bit count of the physical memory bytes to be constructed from data numerals. 00 indicates the final record in the transmission and that the program's entry address was sent in this message. For the Z-SCAN seNd command, the maximum count is 30 (decimal). For Load, the byte_count is limited by the Z-SCAN input buffer size.

checksum1 — 2-digit hexadecimal, 8-bit sum of address and byte-count values, nibble-by-nibble, done prior to conversion to ASCII 0 to F. Data nibbles are simply added successively into an 8-bit register to produce the checksum, which is then translated into the two numerals sent.

data — 2-digit hexadecimal, 8-bit memory bytes, starting with numerals for the byte at the address sent--30 bytes (60 numerals) maximum length for seNd.

checksum2 — 2-digit hexadecimal, 8-bit sum of data-byte nibbles with the overflow ignored. The ability of this checksum to trap errors decreases once the sum of all data nibbles exceeds 255, which also depends on the byte_count.

During reception of data messages, all ASCII control characters (codes below 32 decimal) are ignored except for RETURN. Furthermore, all characters following RETURN up to the beginning of the next message (/) are ignored. In combination with the fact that only sufficient numerals are used to exhaust the byte count and checksum2, identification of don't-care regions within a stream of data messages is possible:

xxxx / address... data checksum2 xxxx RETURN xxxx /...

The xxxx regions could conceivably be used to transmit any information that does not conflict with the basic format used. As with the acknowledgement message, Z-SCAN does not use this feature, though a custom host program can do so, with the limitation that during Load operations,any host should not send more message characters than can fit in Z-SCAN's input buffer (currently 128).

Note that the only characters that have meaning in data messages are /, RETURN, 0 to 9 and A to F, and the overall transmission format corresponds to one originally developed by Tektronix. Each data message must be acknowledged before the next can be sent.

7.4 COMMAND TRANSMISSION

Both the Load and seNd commands initiated in Z-SCAN by the user at the terminal rely on the host to run the appropriate program (LOAD or SEND) to complete the communications link and to transfer the desired data. To begin the process, Z-SCAN sends a command line to the host. This line includes a file name taken from the corresponding field on the Load or seNd command subscreen. The seNd command line also includes the start address, end address and entry point for the file, encoded as four-digit hex numbers. The host program may choose to ignore this information since it is also contained in the data records sent to the host by Z-SCAN. Thus, the host operating system must be able to receive either of two commands from Z-SCAN:

B;LOAD filename RETURN

or:

B;SEND filename start_address end_address entry_point RETURN

The host operating system should then run the appropriate program, passing the parameters, which may be followed by trailing blanks, to the program for parsing.

The "B;" prefix instructs the Zilog RIO operating system to cease verbose mode re-echoing of commands after RETURN. If no such feature exists in the host being used, these two characters must be ignored. In any case, Z-SCAN ignores echoing of the command line and the next line too if it begins with B; and awaits activation of the corresponding program.

The user can enter BREAK to abort command transmission. If this is not done, Z-SCAN does one of three things once the command line has been sent to the host and any echo(s) have been ignored:

host failure 0, 7, or 9 RETURN not received: pass what was received to user's terminal as host error diagnostic and abort.

file unopened 7 or 9 RETURN received: desired file not found or can't be opened: display error message and abort.

proceed 0 received: LOAD/SEND running and ready to send or receive data.

Note that in host failure, the host's response is passed directly to the user's terminal regardless of its structure. The transmission or reception of data by the host begins as soon as the host LOAD/SEND program issues its initial 0 response.

Once Z-SCAN has ignored command echoing, any text beginning with B is skipped until the next RETURN:

```
B;SEND...RETURN... B xxxx RETURN xxxx 0 xxxx RETURN
```

This feature is not used by Zilog's host software, though it may be implemented by a custom host.

7.5 USER ABORT

The Z-SCAN software monitors terminal activity and BREAK keystrokes during Load/seNd activities. In this way it can abort data transfers entirely. Whenever an abort occurs after successful command transmission, the host is sent either 9 or //, depending on the context (Z-SCAN receiving or sending, respectively), and is expected to abort the LOAD/SEND program immediately. The Z-SCAN never waits for acknowledgement of the degenerate error message (//) and terminates seNd activity at once. BREAK aborts Z-SCAN Load/seNd regardless of the host program's current state, functional or not. During Load activity, the 9 is sent just after the last acknowledge, allowing the host to send an extra data block that is acknowledged by the waiting 9 but is never received by the already aborted Z-SCAN.

7.6 DETAILED TRANSMISSION PROTOCOL

The following sections describe the assumptions of the sequential details of host/Z-SCAN Load/seNd transactions and data transmission currently made by the Z-SCAN monitor.

Load/seNd communication is always initiated by Z-SCAN because it responds to commands from the user terminal. Whether Z-SCAN issues a Load or seNd command, the initial sequence of messages is the same:

Z-SCAN Sends	**Host Sends**	
B; command RETURN		
	normal_echo RETURN line feed	(of command)
	verbose_echo RETURN line feed	(possible)
	0 RETURN	(file found and ready)

where the command sent by Z-SCAN always begins with either B;LOAD or B;SEND, and where the host operating system (RIO on Zilog systems) determines the nature and existence of command echoing. Z-SCAN assumes that there is one echo line, and that if B starts the second line received, the host was in verbose mode and acknowledgement will be found in the next line (0...). Thus, the first one or two lines received from the host are completely ignored by Z-SCAN. For custom hosts or host programs, this means that if no echoing occurs (for example, if normal, full-duplex echoing is turned off), all messages sent to Z-SCAN must be preceded by RETURN, since Z-SCAN assumes that each message sent to the host is echoed. The acknowledgement above would then be RETURN 0 RETURN.

Just before sending the host command, Z-SCAN also checks to see if the user has entered BREAK. If so, Z-SCAN sends 9 RETURN to the host, as it does during data transmission, but does not send the command. Typically, the host operating system does not understand this meaningless message and, in turn, generates a meaningless error message.

Note that every message line sent to the host by Z-SCAN is echoed by the host if echoing is on. For simplicity, this is not shown in subsequent examples except for command transmission. Z-SCAN automatically skips one received line for each line it sends.

It is, however, possible that communication is not successfully established. In fact, once Z-SCAN has sent the Load or seNd command line to the host and received and ignored normal and verbose echoes, three possibilities exist:

1. The command contains an error that prevents the operating system from running the load or send program. In this case, a 0 is not transmitted, but any response or error message the operating system chooses is sent. Whatever is returned is passed directly to the user's terminal to indicate the failure of the command.

2. The command is correct, and the load or send program runs but cannot find or open the desired file. In this case, either a 7 or a 9 (no preference) is returned by the host program to Z-SCAN, generating an error message on the user's terminal.

3. The command is correct, the LOAD/SEND program runs, and the desired file is opened. In this case, a 0 is returned to signify that data transmission will (Load), or may begin (seNd).

Samples of successful command transmission appear in the data transmission examples in subsequent sections. Listed here are some failure possibilities:

Z-SCAN Sends	Host Sends
1. B;LOAD X RETURN	CAN'T FIND PROGRAM: LOAD RETURN line feed
2. B;LOAD X RETURN	7 RETURN (LOAD can't open X)
3. 9 RETURN	ILLEGAL FILENAME: 9 RETURN line feed (user entered BREAK)

In the first case, Z-SCAN copies the host message to the user's terminal, while in the second case, Z-SCAN generates its file-error message. In all cases, Z-SCAN aborts the Load/seNd activity, and in the second case, the host program does the same.

7.6.1 Load Protocol

After command transmission to the host, the message-exchange protocol for down-loading a program into Z-SCAN is:

Host Message	Path	Z-SCAN Message	Notes
data message	-->	--	Skip until /, then wait for RETURN. Check first message character for error flag (/), validate checksums, and load data into memory.
--	<--	0	Acknowledge valid data so next block can be sent. Done if final block. Check for BREAK by user.
--	<--	7	Request re-sending of block due to checksum error. Check for BREAK by user.
--	<--	9	Acknowledge error message or indicate that BREAK was typed, and transmission must be aborted. Done.

Possible message sequences between the Z-SCAN and the host might be:

Z-SCAN Sends	Host Sends
B;LOAD X RETURN	
	B;LOAD X RETURN line feed (normal echo)
	0 RETURN (LOAD acknowledgement)
	/ data1 RETURN
0 RETURN	
	/ data2 RETURN
7 RETURN	
	/ data2 RETURN (data re-sent)
0 RETURN	
	/ data3 RETURN
. . .	
	. . .
	/ dataend RETURN (0-count data message with entry point address)
0 RETURN	

Here, both Z-SCAN and the host system reach normal completion after the final acknowledge.

An error transmission might be:

Z-SCAN Sends	Host Sends	
B;LOAD X RETURN		
	B;LOAD X RETURN line feed	(from host OS)
	0 RETURN	(from LOAD program)
	/ data1 RETURN	
7 RETURN		(checksum bad)
	/ data1 RETURN	
.		
.		
.		
	.	
	.	
	.	
7 RETURN		
	// error RETURN	(retries exhausted)
9 RETURN		

Here, both Z-SCAN and the host program abort the transmission attempt after error-message acknowledgement.

An example of a user-induced abort during data transmission might be:

Z-SCAN Sends	Host Sends	
B;LOAD X RETURN		
	B;LOAD X RETURN line feed	
	0 RETURN	
	/ data1 RETURN	(user enters BREAK)
0 RETURN 9 RETURN		(host sees 0)
	/ data2 RETURN	(sends next data)
		(then sees 9)

The BREAK is entered during data1 transmission, but Z-SCAN acknowledges the data anyway and then immediately sends its abort signal. The host, operating line by line on Z-SCAN messages, sends data2, sees the abort acknowledgement waiting in its input buffer, and then terminates. Z-SCAN terminated earlier, upon sending the 9.

7.6.2 Send Protocol

In the seNd protocol, as in Load, command transmission takes place first. Subsequent protocol elements are:

Host Message	Path	Z-SCAN Message	Notes
--	<--	data message	See Section 7.7, Message Syntax
0 RETURN	-->	--	Acknowledge valid data
7 RETURN	-->	--	Request to re-send data
		:	
--	<--	//	Error - re-send count exhausted
9 RETURN	-->	--	Acknowledge error and abort

A message sequence might look like:

Z-SCAN Sends	Host Sends
B;SEND W start end entry RETURN	
	B;SEND W RETURN line feed 0 RETURN
/ data1 RETURN	
	0 RETURN
/ data2 RETURN	
	. . .
. . .	
/ dataend RETURN	
	0 RETURN

As a result of this sequence, the Z-SCAN command and the host program simply terminate.

An error transmission might be:

Z-SCAN Sends	Host Sends
B;SEND W start end entry RETURN	
	B;SEND W RETURN line feed
	0 RETURN
/ data1 RETURN	
	7 RETURN
.	
.	
.	
	.
	.
	.
// RETURN	
	9 RETURN

Again both programs terminate, but in failure.

A user abort might take place before command transmission (see Section 7.5, User Abort) or during data transmission:

Z-SCAN Sends	Host Sends
B;SEND W start end entry RETURN	
	B;SEND W start end entry RETURN line feed
	0
/ data1 RETURN	
	0 (user enters BREAK)
//	
	9

Here, BREAK was struck after Z-SCAN began sending data1 and the error-abort message was sent instead of data2.

7.7 MESSAGE SYNTAX

The three types of messages symmetrically exchanged over the host-Z-SCAN link can be described in Backus-Nauer form:

```
<Z-SCAN message>     ::=<message string> RETURN
<message string>     ::=<acknowledgement> |<data string> |<end string>|
                          <error string>
<acknowledgement>    ::="0" | "7" | "9"
<data string>        ::="/" <address> <count> <sum> <data> <sum>
<end string>         ::="/" <address> "00" <sum>
<address>            ::=<digit> <digit> <digit> <digit>
<count>              ::=<digit> <digit>
<sum>                ::=<digit> <digit>
<data>               ::=<digit> | <data> <digit>
<error string>       ::="//"   <error string> <ASCII char>
<digit>              ::="0" | "1" | ... | "9"
                         "A" | "B" | ... | "F"
```

A valid data message must have twice the value of <count> data digits in the <data> string. The end string is a special case of the data string and is the last message to be sent by the sender. In this case, <address> is the entry (starting) point of the procedure (program) file being transferred and must be even for Z8000 programs.

The two checksum (<sum>) values are computed by adding all nibbles of the data they are to check. This is done in an 8-bit accumulator, which generally overflows for the <data> portion of the transmission. Note that the addition occurs before the data is translated into ASCII numerals. Thus the host SEND program must translate the received Z-SCAN data before computing the checksum for comparison with the value of either <sum>.

For any of the three message types, Z-SCAN ignores control characters (except RETURN) and any characters between those needed to decode the message and the end of the line sent (RETURN). Z-SCAN also ignores characters between the last RETURN and the next / when receiving data/error messages.

Command transmission syntax is defined by Z-SCAN, and because it describes unidirectional communication,it is of interest only to the host:

```
<Z-SCAN command>        ::="B;" <name> sp <body> sp* RETURN
<name>                  ::="LOAD" | "SEND"
<body>                  ::=<filename> [<parameter>]*
<parameter>             ::=start_address | end_address | entry_point
```

where sp denotes ASCII space, filename is the name entered by the user, and * indicates an item that occurs zero or more times.

7.8 HOST PROGRAM CONTROL FLOW

The following outlines the sequential behavior of a host LOAD or SEND program that successfully interacts with Z-SCAN. It is an alternative way of stating the protocol definition already discussed from the host program's point of view. An example of such a program, the LOAD utility for Zilog's Z80 RIO operating system, is given in Appendix A.

7.8.1 Load Program

The host LOAD program flow is shown in Figure 7-1.

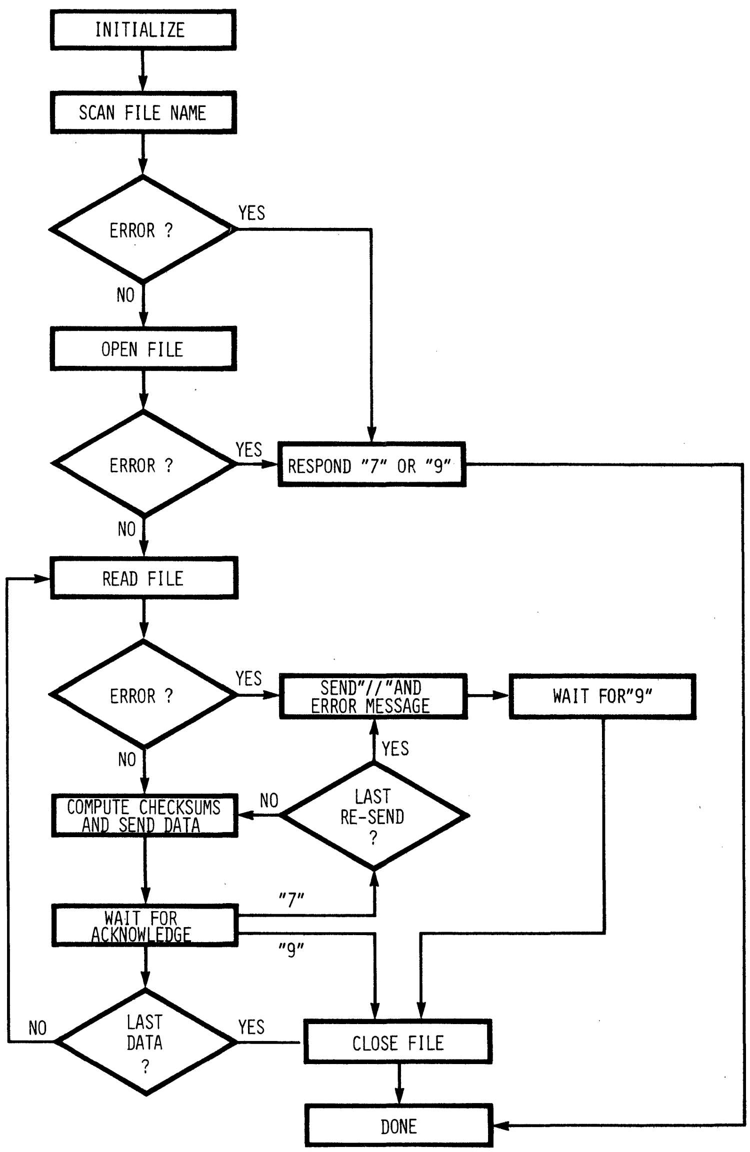


Figure 7-1. Flowchart for LOAD Program

7.8.2 seNd Program

The host SEND program flow is shown in Figure 7-2.

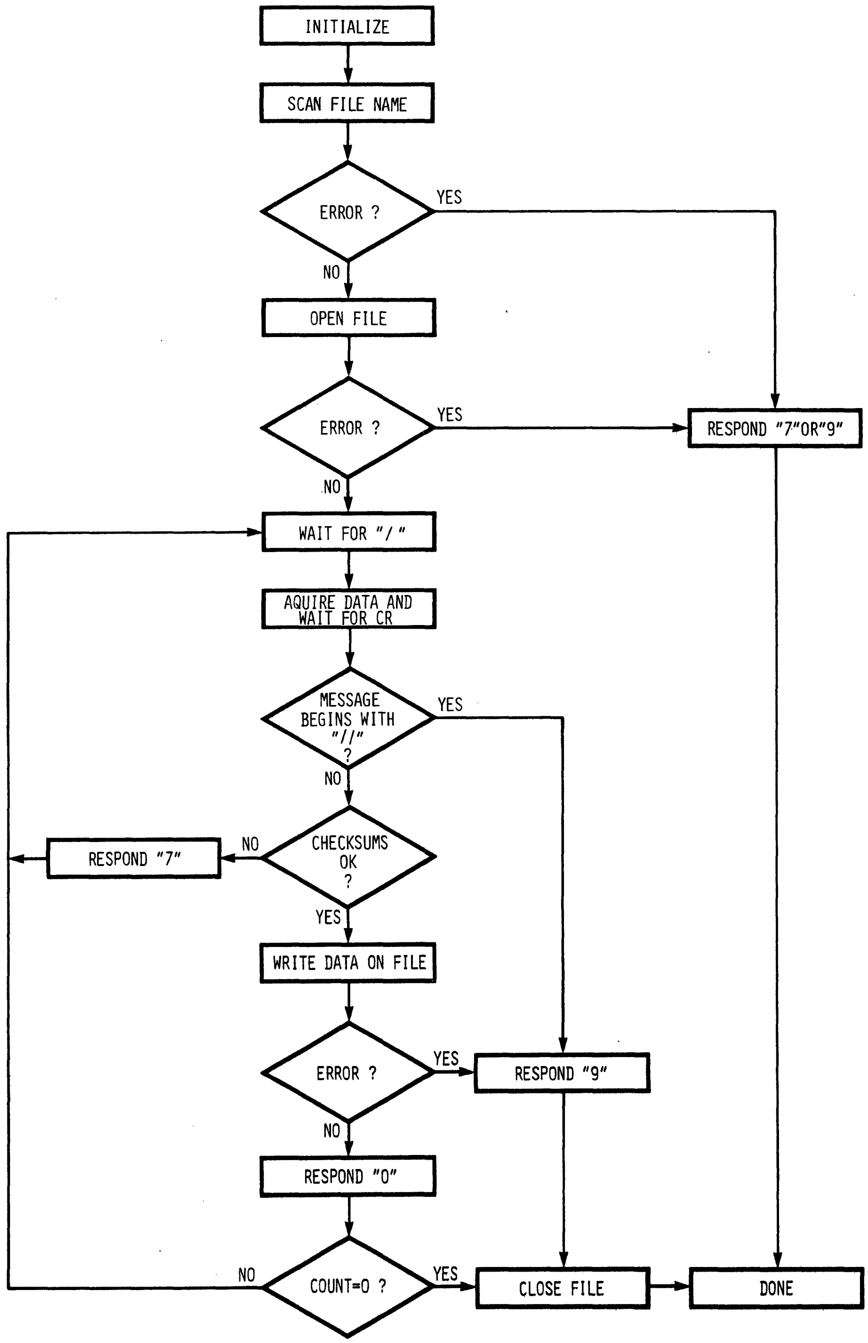


Figure 7-2. Flowchart for SEND Program

APPENDIX A

TERMINALS SUPPORTED BY Z-SCAN

APPENDIX A

Terminals Supported by Z-SCAN

A.1 INTRODUCTION

The two-dimensional user interface of the Z-SCAN monitor software requires a CRT terminal with a 24-line by 80-column display. In addition, the monitor software requires the following functions:

- Clear screen
- Clear to end of line
- Position cursor
- Cursor control keys (move cursor up, down, left or right)

Most terminals offer these features, but the character sequences that distinguish them vary according to the terminal manufacturer.

The monitor software supports nine distinct CRT control protocols, allowing Z-SCAN to be used in conjunction with terminals from many different manufacturers. The user must select the terminal type by entering a hex digit during the initialization of the software. See Section 3.5 for further details. Table A-1 summarizes the terminal types that are supported.

A.2 TERMINAL DETAILS

The Z-SCAN can be used with a terminal not listed in Table A-1 provided that the terminal is compatible with one of those which is listed. The following paragraphs detail the control sequences generated by the monitor for each supported control protocol. Table A-2 lists the symbols used to describe the protocols. Commas and spaces that appear in the descriptions are simply separators: they are not part of the transmitted sequences.

Table A-1. Terminals Supported by the Z-SCAN Monitor

Selection Digit	Manufacturer	Model
0	Lear Siegler Televideo Zentec Soroc	ADM 31 TVI 912 TVI 920 Zephyr IQ 120 IQ 135
1	ADDS	Regent series
2	Beehive	Bee 100 Bee 107 Micro-B 1
3	DEC	VT52
4	DEC (any)	VT100 ANSI A3.64 or ISO DP 6429 compatible
5	General Terminals Inc.	I-200 I-400
6	Hazeltine	1420 1500 Exec 80
7	Hewlett Packard	2620 2640
8	IBM	3101

Table A-2. Control Sequence Protocol Symbols

Symbol Type	Representation	Notes
escape (hex 1B)	esc	
other control chars. (codes 00 to 1F)	^ char	This code is that transmitted when the control key and the given character key are pressed together.
tilde (hex 7E)	tilde	This character, ~, does not display on some terminals.
cursor row (binary)	rb	This is a single character giving the row number, 1-24 decimal, offset by 31 decimal unless otherwise stated.
cursor column (binary)	cb	This is a single character giving the column number, 1-80 decimal, offset by 31 decimal unless otherwide stated.
cursor row (decimal)	rd	This is the row number sent as two printable characters. The range is 00 to 23 for HP terminals and 01 to 24 for the ANSI compatible DEC VT100.
cursor column (decimal)	cd	This is the column number sent as two printable characters. The range is 00 to 79 for HP terminals and 01 to 80 for the ANSI compatible DEC VT100.
all others	as displayed	

Many terminals offer options that can be selected by entering commands at the keyboard when the terminal is in a special set-up mode, or by setting concealed switches. The Z-SCAN monitor is not able to function correctly when certain options are selected. The option settings required on each major terminal type are listed. The monitor is not sensitive to the setting of any option that is not mentioned. Refer to the manufacturer's documentation if further information is required about any particular terminal.

Special considerations apply when certain terminals are used with Z-SCAN. For example, some low-cost terminals do not have cursor movement keys, so the user must enter control characters manually in order to acheive the desired effect. Such considerations are mentioned when they apply.

A.2.1 Lear Sielger ADM 31 and Soroc Terminals

Table A-3 lists the control sequences used with the ADM 31 and compatible terminals:

Table A-3. ADM 31 Control Sequences

Clear Screen Sequence	Clear Line Sequence	Move Cursor Sequence	Cursor Up	Cursor Down	Cursor Left	Cursor Right
^ ^ ,esc, Y	esc, T	esc, =, rb, cb	^ K	^ J	^ L	^ H

The following internal switch settings are required when an ADM-31 is used in conjunction with Z-SCAN. It should not be necessary to check these settings unless problems are experienced during use:

Table A-4. ADM 31 Option Settings

Switch Bank	Switch	Setting	Function
1	1	On	Break enable
	5	Off	8 bits, no parity
	7	Off	Conversational mode
	8	On	Full duplex
3	8	On	Disable polling
4	4	Off	Current loop disable
	6	On	Display nulls as nulls

The following ADM 31 baud rates may be used in conjunction with the Z-SCAN monitor. Note that, because some of the features of the terminal do not function at 19200 baud, this speed should not be used. The baud rate selection switch is at the left rear of the terminal.

Table A-5. ADM 31 Baud Rates Supported by Z-SCAN

Baud Rate	Switch Setting	Baud rate	Switch Setting
75	1	1200	7
110	2	1800	8
134.5	3	2400	10
150	4	4800	12
300	5	9600	14
600	6		

The sequences used to control the ADM 31 are compatible with the Soroc IQ 120 and IQ 135 terminals. Z-SCAN supports all baud rates available on these terminals except 1000, 2000, 3600 and 7200 baud. Users of the IQ 135 should be aware that either of the SHIFT keys must be pressed at the same time as BREAK in order for a break to be transmitted. Switch bank K8 switch, 8 must be in the up position. Parity should be disabled when a Soroc terminal is used with Z-SCAN.

A.2.2 ADDS (Applied Digital Data Systems) Regent Series

The Z-SCAN monitor supports the protocol used by the Regent 40 and other compatible terminals in the ADDS range. Table A-6 lists the control sequences used.

Table A-6. Regent 40 Control Sequences

Clear Screen Sequence	Clear Line Sequence	Move Cursor Sequence	Cursor Up	Cursor Down	Cursor Left	Cursor Right
^ L	esc, K	esc, Y, rb, cb	^ Z	^ J	^ F	^ U

Because of the wide variety of terminals in the Regent series, it is not possible to list the options and baud rates required for compatibility with the Z-SCAN monitor in each case. If difficulties are experienced, refer to the terminal documentation and check that each of the following statements is true:

- Line mode is full duplex.
- Parity bit is spacing.
- Baud rate is supported by Z-SCAN (See Table 6-7).
- Termination character is CR.
- Auto scroll is on.
- Interface is EIA (not current loop).
- Z-SCAN is connected to EAI/CURRENT LOOP connector.

A.2.3 Beehive Terminals

The protocol used by the Bee 100, Bee 107 and Micro B 1 terminals is given in Table A-7.

Table A-7. Beehive Control Sequences

Clear Screen Sequence	Clear Line Sequence	Move Cursor Sequence	Cursor Up	Cursor Down	Cursor Left	Cursor Right
esc,E	esc, K	esc, F, rb, cb	esc, A	esc, B	esc, C	esc, D
alternative sequences - see below -->			^ K	^ J	^ L	^ H

The cursor up, down, left and right keys on the Bee 100 operate locally: they move the cursor on the screen but do not transmit codes on the serial link. They are thus unsuitable for use with the Z-SCAN monitor. The user must enter the sequences manually. The single control characters listed above may be used to save keystrokes. As a further alternative, Beehive's service organization can arrange for incorporation of the field change that transforms a Bee 100 into a Bee 107. The Bee 107 cursor control keys transmit escape sequences on the serial link. Cursor control sequence transmission is a switch-selectable option on the Micro B1. The option should be enabled.

A.2.4 DEC (Digital Equipment Corporation) VT52

Table A-8 lists the control sequences used with the DEC VT 52 terminal.

Table A-8. VT 52 Control Sequences

Clear Screen Sequence	Clear Line Sequence	Move Cursor Sequence	Cursor Up	Cursor Down	Cursor Left	Cursor Right
esc, H, esc, J	esc, K	esc, Y, rb, cb	esc, A	esc, B	esc, C	esc, D

A.2.5 DEC (Digital Equipment Corporation) VT 100

The VT 100 terminal control sequences are compatible with those specified by ANSI standard A 3.64 and ISO standard DP 6429 and so can be used with any terminal that supports either standard. The standards allow some flexibility in implementation, so users should check that any alternative terminal is compatible with the VT 100 for the small number of functions required. The sequences used by the Z-SCAN monitor are shown in Table A-9. Note that there are three valid sequences, each starting with esc, [, for each of the four cursor movement keys:

Table A-9. VT 100 Control Sequences

Clear Screen Sequence	Clear Line Sequence	Move Cursor Sequence	Cursor Up	Cursor Down	Cursor Left	Cursor Right
esc, [, 2, J	esc, [, K	esc, [, rd, ;, cd, H	esc, [, A or 0, A or 1, A	esc, [, B or 0, B or 1, B	esc, [, C or 0, C or 1, C	esc, [, D or 0, D or 1, D

The VT 100 allows 80 or 132 columns per line. The Z-SCAN screens are designed for 80 column presentation. This format should be selected on the set-up A screen. The screens will not display correctly in 132 column format unless the Advanced Video option is installed in the terminal. Most VT 100 options are selected on the set-up B screen. Four settings are important to the Z-SCAN monitor:

- ANSI mode must be on (unless the terminal is to be used in VT 52 mode - see A.4.2).
- Auto XON XOFF must be enabled if the baud rate is 9600 or 19200, or if smooth scrolling is selected.
- New line must be disabled.
- Parity must be off.

In addition, the transmit speed must be the same as the receive speed. The Z-SCAN monitor supports all VT 100 baud rates except 2000 and 3600 baud.

A.2.6 GTI (General Terminals Inc.) I-200 etc.

The control sequences used by GTI (formerly Infoton) terminals are listed in Table A-10.

Table A-10. GTI Control Sequences

Clear Screen Sequence	Clear Line Sequence	Move Cursor Sequence	Cursor Up	Cursor Down	Cursor Left	Cursor Right
^L	^K	^W, cb, rb	^\	^]	^ Y	^ H
alternative sequences - see below -->			^K	^ J	^ L	^H

Some GTI models do not have cursor movement keys. The operator must enter the control characters listed in order to move the cursor. The alternative keys listed above can also be used.

The following comments apply to the I 200 terminal. Refer to the terminal documentation for information on other models.

- The indicators in the Caps Only, Page and Line keys at the top right of the keyboard should all be illuminated for correct operation with Z-SCAN.
- The two rear panel parity switches, the Page/Bottom line entry and Int clk/Ext clk switches should be in the off position.
- Any baud rate except 3600 or 7200 can be selected.

A.2.7 Hazeltine Terminals

Hazeltine terminals, especially more recent models, support a variety of protocols. Table A-11 describes the basic sequences implemented by all Hazeltine terminals. Note that the binary column code used is the actual binary column number unless the column number is in the range 1-31 decimal, in which case an offset of 95 decimal is added.

Table A-11. Hazeltine Control Sequences

Clear Screen Sequence	Clear Line Sequence	Move Cursor Sequence	Cursor Up	Cursor Down	Cursor Left	Cursor Right
tilde, ^\	tilde,^O	tilde, ^Q, cb, rb	tilde, L	tilde, K	^ P	^ H
alternative sequences - see below -->			^ K	^ J	^ L	^ H

Some Hazeltine terminals do not have cursor movement keys. The user must enter the correct sequences manually if Z-SCAN is used in conjunction with these models. Alternatively, the second group of control codes shown above can be used.

In order to work correctly with the Z-SCAN monitor, the automatic line feed and parity options should be disabled on Hazeltine terminals. Where a user option allows either escape or tilde to be used in control sequences, tilde should be selected. Where a choice of protocols exists, select Hazeltine protocol.

The Hazeltine current loop option uses the same connector as the RS 232 interface and can intefere with data sent to the terminal by Z-SCAN. If problems are experienced when a 25-way cable is used for the RS 232 link, substitute a cable that connects only pins 2, 3, 4, 5, 7 and 20.

A.2.8 Hewlett Packard Terminals

Hewlett Packard 2620 series and 2640 series terminals use the control sequences listed in Table A-12.

Table A-12. Hewlett Packard Control Sequences

Clear Screen Sequence	Clear Line Sequence	Move Cursor Sequence	Cursor Up	Cursor Down	Cursor Left	Cursor Right
esc, J	esc, K	esc, &, a, rd, Y, cd, C	esc, A	esc, B	esc, C	esc, D

The sequences required by some early models of the 2640 may differ. These terminals are not supported by the Z-SCAN monitor. Users experiencing difficulties should contact a Hewlett Packard sales office.

A user option on HP terminals determines whether the cursor movement keys act locally or cause escape sequences to be transmitted. The Z-SCAN monitor requires that escape sequences are transmitted. See the terminal manual for further details. Parity must be disabled when a Hewlett Packard terminal is used in conjunction with Z-SCAN.

A.2.9 IBM (International Business Machines) 3101 Terminal

Table A-13 lists the control sequences used with the IBM 3101 asynchronous terminal:

Table A-13. IBM 3101 Control Sequences

Clear Screen Sequence	Clear Line Sequence	Move Cursor Sequence	Cursor Up	Cursor Down	Cursor Left	Cursor Right
esc, K	esc, I	esc, Y, rb, cb	esc, A	esc, B	esc, C	esc, D

The following user options should be selected when this terminal is used in conjunction with Z-SCAN:

- Auto LF off
- Terminator is CR only
- SCRL on
- Parity off

The transmit and receive baud rates must be the same and must correspond to one of the rates supported by Z-SCAN.

The keyboard of the 3101 differs from those of most terminals in several respects:

- The ALT and R keys must be pressed together to generate the control and R required to restore the previous contents of a variable field.
- The SHIFT lock key is like that of a typewriter in that it shifts all four rows of keys.
- The < and > characters are generated by a single key at the lower left of the keyboard.

APPENDIX B

TUTORIAL TEST PROGRAM

```
LOC     OBJ CODE     STMT SOURCE STATEMENT

EXAMNSG MODULE
!
MODULE NAME

    EXAMNSG


ASSOCIATED MODULES

    None


FUNCTION

    To provide a demonstration of Z-SCAN's breakpoint
    and  Target  mode functions.   This is  the  only
    function of the program.


TARGET PROCESSOR

    Z8002.   Since  this  program contains  trap  and
    reset  vectors,  it  cannot be run on the  Z8001,
    even  if  that processor is run  in  nonsegmented
    mode.  See chapter 7 of the "Z8000 CPU  Technical
    Manual"  for  futher  details.   The  following
    assembler  directive  is not  strictly  necessary
    because Z8000ASM assembles in non-segmented  mode
    by default.
!
    $NONSEGMENTED                  ! Assemble for Z8002 !
!

REQUIRED HARDWARE

    Z-SCAN  8000  with  Z8002  processor.   No  other
    hardware  is required or assumed when the program
    is running.
```

```
PROGRAM SECTIONS USED

    All  code and data is directed to a single  named
    program section, EXAMNSG_P.  This allows code and
    data  areas  to appear in memory in  exactly  the
    order that they are defined in the  program.  By
    default, the assembler would direct the data into
    a  second section,  EXAMNSG_D.  The defaults are
    useful  in applications where all procedure  code
    must  be assigned to one memory area (often  ROM)
    and all data to another.  IMAGER is used to make
    such assignments.  This particular program needs
    only a simple IMAGER command line (see tutorial).
!
    $SECTION     EXAMNSG_P          ! Make imaging easy   !
    $REL         %0000              ! Start at loc. 0000  !

!
RESET PROGRAM STATUS

    Following a reset,  a Flag and Control Word (FCW)
    value  of %4000 is loaded to make  the  processor
    start in system mode with interrupts and extended
    processing disabled.  The Program Counter (PC) is
    loaded     with    the    start    address    of    the
    initialization routine,  INIT.  This information
    is automatically fetched from locations %0002 and
    %0004  respectively.  See  section  7.4  of  the
    "Z8000  CPU Technical Manual" for  more  details.
    The  reset status is defined with an  initialized
    array  declaration which has the effect  of
    reserving  two  words  of  memory.  The  $REL
    directive  fixes  the data at locations 0002  and
    0004.
!
INTERNAL                            ! Private data area   !
NEW_STATUS_AREA:                    ! See note below      !
    $REL         %0002
0002 4000   002A'
    RESET        ARRAY   [ 2 WORD ] := [ %4000, INIT ]

```

```
                    !
                    NEW PROGRAM STATUS AREA

                        The  new program status area defines the  program
                        statuses which will be loaded following each type
                        of interrupt or trap, as described in section 7.6
                        of  the  "Z8000  CPU  Technical  Manual".    This
                        example uses only three of its many entries.   It
                        can   start on  any  256 byte boundary in  system
                        code  address  space.   In most  applications  it
                        could  not  start at location  zero  because  the
                        reset   program  status  overlaps  the   extended
                        instruction  trap  vector,  which is not used  in
                        this     example     program.     The  identifier
                        NEW_STATUS_AREA,  defined  above,  points to  the
                        start of the area.

                        The  destination  of the  privileged  instruction
                        trap  is  the  same procedure  which  is  entered
                        following a reset.  Bit two, the half carry flag,
                        of the FCW is set in the first case and clear  in
                        the   second so that the user may easily identify
                        why  the  routine  was  entered.   Two  different
                        System  Call instruction opcodes are used  for  a
                        similar  reason  - examination of the top of  the
                        system stack shows why BREAKER was entered.

                        The  last  vector  defined  handles  Non-Maskable
                        Interrupts.  Like the system call and  privileged
                        instruction traps, an NMI forces the CPU into the
                        INIT procedure.  The decimal adjust flag is set.
                    !

                    INTERNAL

                        $REL    %0008                 ! Privileged instr.   !
0008 4004   002A'       PRIV_VECTOR ARRAY [ 2 WORD ] := [ %4004, INIT ]

                        $REL    %000C                 ! System Call         !
000C 4000   003C'       SC_VECTOR   ARRAY [ 2 WORD ] := [ %4000, BREAKER ]

                        $REL    %0014                 ! Non-Maskable Int.   !
0014 4008   002A'       NMI_VECTOR  ARRAY [ 2 WORD ] := [ %4008, INIT ]

```

```
                       125 !
                       126 DATA AREA
                       127
                       128     The  program uses just two words of static  data:
                       129     PASS  is incremented by one for each pass through
                       130     the procedure BREAKER (see below).  On each loop,
                       131     the  previous value of PASS is copied into  LAST.
                       132     Both variables are zero when the program is first
                       133     loaded,   but  as they are not intialized by  the
                       134     program's initialization routine,  their contents
                       135     must  be considered to be undefined  following  a
                       136     reset.  Their position in memory is not critical.
                       137 !
                       138
                       139 INTERNAL                           ! Private data area   !
                       140
0018 0000  0000        141     PASS, LAST  WORD := 0
                       142
                       143 !
                       144 STACK AREA
                       145
                       146     The   Z8000  requires  two  call-return   linkage
                       147     stacks;  one  for  system mode operation and  the
                       148     other for normal mode.  For more information, see
                       149     section    4.3.3   of  the  "Z8000  CPU   Technical
                       150     Manual".   The  procedure  INIT sets up  the  two
                       151     stack pointers.   The normal stack is used by the
                       152     instructions  on  lines 368 through  370  in  the
                       153     procedure BREAKER.  These instructions affect the
                       154     top  word  of the stack area  only.   The  system
                       155     stack  is  used  by  the  same  instructions  and
                       156     additionally by the System Call (SC) instructions
                       157     on  lines 230 and 372.   This instruction  stacks
                       158     the  old  Program   Counter,  the  old  Flag  and
                       159     Control Word and an identifier, giving a total of
                       160     three  words.   The identifier is the instruction
                       161     itself.   For  more details of this  instruction,
                       162     which  is  used rather unconventionally  in  this
                       163     program,   see  section  6  of  the  "Z8000   CPU
                       164     Technical Manual".  The position of the stacks in
                       165     memory is not important.
                       166 !
                       167
                       168 INTERNAL                           ! Private data area   !
                       169
001C 0000  ...         170     NML_STK     ARRAY [ 4 WORD ] := 0
                       171
0024 0000  0000        172     SYS_STK     RECORD [ ID OLD_FCW OLD_PC WORD ] := 0
0028 0000
                       173
```

```
174 !
175 PROCEDURE NAME
176
177     INIT
178
179
180 FUNCTION
181
182     Sets  up the Program Status Area Pointer and both
183     system  and  normal mode  stack  pointers  before
184     using  a  System  Call instruction  which  passes
185     control to the main routine, BREAKER.
186
187
188 REGISTERS USED
189
190     R0                           Temporary register
191     R15  (System stack pointer) Initialized
192     R15' (Normal stack pointer) Initialized
193
194
195 DESCRIPTION
196
197     The  routine is designed to be entered in  system
198     mode.   If  it  is not,  its  first  action,  the
199     loading of the Program Status Area Pointer,  will
200     result  in a privileged instruction trap.   As  a
201     result of the trap,  the routine will be  entered
202     again, this time in  system mode.  The new status
203     defined  on  line  117 of this  program  arranges
204     this.   Note  that this trap will  only  function
205     correctly  if the PSAP register has already  been
206     set  up  either by the Z-SCAN monitor,  or  by  a
207     previous correct execution of the routine.
208
209     The  system  stack  pointer is set up  simply  by
210     loading  R15  while the processor is  running  in
211     system  mode.  The SIZEOF operator is used to add
212     the length of the stack,  in bytes, to its lowest
213     address to obtain the correct initial value.  The
214     normal  stack pointer is set up by first  loading
215     its  initial  value  into  R0,   then  using  the
216     privileged LDCTL instruction.
217
218     BREAKER  is  reached  by using  the  System  Call
219     instruction  which terminates the  routine.   See
220     below for further details.
221 !
222
```

```
002A                 GLOBAL INIT PROCEDURE            ! Set up control reg's!
                     ENTRY
002A 7600  0000'         LDA     R0,NEW_STATUS_AREA  !   and both stacks   !
002E 7D0D                LDCTL   PSAP,R0
0030 210F  002A'         LD      R15,#SYS_STK + SIZEOF SYS_STK
0034 7600  0024'         LDA     R0,NML_STK + SIZEOF NML_STK
0038 7D0F                LDCTL   NSP,R0
003A 7F12                SC      #%12                 ! Trap into BREAKER   !
003C                 END INIT
```

```
233 !
234 PROCEDURE NAME
235
236     BREAKER
237
238
239 FUNCTION
240
241     Loops   continuously  once   entered,   providing
242     examples  of  many word bus cycle types  in  both
243     system and normal modes.
244
245
246 REGISTERS USED
247
248     R0                           Temporary register
249                                  Dest. for special I/O
250     R1                           Holds I/O port address
251     R2                           Dest for I/O read
252                                  Holds address of PASS
253     R3                           Src. for I/O, sp. I/O
254     R4                           Dest for read of PASS
255     R15  (System stack pointer) Data pointer; linkage
256     R15' (Normal stack pointer) Data pointer
257
258
259 DESCRIPTION
260
261     This  routine  is somewhat  'tricky',  so  it  is
262     described in some detail.
263
264     The procedure is always entered in system mode as
265     the  result  of  the execution of a  System  Call
266     instruction.  The new status defined on line 120
267     of  this program arranges this.   Z8000  programs
268     generally  handle  System Calls  by  executing  a
269     service   routine   which  terminates   with   an
270     Interrupt   Return  (IRET)   instruction.    This
271     restores  the FCW to its state before the  System
272     Call, and the PC to the address of the instrucion
273     following   the  call.   Instead of  doing  this,
274     BREAKER's  first action is to discard the old  PC
275     and  FCW  values  by restoring the  system  stack
276     pointer  to  the value which it held  before  the
277     System Call.
278
```

Alternate passes of the routine execute in system and normal modes. This is arranged by entering normal mode if and only if the status on the previous pass through the routine was made in system mode. The previous status is examined by testing the System/Normal bit (bit 14) of the old FCW value on the system stack with the BIT instruction on line 354. Note the use of the assembler's 'dot construct' to access a field within a record.

If the previous pass was in normal mode, the instructions on lines 356 through 358 are not executed, so system mode, set during the system call, remains in force. The instructions on lines 361 through 364 are executed to demonstrate I/O and special I/O accesses, which may only be made in system mode. Because the input instructions access non-existent ports, the data which they read is not predictable - though it is generally their own port address, held over by bus capacitance. The output instructions, over a period of about four seconds, output all possible odd data patterns (R3 is odd during passes through BREAKER in system mode).

The instructions from line 366 onwards are executed both on system mode and on normal mode passes. The first Load Address, demonstrates an internal operation cycle. The Byte/Word line, which is not defined during internal operations, goes to the byte state during this particular cycle.

The indirect Load instruction of line 367 performs a memory read which will eventually read all possible even values on system mode passes and all possible odd values on normal passes. Similarly, the PUSH on the next line will write those values, but access stack space instead of data space.

System mode passes result in the indirect INCrement instruction reading an even value from, and writing an odd value to, stack space. In normal mode odd data is read and even data written. Finally, the PUSH to a direct address reads stack space and writes data space. The value is odd in system mode, even in normal.

The final data manipulation of the routine is a write to program memory performed by a Load Relative instruction. Data is even for system mode, odd for normal.

```
                                334      Exit from,  or rather return to the start of, the
                                335      routine  is  achieved by means of a System  Call.
                                336      This  instruction  has  to  be  used  because  it
                                337      provides   the   only   'gateway'   (apart   from
                                338      privileged   and  extended   instruction   traps)
                                339      between   normal mode and system mode.   The  new
                                340      status  loaded by the instruction,  after it  has
                                341      saved  the  old status and its own opcode on  the
                                342      system  stack,  is  defined  on  line  120.   The
                                343      instruction  provides an example of  the  Z8002's
                                344      trap  response  sequence,  which starts  with  an
                                345      aborted  instruction  fetch  from   the  location
                                346      following  the System Call,  then pushes the  old
                                347      processor  status into system stack space  before
                                348      reading a new status from system code space.
                                349
                                350 !
003C                            351 INTERNAL BREAKER PROCEDURE      ! Demonstrate bus      !
                                352 ENTRY                           !  cycle types.        !
003C A9F5                       353     INC     R15,#SIZEOF SYS_STK ! Fix up system stack !
003E 670E   0026'               354     BIT     SYS_STK.OLD_FCW,#14 ! Check previous mode !
0042 E604                       355     JR      Z,ELSE_             ! If mode was system  !
0044 7D02                       356         LDCTL   R0,FCW          !  set normal mode by !
0046 A30E                       357         RES     R0,#14          !  clearing bit 14    !
0048 7D0A                       358         LDCTL   FCW,R0          !  of FCW;            !
004A E808                       359         JR      FI_             !  else do I/O.       !
004C 2101   ABCD                360 ELSE_:  LD      R1,#%ABCD       ! Dummy port address. !
0050 3D12                       361         IN      R2,@R1          ! I/O read            !
0052 3F13                       362         OUT     @R1,R3          ! I/O write           !
0054 3B05   1234                363         SIN     R0,%1234        ! Special I/O read    !
0058 3B37   1234                364         SOUT    %1234,R3        ! Special I/O write   !
                                365 FI_:                            ! Memory op's follow: !
005C 7602   0018'               366     LDA     R2,PASS             ! Internal operation  !
0060 2124                       367     LD      R4,@R2              ! Data read           !
0062 93F4                       368     PUSH    @R15,R4             ! Stk write           !
0064 29F0                       369     INC     @R15                ! Stk read, stk write !
0066 57F0   0018'               370     POP     PASS,@R15           ! Stk read, data write!
006A 3304   FFAC                371     LDR     LAST,R4             ! Code write          !
006E 7FEF                       372     SC      #%EF                ! Trap sequence       !
0070                            373 END BREAKER
                                374
                                375 END EXAMNSG

   0 errors
Assembly complete
```

Z8000ASM 2.02

```
LOC     OBJ CODE    STMT SOURCE STATEMENT

EXAMSEG MODULE
!
MODULE NAME

    EXAMSEG


ASSOCIATED MODULES

    None


FUNCTION

    To provide a demonstration of Z-SCAN's breakpoint
    and Target mode functions.  This is the only
    function of the program.


TARGET PROCESSOR

    Z8001.  Since this program contains trap and
    reset vectors, it cannot be run on the Z8002,
    even if that processor is run in nonsegmented
    mode.  See chapter 7 of the "Z8000 CPU Technical
    Manual" for futher details.  The following
    assembler directive selects segmented assembly
    for the Z8001.
!
    $SEGMENTED                      ! Assemble for Z8001 !
!

REQUIRED HARDWARE

    Z-SCAN 8000 with Z8001 processor.  No other
    hardware is required or assumed when the program
    is running.
```

```
                    PROGRAM SECTIONS USED

                        All  code and data is directed to a single  named
                        program section, EXAMSEG_P.  This allows code and
                        data  areas  to appear in memory in  exactly  the
                        order that they are defined in the  program.   By
                        default, the assembler would direct the data into
                        a  second section,  EXAMSEG_D.   The defaults are
                        useful  in applications where all procedure  code
                        must  be assigned to one memory area (often  ROM)
                        and all data to another.   IMAGER is used to make
                        such assignments.   This particular program needs
                        only a simple IMAGER command line (see tutorial).
                    !
                        $SECTION     EXAMSEG_P          ! Make imaging easy   !
                        $REL         %0000              ! Start at loc. 0000  !

                    !
                    RESET PROGRAM STATUS

                        Following a reset,  a Flag and Control Word (FCW)
                        value  of  %C000 is loaded to make the  processor
                        start  in segmented system mode  with  interrupts
                        and  extended processing disabled.   The  Program
                        Counter  (PC)  segment and offset  registers  are
                        loaded     with    the    start    address    of    the
                        initialization routine,  INIT.   This information
                        is  automatically  fetched from locations  %0002,
                        %0004 and %0006 in segment 00.   See section  7.4
                        of  the  "Z8000  CPU Technical Manual"  for  more
                        details.   The  reset status is defined  with  an
                        initialized   array  declaration  which  has  the
                        effect  of reserving two long (32 bit)  words  of
                        memory.
                    !

                    INTERNAL                            ! Private data area   !
                    NEW_STATUS_AREA:                    ! See note below      !

0000 0000  C000         RESET       ARRAY   [ 2 LONG ] := [ %C000, INIT ]
0004 8000' 0044'


                    !
                    NEW PROGRAM STATUS AREA

                        The  new program status area defines the  program
                        statuses which will be loaded following each type
                        of interrupt or trap, as described in section 7.6
                        of  the  "Z8000  CPU  Technical  Manual".    This
                        example uses only three of its many entries.   It
                        can start on any 256 byte boundary in system code
                        address  space.   The identifier NEW_STATUS_AREA,
                        defined above, points to the start of the area.
```

```
                    The destination of the privileged instruction
                    trap is the same procedure which is entered
                    following a reset.  Bit two, the half carry flag,
                    of the FCW is set in the first case and clear in
                    the second so that the user may easily identify
                    why the routine was entered.  Two different
                    System Call instruction opcodes are used for a
                    similar reason - examination of the top of the
                    system stack shows why BREAKER was entered.

                    The last vector defined handles Non-Maskable
                    Interrupts.  Like the system call and privileged
                    instruction traps, an NMI forces the CPU into the
                    INIT procedure.  The decimal adjust flag is set.
                !

                INTERNAL

                    $REL    %0010                 ! Privileged instr.   !
0010 0000  C004     PRIV_VECTOR ARRAY [ 2 LONG ] := [ %C004, INIT ]
0014 8000' 0044'

                    $REL    %0018                 ! System Call         !
0018 0000  C000     SC_VECTOR   ARRAY [ 2 LONG ] := [ %C000, BREAKER ]
001C 8000' 005A'

                    $REL    %0028                 ! Non-Maskable Int.   !
0028 0000  C008     NMI_VECTOR  ARRAY [ 2 LONG ] := [ %C008, INIT ]
002C 8000' 0044'


                !
                DATA AREA

                    The program uses just two words of static data:
                    PASS is incremented by one for each pass through
                    the procedure BREAKER (see below).  On each loop,
                    the previous value of PASS is copied into LAST.
                    Both variables are zero when the program is first
                    loaded, but as they are not intialized by the
                    program's initialization routine, their contents
                    must be considered to be undefined following a
                    reset.  Their position in memory is not critical.
                !

                INTERNAL                          ! Private data area   !

0030 0000  0000     PASS, LAST  WORD := 0

```

```
                      138 !
                      139 STACK AREA
                      140
                      141     The   Z8000  requires  two  call-return   linkage
                      142     stacks;  one  for  system mode operation and  the
                      143     other for normal mode.  For more information, see
                      144     section   4.3.3  of  the  "Z8000  CPU   Technical
                      145     Manual".   The  procedure  INIT sets up  the  two
                      146     stack pointers.   The normal stack is used by the
                      147     instructions  on  lines 370 through  372  in  the
                      148     procedure BREAKER.  These instructions affect the
                      149     top  word  of the stack area  only.   The  system
                      150     stack  is  used  by  the  same  instructions  and
                      151     additionally by the System Call (SC) instructions
                      152     on  lines 232 and 374.   This instruction  stacks
                      153     the old Program  Counter offset and segment,  the
                      154     old  Flag  and Control Word  and  an  identifier,
                      155     giving a total of four words.   The identifier is
                      156     the instruction itself.  For more details of this
                      157     instruction,     which     is      used     rather
                      158     unconventionally in this program,  see section  6
                      159     of   the  "Z8000  CPU  Technical  Manual".    The
                      160     position   of  the  stacks  in  memory   is   not
                      161     important.
                      162 !
                      163
                      164 INTERNAL                         ! Private data area   !
                      165
0034 0000   ...       166     NML_STK     ARRAY [ 4 WORD ] := 0
                      167
003C 0000   0000      168     SYS_STK RECORD [ID OLD_FCW WORD OLD_PC LONG] := 0
0040 0000   0000
                      169
```

```
170 !
171 PROCEDURE NAME
172
173     INIT
174
175
176 FUNCTION
177
178     Sets  up the Program Status Area Pointer and both
179     system  and  normal mode  stack  pointers  before
180     using  a  System  Call instruction  which  passes
181     control to the main routine, BREAKER.
182
183
184 REGISTERS USED
185
186     RRO                             Temporary register
187     RR14 (System stack pointer) Initialized
188     RR14'(Normal stack pointer) Initialized
189
190
191 DESCRIPTION
192
193     The  routine is designed to be entered in  system
194     mode.   If  it  is not,  its  first  action,  the
195     loading of the Program Status Area Pointer,  will
196     result  in a privileged instruction trap.   As  a
197     result of the trap,  the routine will be  entered
198     again, this time in  system mode.  The new status
199     defined  on  line  111 of this  program  arranges
200     this.   Note  that this trap will  only  function
201     correctly if the PSAP registers have already been
202     set  up  either by the Z-SCAN monitor,  or  by  a
203     previous correct execution of the routine.
204
205     The  system  stack pointer is set  up  simply  by
206     loading  RR14  while the processor is running  in
207     system  mode.  The SIZEOF operator is used to add
208     the length of the stack,  in bytes, to its lowest
209     address  to  obtain the  correct  initial  value.
210     Because  the stack resides in the first 256 bytes
211     of its segment, its address can be coded in short
212     offet form.   In order to conserve memory  space,
213     all  direct  addresses in the example  take  this
214     form.   The  normal  stack pointer is set  up  by
215     first  loading its initial value into  RRO,  then
216     using two privileged LDCTL instructions.
217
218     BREAKER  is  reached  by using  the  System  Call
219     instruction  which terminates the  routine.   See
220     below for further details.
221 !
222
```

```
0044                  GLOBAL INIT PROCEDURE            ! Set up control reg's!
                      ENTRY                            !   and both stacks   !
0044 7600   00' 00'       LDA      RR0,|NEW_STATUS_AREA|
0048 7D0C                 LDCTL    PSAPSEG,R0
004A 7D1D                 LDCTL    PSAPOFF,R1
004C 760E   00' 44'       LDA      RR14,|SYS_STK + SIZEOF SYS_STK|
0050 7600   00' 3C'       LDA      RR0,|NML_STK + SIZEOF NML_STK|
0054 7D0E                 LDCTL    NSPSEG,R0
0056 7D1F                 LDCTL    NSPOFF,R1
0058 7F12                 SC       #%12                    ! Trap into BREAKER   !
005A                  END INIT
```

```
235 !
236 PROCEDURE NAME
237
238     BREAKER
239
240
241 FUNCTION
242
243     Loops   continuously   once   entered,   providing
244     examples  of  many word bus cycle types  in  both
245     system and normal modes.
246
247
248 REGISTERS USED
249
250     R0                              Temporary register
251                                     Dest. for special I/O
252     R1                              Holds I/O port address
253     R2                              Dest for I/O read
254     RR2                             Holds address of PASS
255     R3                              Src. for I/O, sp. I/O
256     R4                              Dest for read of PASS
257     RR14 (System stack pointer) Data pointer; linkage
258     RR14'(Normal stack pointer) Data pointer
259
260
261 DESCRIPTION
262
263     This  routine  is somewhat  'tricky',  so  it  is
264     described in some detail.
265
266     The procedure is always entered in system mode as
267     the  result  of  the execution of a  System  Call
268     instruction.  The new status defined on line 114
269     of  this program arranges this.  Z8000  programs
270     generally  handle  System Calls  by  executing  a
271     service   routine   which   terminates   with   an
272     Interrupt   Return  (IRET)   instruction.    This
273     restores  the FCW to its state before the  System
274     Call, and the PC to the address of the instrucion
275     following   the  call.  Instead of  doing  this,
276     BREAKER's  first action is to discard the old  PC
277     and  FCW  values  by restoring the  system  stack
278     pointer  to  the value which it held  before  the
279     System Call.
280
```

Alternate passes of the routine execute in system and normal modes. This is arranged by entering normal mode if and only if the status on the previous pass through the routine was made in system mode. The previous status is examined by testing the System/Normal bit (bit 14) of the old FCW value on the system stack with the BIT instruction on line 356. Note the use of the assembler's 'dot construct' to access a field within a record.

If the previous pass was in normal mode, the instructions on lines 358 through 360 are not executed, so system mode, set during the system call, remains in force. The instructions on lines 363 through 366 are executed to demonstrate I/O and special I/O accesses, which may only be made in system mode. Because the input instructions access non-existent ports, the data which they read is not predictable - though it is generally their own port address, held over by bus capacitance. The output instructions, over a period of about four seconds, output all possible odd data patterns (R3 is odd during passes through BREAKER in system mode).

The instructions from line 368 onwards are executed both on system mode and on normal mode passes. The first Load Address, demonstrates an internal operation cycle. The Byte/Word line, which is not defined during internal operations, goes to the byte state during this particular cycle.

The indirect Load instruction of line 369 performs a memory read which will eventually read all possible even values on system mode passes and all possible odd values on normal passes. Similarly, the PUSH on the next line will write those values, but access stack space instead of data space.

System mode passes result in the indirect INCrement instruction reading an even value from, and writing an odd value to, stack space. In normal mode odd data is read and even data written. Finally, the PUSH to a direct address reads stack space and writes data space. The value is odd in system mode, even in normal.

The final data manipulation of the routine is a write to program memory performed by a Load Relative instruction. Data is even for system mode, odd for normal.

```
                          336     Exit from,  or rather return to the start of, the
                          337     routine  is  achieved by means of a System  Call.
                          338     This  instruction  has  to  be  used  because  it
                          339     provides   the   only   'gateway'   (apart   from
                          340     privileged   and  extended   instruction   traps)
                          341     between   normal mode and system mode.   The  new
                          342     status  loaded by the instruction,  after it  has
                          343     saved  the  old status and its own opcode on  the
                          344     system  stack,  is  defined  on  line  114.   The
                          345     instruction  provides an example of  the  Z8001's
                          346     trap  response  sequence,  which starts  with  an
                          347     aborted  instruction  fetch  from   the  location
                          348     following  the System Call,  then pushes the  old
                          349     processor  status into system stack space  before
                          350     reading a new status from system code space.
                          351
                          352 !
005A                      353 INTERNAL BREAKER PROCEDURE      ! Demonstrate bus     !
                          354 ENTRY                           !  cycle types.       !
005A A9F7                 355     INC     R15,#SIZEOF SYS_STK ! Fix up system stack !
005C 670E   00' 3E'       356     BIT     |SYS_STK.OLD_FCW|,#14!Check previous mode !
0060 E604                 357     JR      Z,ELSE_             ! If mode was system  !
0062 7D02                 358         LDCTL   R0,FCW          !  set normal mode by !
0064 A30E                 359         RES     R0,#14          !  clearing bit 14    !
0066 7D0A                 360         LDCTL   FCW,R0          !  of FCW;            !
0068 E808                 361         JR      FI_             !  else do I/O.       !
006A 2101   ABCD          362 ELSE_:  LD      R1,#%ABCD       ! Dummy port address. !
006E 3D12                 363         IN      R2,@R1          ! I/O read            !
0070 3F13                 364         OUT     @R1,R3          ! I/O write           !
0072 3B05   1234          365         SIN     R0,%1234        ! Special I/O read    !
0076 3B37   1234          366         SOUT    %1234,R3        ! Special I/O write   !
                          367 FI_:                            ! Memory op's follow: !
007A 7602   00' 30'       368     LDA     RR2,|PASS|          ! Internal operation  !
007E 2124                 369     LD      R4,@RR2             ! Data read           !
0080 93E4                 370     PUSH    @RR14,R4            ! Stk write           !
0082 29E0                 371     INC     @RR14               ! Stk read, stk write !
0084 57E0   00' 30'       372     POP     |PASS|,@RR14        ! Stk read, data write!
0088 3304   FFA6          373     LDR     LAST,R4             ! Code write          !
008C 7FEF                 374     SC      #%EF                ! Trap sequence       !
008E                      375 END BREAKER
                          376
                          377 END EXAMSEG

    0 errors
Assembly complete
```

APPENDIX C

RIO LOAD AND SEND

PROGRAM LISTINGS

```
*HEADING MAIN ROUTINE

        EXTERNAL        SYSTEM

;PREDEFINED DEVICES


FILDEV  EQU 5           ;DEVICE FOR FILES
CONIN   EQU 1           ;CONSOLE INPUT
CONOUT  EQU 2           ;CONSOLE OUTPUT

;SPECIAL CODES ETC.

PROC    EQU 80H         ;PROCEDURE FILE
OPNINP  EQU 0           ;OPEN FOR INPUT
MAXRTY  EQU 10          ;MAX TIMES TO RETRY ON LOAD

;DEVICE FUNCTION CODES

ZDASGN  EQU 2           ;ASSIGN
ZDOPEN  EQU 4           ;OPEN
ZDCLOS  EQU 6           ;CLOSE DEVICE
ZDREDB  EQU 0AH         ;READ BINARY
ZDREDA  EQU 0CH         ;READ ASCII LINE
ZDWRTB  EQU 0EH         ;WRITE BINARY
ZDRDST  EQU 40H         ;READ STATUS

```

```
                 *HEADING MAIN PROGRAM
                 ; *****************************************************
                 ; *
                 ; * THIS PROGRAM IS USED TO SEND Z8000 PROGRAM TO THE
                 ; * Z8000 EVALUATION BOARD IN TEKTRONIX FORMAT.
                 ; * IF THE REQUESTED FILE DOES NOT EXIST OR UNABLE TO
                 ; * OPEN FILE, IT SEND RECORD WITH // FOLLOWED BY ERROR
                 ; * MESSAGE TO THE CONSOLE OUTPUT AND PROGRAM WILL BE
                 ; * ABORTED.  EACH DATA RECORD WILL BE RESENDED 10 TIMES
                 ; * IF THE Z8000 EVALUATION BOARD RESPONSE WITH NON-
                 ; * ACKNOWLEDGE.  PROGRAM WILL ALSO BE ABORTED IF THE
                 ; * RESPONSE IS AN ASCII CHARACTER '9'.
                 ; *
                 ; *****************************************************
0000  180B                JR      COMSY1
0002  44415445            DEFM 'DATE:790524'
                 COMSY1:
000D  215803   R          LD      HL,ACKMSG        ; SEND 0 FOR PROGRAM GET
                                                   ;   LOADED OK
0010  CDF301   R          CALL    PUTSTR
0013  E1                  POP     HL               ; RIO RETURN
0014  E3                  EX      (SP),HL          ; GET COMMAND POINTER
0015  7E         SKIPSP:  LD      A,(HL)
0016  FE20                CP      ' '              ; SKIP OVER SPACE
0018  2003                JR      NZ, CHMORE
001A  23                  INC     HL
001B  18F8                JR      SKIPSP
001D  FE3B       CHMORE:  CP      ';'              ; CHECK FOR DELIMITER
001F  2836                JR      Z,FILERR
0021  FE2C                CP      ','
0023  2832                JR      Z, FILERR
0025  FE0D                CP      0DH
0027  282E                JR      Z,FILERR         ; FILENAME ERROR
0029  CD5E00   R          CALL    OPNFIL           ; OPEN FILE FOR INPUT
002C  2016                JR      NZ, OPNB         ; UNABLE TO OPEN FILE
002E  3A4B02   R          LD      A, (FILPRM)      ; CHECK IF PROC.FILE
0031  E6F0                AND     0F0H
0033  FE80                CP      PROC
0035  2809                JR      Z, PROCF         ; YES
0037  FD213F02 R          LD      IY,FILCLS
003B  CD0000   X          CALL    SYSTEM           ; NO, CLOSE FILE
003E  1812                JR      NPROC
                 PROCF:
0040  CD7E00   R          CALL    RDFILE           ; READ FILE & SEND DATA
0043  C9                  RET                      ; BACK TO RIO
```

```
                      86
                      87  OPNB:
0044    21F802    R   88            LD      HL,ERADR          ; INSERT ERROR CODE
0047    FD4E0A        89            LD      C,(IY+10)
004A    CDC501    R   90            CALL    GENCKS
004D    21E002    R   91            LD      HL, NOOPEN        ; UNABLE TO OPEN FILE
0050    1808          92            JR      PUTS
                      93
0052    210E03    R   94  NPROC:    LD      HL, NOPROC        ; NOT PROCEDURE FILE
0055    1803          95            JR      PUTS
0057    21FC02    R   96  FILERR:   LD      HL,ERFILE         ; FILENAME ERROR
005A    CDF301    R   97  PUTS:     CALL    PUTSTR
005D    C9            98            RET
```

```
 LOC    OBJ CODE M STMT SOURCE STATEMENT                        ASM 5.9

                   99  *HEADING OPNFIL ROUTINE
                  100
                  101
                  102
                  103  ; ****************************************************
                  104  ; *
                  105  ; * OPNFIL ASSIGN A LOGICAL UNIT # TO THE GIVEN
                  106  ; *        FILENAME AND ALSO OPEN THE FILE
                  107  ; *
                  108  ; * INPUT:  HL - POINTER TO FILENAME
                  109  ; *
                  110  ; * OUTPUT: RETURN Z IF OPEN FILE SUCCESSFUL
                  111  ; *         RETURN NZ IF UNABLE TO ASSIGN FILENAME
                  112  ; *                  OR FILE DOES NOT EXIST
                  113  ; *
                  114  ; ****************************************************
                  115
                  116  OPNFIL:
005E    220402   R  117          LD      (FILPTR), HL    ; FILENAME PTR INTO IY \
0061    FD210202 R  118          LD      IY, FILASV
0065    CD0000   X  119          CALL    SYSTEM          ; ASSIGN FILE
0068    FDCB0A76    120          BIT     6, (IY+10)      ; ASSIGN OK?
006C    C0          121          RET     NZ              ; NO
                    122
                    123
006D    3E00        124          LD      A,OPNINP        ; OPEN TYPE - OLDFILE
006F    320F02   R  125          LD      (FILSPV),A
0072    FD213202 R  126          LD      IY, FILOPN      ; OPEN FILE
0076    CD0000   X  127          CALL    SYSTEM
0079    FDCB0A76    128          BIT     6, (IY+10)      ; OPEN OK?
007D    C9          129          RET                     ; RETURN NZ OR Z
```

```
               *HEADING RDFILE ROUTINE

               ; *****************************************************
               ; *
               ; * RDFILE READ DATA FROM FILE AND FORMAT THEM IN
               ; *        TEKTRONIX FORMAT TO SEND OUT TO CONSOLE
               ; *        OUTPUT.  MAXIMUM WILL READ 80 BYTES OF DATA
               ; *        EACH TIME.  THE ADDRESS FOR THIS DATA IS
               ; *        OBTAINED FROM FILE SEGMENT TABLE.
               ; *
               ; *
               ; *****************************************************

               RDFILE:
007E  216702   R         LD      HL, FILSMT       ; SEGMENT TABLE LOCATIOI
0081  225D03   R         LD      (SEGPTR), HL     ; POINTER TO SEGMENT TAI

0084  2A5D03   R NXTSEG: LD      HL, (SEGPTR)     ; PTR TO NEXT SEGMENT
0087  4E                 LD      C,(HL)
0088  23                 INC     HL
0089  46                 LD      B, (HL)          ; SEGMENT ADDRESS
008A  23                 INC     HL
008B  5E                 LD      E,(HL)
008C  23                 INC     HL
008D  56                 LD      D,(HL)           ; SEGMENT SIZE
008E  23                 INC     HL
008F  225D03   R         LD      (SEGPTR), HL     ; UPDATE SEGMFNT TBL PTI
0092  7B                 LD      A,E
0093  B2                 OR      D
0094  CA2101   R         JP      Z, FINL          ; SEND LAST RECORD


               ; READ DATA FROM FILE.  IF CURRENT SEGMENT SIZE <400H, 
               ; THE # OF BYTE TO READ IS SEGMENT SIZE, OTHERWISE READ
               ; 400H BYTE EACH TIME.
               ; CORADR - ACTUAL ADDRESS TO BE FORMAT IN THE RECORD
               ; SEGCNT - # OF BYTE IN CURRENT SEGMENT NEED TO BE READ
               ; RDVLEN - ACTUAL # TO READ DATA FROM FILE
               ; CURBUF - POINTER TO FILBFR BUFFER FOR DATA TO BE FORM/
               ; REMCNT - # OF BYTE LEFT IN FILBFR FOR NEXT RECORD


0097  ED435F03 R         LD      (CORADR),BC      ; CORE ADDR USED IN REC(
009B  EB                 EX      DE,HL

009C  226103   R NXTBFR: LD      (SEGCNT), HL     ; # OF BYTE LEFT IN CURI
                                                  ;  SEGMENT TO BE READ
009F  010004             LD      BC,400H          ; MAX BYTE TO READ FILE
00A2  A7                 AND     A
00A3  ED42               SBC     HL,BC
00A5  3004               JR      NC,UBIG1         ;HL>400H SO USE 400H AS
00A7  ED4B6103 R         LD      BC,(SEGCNT)      ;USE SIZE REMAINING
00AB  ED43C302 R UBIG1:  LD (RDVLEN),BC
00AF  21E903   R         LD HL,FILBFR
00B2  226303   R         LD      (CURBUF),HL      ;PTR TO FILBFR FOR DATA
00B5  C5                 PUSH BC                  ;SAVE # OF BYTE TO READ
```

```
00B6     FD21BF02 R   188          LD IY,FILRDV
00BA     CD0000   X   189          CALL SYSTEM              ;READ DATA FROM FILE
00BD     E1           190          POP HL                   ; # OF BYTE THAT READ
00BE     22C302   R   191          LD (RDVLEN),HL           ; MAY HAVE GOTTEN BIGGEI
00C1     FDCB0A76     192          BIT 6,(IY+10)
00C5     280E         193          JR Z,NXTLIN              ;OK
                      194
                      195  ; ERROR IN READING FILE, SEND RECORD WITH ERROR MESSAGE
                      196  ; TO CONSOLE OUTPUT
                      197
00C7     213E03   R   198          LD      HL,ERRADR
00CA     FD4E0A       199          LD      C,(IY+10)        ; INSERT ERROR CODE
00CD     CDC501   R   200          CALL    GENCKS
00D0     212403   R   201          LD      HL,RDBAD         ; FILE READ ERROR
00D3     1876         202          JR      PUTCLS           ; OUTPUT MESSAGE&CLOSEFI
                      203
                      204
                      205
00D5     226503   R   206  NXTLIN: LD (REMCNT),HL           ; # OF BYTE LEFT IN FILE
00D8     111E00       207          LD DE,30                 ; FORMAT 30 BYTE /RECORI
00DB     A7           208          AND A
00DC     ED52         209          SBC HL,DE
00DE     3004         210          JR NC,UBIG2              ;REMCNT>30, USE 30
00E0     ED5B6503 R   211          LD DE,(REMCNT)           ;USE REMCNT (ITS SMALLEI
00E4     ED535B03 R   212  UBIG2:  LD      (RECBYE),DE      ; SAVE RECORD BYTE COUNT
00E8     43           213          LD      B,E
00E9     3E0A         214          LD A,MAXRTY
00EB     326703   R   215          LD (RTYCNT),A            ; RETRY 10 TIMES
                      216
00EE     ED5B6303 R   217          LD      DE, (CURBUF)     ; FILBFR PTR FOR DATA
00F2     CD8201   R   218          CALL    TEKLIN           ;    TO BE FORMATTED
00F5     CD5601   R   219          CALL    TKSWAT           ; GET ACKNOWLEDGE
00F8     2804         220          JR      Z, ACKOK         ; GOOD ACKNOWLEDGE
00FA     3852         221          JR      C,CLSF           ; ABORT-ACKNOWLEDGE,
                      222                                   ;  CLOSE FILE & RETURN
00FC     184A         223          JR      TMUCH            ; RETRY 10 TIMES ALREADY
                      224                                   ; ABORT
                      225
                      226
                      227  ; UPDATE RECORD ADDRESS & CHECK ANY DATA LEFT IN FILBFR
                      228
                      229  ACKOK:
00FE     ED4B5B03 R   230          LD      BC,(RECBYE)      ; RECORD BYTE COUNT
                      231
0102     2A5F03   R   232          LD      HL,(CORADR)      ; UPDATE ADDR TO BE USED
0105     09           233          ADD     HL, BC           ;   IN NEXT RECORD
0106     225F03   R   234          LD      (CORADR),HL      ; UPDATE
0109     2A6503   R   235          LD      HL, (REMCNT)     ; # OF BYTE LEFT IN FILB
010C     A7           236          AND     A
010D     ED42         237          SBC     HL,BC
010F     20C4         238          JR      NZ,NXTLIN        ; MORE IN FILBFR
                      239
                      240  ; CHECK IF ALL DATA IS BEING READ FROM CURRENT SEGMENT
                      241  ; EITHER READ SOME MORE DATA FROM THE SAME SEGMENT
                      242  ; OR READ DATA FROM NEXT SEGMENT
                      243
0111     2A6103   R   244          LD      HL,(SEGCNT)      ; IF # OF BYTE READ=SEGM
0114     ED4BC302 R   245          LD      BC,(RDVLEN)      ;  SIZE, THEN READ NEXT
```

```
 LOC    OBJ CODE M STMT SOURCE STATEMENT

0118    A7          246           AND     A                 ;  SEGMENT
0119    ED42        247           SBC     HL, BC
011B    C29C00   R  248           JP      NZ,NXTBFR         ; READ FROM SAME SEGMEN
                    249
011E    C38400   R  250           JP      NXTSEG            ; READ FROM NEXT SEGMEN
                    251
                    252
                    253  ; SEGMENT SIZE IS ZERO, SO SEND LAST RECORD AS FOLLOWS:
                    254  ;
                    255  ; /<ENTRY ADDR>00<CKSUM><RETURN>
                    256
                    257  FINL:
0121    3E0A        258           LD      A, MAXRTY         ; RETRY 10 TIMES IF FAI
0123    326703   R  259           LD      (RTYCNT),A
0126    21E908   R  260           LD      HL, OUTBUF        ; FORMAT DATA IN OUTBUF
0129    362F        261           LD      (HL),'/'
012B    23          262           INC     HL
012C    0E00        263           LD      C,0
012E    3A5402   R  264           LD      A,(FILSA+1)       ; GET ENTRY ADDRESS
0131    CDC601   R  265           CALL    GENHEX
0134    3A5302   R  266           LD      A,(FILSA)
0137    CDC601   R  267           CALL    GENHEX
013A    AF          268           XOR     A                 ; GENERATE BYTE COUNT=0
013B    CDC601   R  269           CALL    GENHEX
013E    CDAA01   R  270           CALL    TKMSFN            ; FINISH DATA CHECKSUM
                    271                                     ;  SEND OUT DATA
0141    CD5601   R  272           CALL    TKSWAT            ; WAIT FOR ACKNOWLEDGE
0144    2808        273           JR      Z,CLSF            ; GOOD-ACKNOWLEDGE,CLOS
                    274                                     ;  FILE & RETURN
0146    3806        275           JR      C,CLSF            ; ABORT-ACKNOWLEDGE,CLO
                    276                                     ;  FILE & RETURN
                    277
                    278  TMUCH:
0148    214203   R  279           LD      HL,TRYMCH         ; RESEND SAME RECORD 10
                    280                                     ; ALREADY, SEND '//RECO
                    281                                     ;  CKSUM ERROR' TO Z800
014B    CDF301   R  282  PUTCLS:  CALL    PUTSTR
                    283
                    284  CLSF:
014E    FD213F02 R  285           LD      IY,FILCLS         ; CLOSE FILE
0152    CD0000   X  286           CALL    SYSTEM
0155    C9          287           RET
                    288
```

```
                    *HEADING TKSWAT ROUTINE
                    ; ****************************************************
                    ; *
                    ; * TKSWAT WAIT FOR RESPONSE FROM CONSOLE INPUT.  IF
                    ; * ASCII 7 IS RECEIVED, RESEND THE SAME DATA RECORD
                    ; * AGAIN UP TO 10 TIMES.  IF ASCII 0 IS RECEIVED,
                    ; * INDICATE GOOD-ACKNOWLEDGE.  IF ASCII 9 IS RECIEVED,
                    ; * ABORT PROGRAM.
                    ; *
                    ; * OUTPUT:  RETURN Z IF RECEIVING ACKNOWLEDGE
                    ; *          RETURN NZ, NC IF ASCII 7 IS RECIVED EVEN
                    ; *                        RESEND DATA 10 TIMES
                    ; *          RETURN NZ, C  IF ASCII 9 IS RECIVED TO
                    ; *                        ABORT THE PROGRAM
                    ; *
                    ; ****************************************************
                    TKSWAT:
0156  CDE001    R               CALL    CONINP          ; GET RESPOND
0159  FDCB0A76                  BIT     6,(IY+10)       ; CHECK ANY ERROR
015D  C0                        RET     NZ              ; CONSOLE READ ERROR
015E  3A6903    R               LD      A,(CHRBUF)
0161  FE30                      CP      '0'
0163  C8                        RET     Z               ; GOOD ACKNOWLEDGE
0164  FE37                      CP      '7'
0166  2808                      JR      Z, NACK
0168  FE39                      CP      '9'             ; CHECK FOR ABORT
016A  20EA                      JR      NZ,TKSWAT       ; NO, WAIT FOR AGAIN
016C  F601                      OR      1
016E  37                        SCF                     ; RETURN NZ,C FOR ABORT
                                                        ;  ACKNOWLEDGE
016F  C9                        RET
                    NACK:
0170  216703    R               LD      HL,RTYCNT
0173  35                        DEC     (HL)
0174  2003                      JR      NZ, RTRY        ; RETRY
0176  F601                      OR      1
0178  C9                        RET                     ; RETURN NZ,NC FOR RETF
                    RTRY:
0179  FD21D502  R               LD      IY,STRVEC       ; RESEND SAME RECORD
017D  CD0000    X               CALL    SYSTEM
0180  18D4                      JR      TKSWAT          ; WAIT FOR ACKNOWLDGE
```

```
                    342  *HEADING TEKLIN
                    343
                    344
                    345  ; *****************************************************
                    346  ;
                    347  ; DATA IN TEKTRONIX FROMAT
                    348  ; /<ADDRESS><COUNT><CKSUM><DATA>...<DATA><CKSUM>CR
                    349  ;
                    350  ;COUNT, DATA, AND CHECKSUMS ARE EACH 2 CHARATERS OF ASCI
                    351  ;REPRESENTING ONE HEX BYTE
                    352  ;LOAD ADDRESS IS 4 CHARACTER
                    353  ;CHECKSUMS ARE THE SUM OF THE 4 BIT NIBBLES REPRESENTED
                    354  ;THE CHARACTERS LINE TERMINATES WITH A RETURN
                    355  ;CORADR IS THE LOAD ADDRESS
                    356  ;DE IS THE SOURCE OF THE BYTES (IN FILBFR)
                    357  ;B IS THE COUNT
                    358  ;
                    359  ; *****************************************************
                    360
                    361
0182  21E908     R  362  TEKLIN: LD HL,OUTBUF              ;BUILD LINE HERE
0185  362F          363          LD (HL),'/'
0187  23            364          INC HL
0188  0E00          365          LD C,0                    ;FOR CHECKSUM
018A  3A6003     R  366          LD A,(CORADR+1)
018D  CDC601     R  367          CALL    GENHEX            ; CONVERT ADDRESS TO ASC
0190  3A5F03     R  368          LD A,(CORADR)
0193  CDC601     R  369          CALL GENHEX               ;THE ADDRESS
0196  78            370          LD      A, B              ; CONVERT COUNT TO ASCII
0197  CDC601     R  371          CALL    GENHEX
019A  CDC501     R  372          CALL GENCKS               ;SEND C (ACCUMULATED CKS
019D  0E00          373          LD C,0                    ;RESET
                    374  TKLNLP:
019F  1A            375          LD      A, (DE)           ; CONVERT DATA
01A0  13            376          INC DE
01A1  CDC601     R  377          CALL GENHEX
01A4  10F9          378          DJNZ TKLNLP
01A6  ED536303   R  379          LD      (CURBUF), DE      ; FILBFR PTR FOR NEXT DA
                    380
                    381  TKMSFN:
01AA  CDC501     R  382          CALL    GENCKS            ; SEND DATA CKSUM
01AD  360D          383          LD      (HL), 0DH         ; ADD CR
01AF  11E808     R  384          LD      DE,OUTBUF-1       ; FIGURE BUF SIZE TO SEN
01B2  A7            385          AND     A
01B3  ED52          386          SBC     HL,DE
01B5  22D902     R  387          LD      (STRLEN),HL       ;LENGTH OF BUFFER
01B8  21E908     R  388          LD      HL, OUTBUF
01BB  22D702     R  389          LD      (STRDTA), HL      ; ADDR OF BUF
01BE  FD21D502   R  390          LD      IY, STRVEC
01C2  C30000     X  391          JP      SYSTEM
                    392
```

```
                        *HEADING GENHEX, GENCKS


                        ; ******************************************************
                        ; *
                        ; * GENHEX CONVERT THE BYTE DATA IN REGISTER A TO TWO
                        ; *        ASCII CHARACTERS INTO THE OUTPUT STREAM
                        ; *        POINTED BY HL.  ALSO ACCUMULATES THE CHECKSU
                        ; *        INC AS THE SUM OF THE 4 BIT NIBBLES.
                        ; * GENCKS CONVERT THE BYTE DATA IN REGISTER C
                        ; *
                        ; * INPUT:  A OR C - 8 BITS HEX VLAUE
                        ; *          HL - POINTER TO OUTPUT STREAM
                        ; *
                        ; * OUTPUT: HL - INCREMENTED BY 2
                        ; *
                        ; ******************************************************


01C5   79               GENCKS: LD A,C
01C6   F5               GENHEX: PUSH AF
01C7   07                       RLCA
01C8   07                       RLCA
01C9   07                       RLCA
01CA   07                       RLCA
01CB   CDCF01   R               CALL NIBBLE
01CE   F1                       POP AF                  ;AND FALL INTO NIBBLE
01CF   E60F             NIBBLE: AND OFH
01D1   F5                       PUSH AF                 ;SAVE WHILE GENERATING C
01D2   81                       ADD A,C
01D3   4F                       LD C,A
01D4   F1                       POP AF
01D5   FE0A                     CP 10
01D7   3802                     JR C,NIBDIG             ;SEND AS A DIGIT
01D9   C607                     ADD A,'A'-'0'-10        ;SEND AS A LETTER
01DB   C630             NIBDIG: ADD A,'0'
01DD   77                       LD (HL),A
01DE   23                       INC HL
01DF   C9                       RET
```

```
                         433   *HEADING SUBROUTINES.COMM
                         434
                         435   ; GET CHARACTER FROM SERIAL LINE
                         436
                         437
                         438   CONINP:
01E0     216903    R     439            LD       HL, CHRBUF       ; RECEIVE CHARACTER STR
01E3     22CC02    R     440            LD       (CONDTA),HL
01E6     218000          441            LD       HL, 128
01E9     22CE02    R     442            LD       (CONLEN),HL      ; LENGTH OF BUF
01EC     FD21CA02  R     443            LD       IY, CONVEC
01F0     C30000    X     444            JP       SYSTEM
                         445
                         446
                         447
                         448   ; ******************************************************
                         449   ; *
                         450   ; * PUTSTR OUTPUT A MESSAGE, HL POINT TO THE MESSAGE IN
                         451   ; *        DEFT FORMAT.
                         452   ; *
                         453   ; * INPUT:  HL - POINTER TO MESSAGE
                         454   ; *
                         455   ; ******************************************************
                         456
                         457
01F3     7E              458   PUTSTR: LD A,(HL)                  ;GET COUNT
01F4     23              459           INC HL
01F5     22D702    R     460           LD (STRDTA),HL
01F8     32D902    R     461           LD (STRLEN),A              ;OTHER PART IS ALWAYS 0
01FB     FD21D502  R     462           LD IY,STRVEC
01FF     C30000    X     463           JP SYSTEM
                         464
```

```
 LOC    OBJ CODE M STMT SOURCE STATEMENT

                  465  *HEADING SYSTEM CALL VECTOR
                  466
                  467
                  468  ; ASSIGN FILE
                  469
0202    05        470  FILASV: DEFB    FILDEV
0203    02        471          DEFB    ZDASGN          ; ASSIGN REQUEST CODE
0204    0000      472  FILPTR: DEFW    0               ; FILENAME POINTER
0206    0000      473          DEFW    0
0208    0000      474          DEFW    0
020A    0000      475          DEFW    0
020C    00        476          DEFB    0
020D    0F02    R 477          DEFW    FILSPV          ; SUPPL.PARAMETER VECTOI
                  478
020F    00        479  FILSPV: DEFB    0               ; FLAG BYTE
0210              480          DEFS    34
                  481
                  482  ; OPEN FILE
                  483
0232    05        484  FILOPN: DEFB    FILDEV          ; SAME UNIT AS ASSIGN
0233    04        485          DEFB    ZDOPEN          ; OPEN REQUEST CODE
0234    4B02    R 486          DEFW    FILPRM          ; FILE ATTRIBUTES AREA
0236    7400      487          DEFW    116             ; LENGTH OF FILE ATRIBU?
0238    0000      488          DEFW    0
023A    0000      489          DEFW    0
023C    00        490          DEFB    0
023D    0F02    R 491          DEFW    FILSPV          ; SUPPL.PARAMETER VECTOI
                  492
                  493  ; CLOSE FILE
                  494
023F    05        495  FILCLS: DEFB    FILDEV          ; SAME UNIT AS ASSIGN
0240    06        496          DEFB    ZDCLOS          ; CLOSE REQUEST CODE
0241    0000      497          DEFW    0
0243    0000      498          DEFW    0
0245    0000      499          DEFW    0
0247    0000      500          DEFW    0
0249    0000      501          DEFW    0
                  502
                  503  FILPRM:
024B    00        504          DEFB    0       ; FILE TYPE ( MUST BE PROC.)
024C    0000      505          DEFW    0       ; SIZE
024E    0000      506  FILRL:  DEFW    0       ; RECORD LENGTH
0250              507          DEFS    2       ; BLOCKING
0252              508  FILPRP: DEFS    1       ; PROPERTIES
0253    0000      509  FILSA:  DEFW    0       ; STARTING ADDRESS
0255              510          DEFS    18      ; BYTE IN LAST RECORD,CREATION I
0267              511  FILSMT: DEFS    82      ; SEGMENT POINTER TABLE
02B9              512  LOWADR: DEFS    2       ; LOWEST ADDRESS USED
02BB              513  HIGADR: DEFS    2       ; HIGHEST ADDRESS UES
02BD              514          DEFS    2       ; STACK SIZE
                  515
                  516
                  517  ; READ FILE VECTOR
                  518
02BF    05        519  FILRDV: DEFB    FILDEV
02C0    0A        520          DEFB    ZDREDB
02C1    E903    R 521          DEFW    FILBFR
02C3              522  RDVLEN: DEFS    2
```

```
02C5    0000         523            DEFW     0
02C7    0000         524            DEFW     0
02C9    00           525            DEFB     0
                     526
                     527  ;CONSOLE READ AND WRITE
                     528
02CA    01           529  CONVEC: DEFB CONIN
02CB    0C           530          DEFB ZDREDA
02CC    0000         531  CONDTA: DEFW 0
02CE    0000         532  CONLEN: DEFW 0
02D0    0000         533          DEFW 0
02D2    0000         534          DEFW 0
02D4    00           535          DEFB 0
                     536
02D5    02           537  STRVEC: DEFB CONOUT
02D6    0E           538          DEFB ZDWRTB
02D7    0000         539  STRDTA: DEFW 0
02D9    0000         540  STRLEN: DEFW 0
02DB    0000         541          DEFW 0
02DD    0000         542          DEFW 0
02DF    00           543          DEFB 0
                     544
02E0    1B           545  NOOPEN: DEFB     B0-NOOPEN
02E1    2F2F554E     546          DEFM     '//UNABLE TO OPEN FILE ('
02F8    3830         547  ERADR:  DEFW     3038H
02FA    29           548          DEFB     ')'
02FB    0D           549          DEFB     0DH
                     550  B0      EQU      $-1
                     551
                     552
02FC    11           553  ERFILE: DEFB     B1-ERFILE
02FD    2F2F4649     554          DEFM     '//FILENAME ERROR'
030D    0D           555          DEFB     0DH
                     556  B1      EQU      $-1
                     557
030E    15           558  NOPROC: DEFB     B2-NOPROC
030F    2F2F4E4F     559          DEFM     '//NOT PROCEDURE FILE'
0323    0D           560          DEFB     0DH
                     561  B2      EQU      $-1
                     562
0324    1D           563  RDBAD:  DEFB     B3-RDBAD
0325    2F2F4552     564          DEFM     '//ERROR IN READING FILE ('
033E    3830         565  ERRADR: DEFW     3038H
0340    29           566          DEFB     ')'
0341    0D           567          DEFB     0DH
                     568  B3      EQU      $-1
                     569
0342    15           570  TRYMCH: DEFB     B4-TRYMCH
0343    2F2F5245     571          DEFM     '//RECORD CKSUM ERROR'
0357    0D           572          DEFB     0DH
                     573  B4      EQU      $-1
                     574
0358    02           575  ACKMSG: DEFB     2
0359    30           576          DEFB     '0'
035A    0D           577          DEFB     0DH
                     578
035B    0000         579  RECBYE: DEFW     0              ; RECORD BYTE COUNT
035D    0000         580  SEGPTR: DEFW     0              ; SEGMENT TABLE POINTER
```

```
035F    0000       581  CORADR: DEFW     0          ; CORE ADDR USED IN REC(
0361    0000       582  SEGCNT: DEFW     0          ; # OF BYTE REMAINING II
                   583                              ;  CURRENT SEGMENT
0363    0000       584  CURBUF: DEFW     0          ; CURRENT FILBFR ADDRES:
0365    0000       585  REMCNT: DEFW     0          ; REMAINING BYTE IN FILI
0367    0000       586  RTYCNT: DEFW     0          ; RETRY COUNT
0369               587  CHRBUF: DEFS     128        ; INPUT BUFFER
03E9               588  FILBFR: DEFS     500H       ; FILBFR FOR FILE DATA
08E9               589  OUTBUF: DEFS     132        ; BUF FOR OUTPUT MESSAGI
```

SYMBOL	VAL	M	DEFN	REFS						
ACKMSG	0358	R	575	50						
ACKOK	00FE	R	229	220						
B0	02FB	R	550	545						
B1	030D	R	556	553						
B2	0323	R	561	558						
B3	0341	R	568	563						
B4	0357	R	573	570						
CHMORE	001D	R	62	58						
CHRBUF	0369	R	587	315	439					
CLSF	014E	R	284	221	273	275				
COMSY1	000D	R	49	46						
CONDTA	02CC	R	531	440						
CONIN	0001		9	529						
CONINP	01E0	R	438	310						
CONLEN	02CE	R	532	442						
CONOUT	0002		10	537						
CONVEC	02CA	R	529	443						
CORADR	035F	R	581	174	232	234	366	368		
CURBUF	0363	R	584	186	217	379				
ERADR	02F8	R	547	88						
ERFILE	02FC	R	553	96	553					
ERRADR	033E	R	565	198						
FILASV	0202	R	470	118						
FILBFR	03E9	R	588	185	521					
FILCLS	023F	R	495	79	285					
FILDEV	0005		8	470	484	495	519			
FILERR	0057	R	96	63	65	67				
FILOPN	0232	R	484	126						
FILPRM	024B	R	503	75	486					
FILPRP	0252	R	508							
FILPTR	0204	R	472	117						
FILRDV	02BF	R	519	188						
FILRL	024E	R	506							
FILSA	0253	R	509	264	266					
FILSMT	0267	R	511	146						
FILSPV	020F	R	479	125	477	491				
FINL	0121	R	257	161						
GENCKS	01C5	R	412	90	200	372	382			
GENHEX	01C6	R	413	265	267	269	367	369	371	377
HIGADR	02BB	R	513							
LOWADR	02B9	R	512							
MAXRTY	000A		16	214	258					
NACK	0170	R	329	320						
NIBBLE	01CF	R	420	418						
NIBDIG	01DB	R	428	426						
NOOPEN	02E0	R	545	91	545					
NOPROC	030E	R	558	94	558					
NPROC	0052	R	94	81						
NXTBFR	009C	R	177	248						
NXTLIN	00D5	R	206	193	238					
NXTSEG	0084	R	149	250						
OPNB	0044	R	87	71						
OPNFIL	005E	R	116	70						
OPNINP	0000		15	124						
OUTBUF	08E9	R	589	260	362	384	388			
PROC	0080		14	77						
PROCF	0040	R	83	78						
PUTCLS	014B	R	282	202						

```
SYMBOL  VAL  M DEFN REFS

PUTS    005A R   97   92   95
PUTSTR  01F3 R  458   52   97  282
RDBAD   0324 R  563  201  563
RDFILE  007E R  145   84
RDVLEN  02C3 R  522  184  191  245
RECBYE  035B R  579  212  230
REMCNT  0365 R  585  206  211  235
RTRY    0179 R  336  332
RTYCNT  0367 R  586  215  259  330
SEGCNT  0361 R  582  177  183  244
SEGPTR  035D R  580  147  149  158
SKIPSP  0015 R   56   60
STRDTA  02D7 R  539  389  460
STRLEN  02D9 R  540  387  461
STRVEC  02D5 R  537  337  390  462
SYSTEM  0000 X    3   80  119  127  189  286  338  391  444  463
TEKLIN  0182 R  362  218
TKLNLP  019F R  374  378
TKMSFN  01AA R  381  270
TKSWAT  0156 R  309  219  272  322  339
TMUCH   0148 R  278  223
TRYMCH  0342 R  570  279  570
UBIG1   00AB R  184  182
UBIG2   00E4 R  212  210
ZDASGN  0002     20  471
ZDCLOS  0006     22  496
ZDOPEN  0004     21  485
ZDRDST  0040     26
ZDREDA  000C     24  530
ZDREDB  000A     23  520
ZDWRTB  000E     25  538
```

```
*HEADING MAIN ROUTINE

        EXTERNAL        SYSTEM

;PREDEFINED DEVICES


FILDEV  EQU 5           ;DEVICE FOR FILES
CONIN   EQU 1           ;CONSOLE INPUT
CONOUT  EQU 2           ;CONSOLE OUTPUT

; CODE FOR OPEN NEW FILE

OPNEWF  EQU     2       ; OPEN NEW FILE


;DEVICE FUNCTION CODES

ZDASGN  EQU 2           ;ASSIGN
ZDOPEN  EQU 4           ;OPEN
ZDCLOS  EQU 6           ;CLOSE DEVICE
ZDREDB  EQU 0AH         ;READ BINARY
ZDREDA  EQU 0CH         ;READ ASCII LINE
ZDWRTB  EQU 0EH         ;WRITE BINARY

```

```
 LOC   OBJ CODE M STMT SOURCE STATEMENT                          ASM 5.9

                      26  *HEADING MAIN PROGRAM
                      27
                      28
                      29  ; *****************************************************
                      30  ; *
                      31  ; * THIS PROGRAM IS USED TO CREATE A FILE WITH DATA
                      32  ; * RECEIVING FROM THE Z8000 EVALUATION BOARD.
                      33  ; * IF THE FILE HAS ALREADY EXISTED OR UNABLE TO OPEN
                      34  ; * FILE, THIS PROGRAM WILL SEND AN ASCII 9 TO THE
                      35  ; * CONSOLE OUTPUT, AND PROGRAM WILL BE ABORTED.
                      36  ; * IF DATA RECORD WITH TWO LEADING '//' IS RECIVIED,
                      37  ; * THIS PROGRAM WILL ALSO BE ABORTED.
                      38  ; * IF THE CHECKSUM OF THE DATA RECORD RECEIVING FROM
                      39  ; * THE CONSOLE INPUT ARE VERIFIED, AN ASCII 0 WILL BE
                      40  ; * SEND TO THE CONSOLE OUTPUT.  AN ASCII 7 WILL BE SENI
                      41  ; * IF CHECKSUM DOES NOT VERIFY.
                      42  ; *
                      43  ; *****************************************************
                      44
                      45
0000   180B           46          JR      COMSY1
0002   44415445       47          DEFM 'DATE:790629'
                      48
                      49  COMSY1:
000D   E1             50          POP     HL              ; RIO RETURN
000E   E3             51          EX      (SP),HL         ; GET COMMAND POINTER
                      52
000F   7E             53  SKIPSP: LD      A,(HL)
0010   FE20           54          CP      ' '             ; SKIP OVER SPACE
0012   2003           55          JR      NZ, CHMORE
0014   23             56          INC     HL
0015   18F8           57          JR      SKIPSP
                      58
0017   FE3B           59  CHMORE: CP      ';'             ; CHECK FOR DELIMITER
0019   2819           60          JR      Z,SEND9
001B   FE2C           61          CP      ','
001D   2815           62          JR      Z, SEND9
001F   FE0D           63          CP      0DH
0021   2811           64          JR      Z,SEND9
                      65
                      66
0023   CD3800   R     67          CALL    OPNFIL          ; OPEN FILE FOR INPUT
0026   200C           68          JR      NZ, SEND9       ; UNABLE TO OPEN FILE
                      69
                      70
0028   CD5800   R     71          CALL    STRFIL          ; GET DATA FROM CONSOLE
                      72                                  ;  & STORE IN FILE
002B   C8             73          RET     Z               ; FILE CLOSE,EVERYTHING
                      74
                      75  ; RECEIVE ABORT MESSAGE OR HAVING WRITE FILE PROBLEMS
                      76
002C   FD213F03 R     77          LD      IY,FILCLS
0030   CD0000   X     78          CALL    SYSTEM          ; CLOSE FILE
0033   C9             79          RET
                      80
0034   CD8101   R     81  SEND9:  CALL    ABRCKS          ; SEND '9'
0037   C9             82          RET
```

```
                       *HEADING OPNFIL ROUTINE


                       ; ****************************************************
                       ; *
                       ; * OPNFIL ASSIGN A LOGICAL UNIT # TO THE GIVEN
                       ; *        FILENAME AND ALSO OPEN THE FILE
                       ; *
                       ; * INPUT:  HL - POINTER TO FILENAME
                       ; *
                       ; * OUTPUT: RETURN Z IF OPEN FILE SUCCESSFUL
                       ; *         RETURN NZ IF UNABLE TO ASSIGN FILENAME
                       ; *                   OR FILE ALREADY EXISTED
                       ; *
                       ; ****************************************************

                       OPNFIL:
0038    220403     R            LD      (FILPTR), HL     ; FILENAME PTR INTO IY \
003B    FD210203   R            LD      IY, FILASV
003F    CD0000     X            CALL    SYSTEM           ; ASSIGN FILE
0042    FDCB0A76                BIT     6, (IY+10)       ; ASSIGN OK?
0046    C0                      RET     NZ               ; NO


0047    3E02                    LD      A,OPNEWF         ; OPEN TYPE - NEWFILE
0049    320F03     R            LD      (FILSPV),A
004C    FD213203   R            LD      IY, FILOPN       ; OPEN FILE
0050    CD0000     X            CALL    SYSTEM
0053    FDCB0A76                BIT     6, (IY+10)       ; OPEN OK?
0057    C9                      RET                      ; RETURN NZ OR Z
```

```
 LOC   OBJ CODE M STMT SOURCE STATEMENT                                ASM 5.9

                  113  *HEADING STRFIL ROUTINE
                  114
                  115
                  116   ; *****************************************************
                  117   ; *
                  118   ; * STRFIL RECEIVE DATA RECORD FROM CONSOLE INPUT IN
                  119   ; *        TEKTRONIX FORMAT.  VERIFIY TWO CHECKSUMS IN
                  120   ; *        THE RECEIVING DATA, UNPACK DATA TO HEX VALUE
                  121   ; *        AND STORE THEM IN A FILE
                  122   ; *        DATA RECORD WITH ZERO BYTE COUNT INDICATES
                  123   ; *        END OF DATA, THEN THE FILE WILL BE CLOSED.
                  124   ; *        IF DATA RECORD WITH TWO LEADING //, ALSO
                  125   ; *        WILL STOP RECIVING DATA AND CLOSE THE FILE.
                  126   ; *        IF ERROR OCCUR WHILE WRITING DATA INTO FILE,
                  127   ; *        AN ASCII 9 WILL BE SEND OUT TO ABORT THE
                  128   ; *        DATA TRANSFERING.
                  129   ; *
                  130   ; * TEKTRONIX FORMAT:
                  131   ; *     /<CORE ADDRESS(4)><BYTE COUNT(2)><CHECKSUM1(2)>
                  132   ; *      <DATA(2)><DATA(2)> ... <DATA(2)><CHECKSUM2(2)>
                  133   ;
                  134   ; *
                  135   ; * OUTPUT:  RETURN Z FOR SAVING DATA ON FILE
                  136   ; *          RETURN NZ FOR HAVING PROBLEM IN WRITING
                  137   ; *                   DATA ON FILE
                  138   ; *
                  139   ; ***************************************************
                  140
                  141   STRFIL:
0058   210000     142          LD      HL,0
005B   22F603   R 143          LD      (SEGPTR), HL    ; INIT SEGMENT TABLE PTI
005E   216703   R 144          LD      HL,FILSMT
0061   116803   R 145          LD      DE,FILSMT+1
0064   015200     146          LD      BC,82           ; INIT SEGMENT TABLE TO
0067   3600       147          LD      (HL),0
0069   EDB0       148          LDIR
006B   CD3B02   R 149          CALL    BLKBUF          ; BLANK FILBFR WITH FF
                  150
                  151   LODFRO:
006E   CD7501   R 152          CALL    GODCKS          ; SEND ACKNOWLEDGE FOR
                  153   LODF:
0071   CD0201   R 154          CALL    LODREC          ; GET RECORD OF DATA
0074   3A0009   R 155          LD      A,(OUTBUF)
0077   FE2F       156          CP      '/'             ; CHECK IF ERROR RECORD
0079   287E       157          JR      Z,CLSE          ; YES, ABORT & CLOSE FIL
                  158
                  159
                  160   ; VERIFY TWO CHECKSUM & UNPACKED DATA
                  161
007B   210009   R 162          LD      HL, OUTBUF      ; VERIFY ADDR & CNT CKSL
007E   0603       163          LD      B,3
0080   CD8701   R 164          CALL    CHKCKS
0083   380B       165          JR      C, CHKBAD       ; BAD CHECKSUM
0085   78         166          LD      A,B             ; CHECK BYTE COUNT
0086   A7         167          AND     A
0087   280C       168          JR      Z, LODOK        ; COUNT=0, NO DATA CKSUI
                  169
                  170   ; BYTE COUNT NOT ZERO, VERIFY DATA CHECKSUM
```

```
0089   C5                  PUSH   BC              ; SAVE BYTE COUNT
008A   CD8701    R         CALL   CHKCKS          ; CKSUM DATA,B=BYTE COU
008D   C1                  POP    BC
008E   3005                JR     NC, LODOK
                    CHKBAD:
0090   CD7B01    R         CALL   BADCKS          ; SEND NON-ACKNOWLEDGE
0093   18DC                JR     LODF            ; GET DATA RECORD

                    ; CHECKSUM ARE VERIFIED, GENERATE ADDRESS & POINTER FOR
                    ;  FILE ATTRIBUTES, FILE RECORD ..ETC

0095   78           LODOK: LD     A, B            ; BYTE COUNT=0 INDICATE
0096   A7                  AND    A               ;  LAST RECORD
0097   2828                JR     Z, FINLOD       ; YES, LAST RECORD

0099   ED43F403  R         LD     (RECBYE), BC    ; SAVE BYTE COUNT

009D   210009    R         LD     HL,OUTBUF
00A0   CD6B01    R         CALL   GENADL          ; GET ADDRESS FROM RECO
00A3   CD9A01    R         CALL   BUFADM          ; GENERATE BUF ADDRESS
00A6   2806                JR     Z, LODCON       ; NO ERROR IN WRITING F

                    SED9:
00A8   CD8101    R         CALL   ABRCKS          ; SEND '9' TO ABORT
00AB   F601         RETNZ: OR     1               ; RETURN NZ FOR FAILURE
00AD   C9                  RET

                    ; UNPACK DATA IN OUTBUF TO FILBFR
                    ;      HL - FILBFR ADDRESS TO STORE DATA

                    LODCON:
00AE   D9                  EXX
00AF   ED4BF403  R         LD     BC, (RECBYE)    ; BYTE COUNT
00B3   210809    R         LD     HL,OUTBUF+8     ; ADDRESS OF 1ST DATA B
00B6   CD3601    R  LODFR3: CALL  LODBYL          ; CONVERT TO HEX VALUE
00B9   D9                  EXX
00BA   77                  LD     (HL),A          ; SAVE IN FILE BUFF
00BB   23                  INC    HL
00BC   D9                  EXX
00BD   10F7                DJNZ   LODFR3          ; NEXT BYTE
00BF   18AD                JR     LODFR0          ; DONE WITH CURRENT REC
                                                  ;  SEND '0',GET NEXT RE


                    ; RECEIVE LAST DATA RECORD WITH BYTE COUNT=0, SET UP FIL
                    ; DESCRIPTOR RECORD, AND WRITE FILE TO DISK

                    FINLOD:


00C1   210009    R         LD     HL,OUTBUF
00C4   CD6B01    R         CALL   GENADL          ; GENERATE ENTRY ADDRESS
00C7   225303    R         LD     (FILSA),HL      ;  & STORE IN FILE ATTRI
00CA   CDBC02    R         CALL   SETCNT          ; ADJUST CURRENT SEGMENT
                                                  ;  ENTRY 2ND WORD
00CD   DD360000            LD     (IX),0          ; CLEAR ONE SEGMENT ENT
00D1   DD360100            LD     (IX+1),0        ;  TO INDICATE END
```

```
00D5    DD360200     229            LD      (IX+2),0
00D9    DD360300     230            LD      (IX+3),0
00DD    CD4D02     R 231            CALL    TSTMXR          ; FIX UP HIGH ADDRESS
00E0    2ABB03     R 232            LD      HL,(HIGADR)
00E3    2B           233            DEC     HL              ; ADDRESS IS 1 TOO HIGH
00E4    22BB03     R 234            LD      (HIGADR),HL
00E7    CD5D02     R 235            CALL    WRTDSK          ; WRITE CURRENT FILE RE(
00EA    20BC         236            JR      NZ,SED9         ; WRITE FILE ERROR
00EC    CD6F02     R 237            CALL    ADJDSK          ; CHECK ANY MORE DATA
00EF    3805         238            JR      C,CLSS          ; NO MORE DATA,CLOSE FII
00F1    CD5D02     R 239            CALL    WRTDSK          ; MORE DATA TO WRITE
00F4    20B2         240            JR      NZ,SED9         ; WRITE FILE ERROR
00F6    CD7501     R 241    CLSS:   CALL    GODCKS
                     242    CLSE:
00F9    FD21CA03   R 243            LD IY,FILCLP            ;CLOSE PROCEDURE FILE
00FD    CD0000     X 244            CALL    SYSTEM
0100    AF           245            XOR     A               ; RETURN Z FOR OK
0101    C9           246            RET
                     247
```

```
                         *HEADING LODREC ROUTINE


                         ; *******************************************************
                         ; *
                         ; * LODREC RECEIVING AN INPUT LINE FROM CONSOLE INPUT,
                         ; *        MAXIMUM READ 128 CHARACTERS TERMAINATED WITH
                         ; *        CARRIAGE RETURN.  ONLY ACCEPT CHARACTERS
                         ; *        AFTER A '/', AND IGNORE ALL CONTROL CHARACTEI
                         ; *        THE LEADING / WILL BE INCLUDED IN THE BUFFER.
                         ; *
                         ; * OUTPUT:  OUTBUF - CONTAIN DATA STRING
                         ; *
                         ; *******************************************************


0102   CDE002     R      LODREC: CALL    CONINP          ; GET INPUT LINE 128
0105   3AD903     R              LD      A,(CONLEN)      ; # OF CHAR IN INPUT LII
0108   47                        LD      B,A
0109   218009     R              LD      HL, CHRBUF      ; INPUT LINE BUFFER

010C   7E                NCHR:   LD      A, (HL)         ; LOOK FOR '/'
010D   23                        INC     HL
010E   E67F                      AND     7FH
0110   FE2F                      CP      '/'
0112   2807                      JR      Z, GOTLSH

0114   10F6                      DJNZ    NCHR            ; NO '/', CHECK NEXT CHA
0116   CD7B01     R              CALL    BADCKS          ; CAN'T FIND '/', GET NI
0119   18E7                      JR      LODREC          ;   INPUT LINE

                         GOTLSH:
011B   05                        DEC     B               ; CHAR COUNT
011C   110009     R              LD      DE,OUTBUF       ; FILL OUTBUF WITH DATA
011F   7E                NXT:    LD      A,(HL)
0120   23                        INC     HL
0121   E67F                      AND     7FH
0123   12                        LD      (DE),A          ; MOVE CHAR INTO OUTBUF
0124   FE0D                      CP      0DH             ; CHECK END OF LINE
0126   C8                        RET     Z

0127   FE20                      CP      ' '             ; CHECK FOR CONTROL CHAI
0129   3004                      JR      NC, CHROK       ; CHAR OK
012B   10F2                      DJNZ    NXT             ; NEXT CHARACTER
012D   1803                      JR      CEND            ; TERMINATE WITH CR

                         CHROK:
012F   13                        INC     DE
0130   10ED                      DJNZ    NXT

                         CEND:
0132   3E0D                      LD      A,0DH           ; JUST PUT CR AT END
0134   12                        LD      (DE),A
0135   C9                        RET

```

```
                         *HEADING LODBYL ROUTINE

                         ; ******************************************************
                         ; *
                         ; * CONVERT THE NEXT TWO ASCII CHARACTERS IN THE INPUT
                         ; * STREAM TO A 8-BIT HEX VALUE. AFTER EACH ASCII CHAR
                         ; * IS CONVERTED TO 4-BIT HEX VALUE, THE HEX VALUE IS
                         ; * ADDED TO A CHECKSUM ACCUMULATOR.
                         ; *
                         ; * INPUT:  HL - POINTER TO CHARACTER STREAM
                         ; *          C  - CHECKSUM ACCUMULATOR
                         ; *
                         ; * OUTPUT: HL - INCREMENTED BY 2
                         ; *          C  - UPDATED CHECKSUM ACCUMULATOR
                         ; *          A  - 8 BIT HEX VALUE OF TWO ASCII CHAR
                         ; *
                         ; *          RETURN NC  FOR CONVERSION SUCCESSFUL
                         ; *          RETURN C   FOR NON-ASCII CHAR IN BUFFER
                         ; *
                         ; ******************************************************

0136    CD4D01    R      LODBYL: CALL HEXDCD                ;GET CHR AND CHECK
0139    D8                       RET C                      ;BAD
013A    F5                       PUSH AF
013B    81                       ADD A,C
013C    4F                       LD C,A
013D    F1                       POP AF
013E    07                       RLCA
013F    07                       RLCA
0140    07                       RLCA
0141    07                       RLCA
0142    5F                       LD E,A                     ;SAVE PARTIAL BYTE
0143    CD4D01    R              CALL    HEXDCD             ; NEXT BYTE
0146    D8                       RET C
0147    F5                       PUSH AF
0148    81                       ADD A,C
0149    4F                       LD C,A
014A    F1                       POP AF
014B    B3                       OR      E                  ; MERGE TWO HEX VALUE
                                                            ; OR SET NC FOR OK RETUF
014C    C9                       RET



                         ; ******************************************************
                         ; *
                         ; * HEXDCD CONVERT A SINGLE ASCII CHARACTER TO A 4-BIT
                         ; *         HEX VALUE.  ( 41 -> A,  37 -> 7 )
                         ; *
                         ; * INPUT:  HL - POINTER TO ASCII CHARACTER
                         ; *
                         ; * OUTPUT: HL - INCREMENTED BY 1
                         ; *          A  - 4-BITS HEX VALUE
                         ; *
                         ; *          RETURN NC FOR CONVERSION SUCCESSFUL
                         ; *          RETURN C  FOR CHARACTER IS NON-ASCII
                         ; *
                         ; ******************************************************
```

```
014D  7E            HEXDCD: LD      A, (HL)         ; GET CHARACTER
014E  23                    INC     HL
014F  FE5B                  CP      'Z'+1           ; CHECK FOR LOWER CASE
0151  3802                  JR      C, HEXUPR       ; NO
0153  CBAF                  RES     5,A             ; FORCE TO UPPER CASE
0155  FE30          HEXUPR: CP      '0'
0157  D8                    RET     C               ; NOT 0-9
0158  FE47                  CP      'F'+1
015A  300D                  JR      NC, HEXBAD      ; NOT A-F
015C  FE3A                  CP      '9'+1
015E  3806                  JR      C,USEDIG        ; IS DIGIT
0160  FE41                  CP      'A'
0162  3805                  JR      C,HEXBAD
0164  D637                  SUB     'A'-10
0166  E60F          USEDIG: AND     0FH
0168  C9                    RET                     ; RETURN NC FROM AND

0169  37            HEXBAD: SCF                     ; RETURN C FOR BAD CHAR
016A  C9                    RET

                    ; *****************************************************
                    ; *
                    ; * GENADL CONVERT 4 ASCII CHARACTERS TO A 16 BIT HEX
                    ; *         VALUE IN HL.
                    ; *
                    ; * INPUT:  HL - POINTER TO FIRST ASCII CHARACTER
                    ; *
                    ; * OUTPUT: HL - 16-BIT HEX VALUE
                    ; *
                    ; *****************************************************

016B  CD3601    R   GENADL: CALL LODBYL
016E  57                    LD D,A
016F  CD3601    R           CALL LODBYL
0172  5F                    LD E,A
0173  EB                    EX DE,HL
0174  C9                    RET

                    ; *****************************************************
                    ; *
                    ; * GODCKS SEND ASCII 0 FOLLOWED BY CARRIAGE RETURN FOR
                    ; * ACKNOWLEDGE.
                    ; *
                    ; * BADCKS SEND ASCII 7 FOLLOWED BY CARRIAGE RETURN FOR
                    ; * NON-ACKNOWLEDGE.
                    ; *
                    ; * ABRCKS SEND ASCII 9 FOLLOWED BY CARRIAGE RETURN FOR
                    ; * EITHER ERROR IN OPEN FILE OR ERROR IN WRITE DATA
                    ; * ON DISK FILE.
                    ; *
                    ; *****************************************************
```

```
0175   21EB03   R  419  GODCKS: LD     HL,ACKMSG        ; SEND ASCII '0'
0178   C3F302   R  420          JP     PUTSTR
                   421
017B   21EE03   R  422  BADCKS: LD     HL,NAKMSG        ; SEND ASCII '7'
017E   C3F302   R  423          JP     PUTSTR
                   424
0181   21F103   R  425  ABRCKS: LD     HL,ABRMSG        ; SEND ASCII '9'
0184   C3F302   R  426          JP     PUTSTR
```

```
                        427  *HEADING CHKCKS ROUTINE
                        428
                        429
                        430  ; ********************************************************
                        431  ;
                        432  ; CHKCKS VERIFIES RECORD CHECKSUM.  FIRST CONVERT THE AS
                        433  ;  CHARACTERS TO THEIR CORRESPONDING HEX VALUE, CALCULA
                        434  ;  THE CHECKSUM BY ADDING THESE HEX VALUES, THEN CONVERT
                        435  ;  THE ASCII RECORD CHECKSUM WHICH FOLLOWED THE DATA TO
                        436  ;  HEX VALUES AND COMPARE TO THE CALCULATED CHECKSUM.
                        437  ;
                        438  ;  INPUT:   HL - POINTER TO ASCII STRING
                        439  ;           B  - # OF CHARACTER PAIR TO BE CHECKSUM
                        440  ;                ( TOTAL CHARACTERS TO CHECKSUM IS (B*2)
                        441  ;
                        442  ; OUTPUT:   RETURN NC IF CHECKSUM VERIFY
                        443  ;           RETURN C  IF INCORRECT CHECKSUM
                        444  ;
                        445  ; ********************************************************
                        446
                        447
0187   0E00             448  CHKCKS: LD C,0                       ;INITIAL CKSUM ACCUMULA
0189   CD3601     R     449  CK1:    CALL LODBYL                  ; KEEPS C UPDATED
018C   D8               450          RET C                        ;ERROR, NON-ASCII CHAR
018D   10FA             451          DJNZ CK1
018F   47               452          LD B,A
0190   C5               453          PUSH BC                      ;SAVE ACCUMULATED CHECKS
0191   CD3601     R     454          CALL    LODBYL               ; CONVERT STORED CKSUM
0194   C1               455          POP BC
0195   D8               456          RET C                        ;ERROR, NON-ASCII CHAR
0196   B9               457          CP      C                    ; COMPARE STORED CKSUM (
                        458                                       ;  WITH CALC.CKSUM (C)
0197   C8               459          RET Z                        ;OK
0198   37               460          SCF                          ;BAD CHECKUSM, FORCE ERF
0199   C9               461          RET
```

```
                      *HEADING BUFADM ROUTINE
                      ; *****************************************************;
                      ;
                      ; BUFADM BREAKS UP THE LOAD STREAM INTO DISK SEGMENTS.
                      ;
                      ; INPUT:  HL- CORE ADDRESS IN LOAD RECORD
                      ;         B - BYTE COUNT IN LOAD RECORD
                      ;
                      ; OUTPUT: HL - FILBFR ADDRESS TO STORE DATA
                      ;
                      ; OUTPUT:  RETURN Z IF NO ERROR IN FILE WRITING
                      ;          RETURN NZ IF PROBLEMS IN FILE WRITING
                      ;
                      ; IT USES FLAG (SEGPTR=0) TO INDICATE THAT THIS IS THE
                      ; FIRST SEGMENT.
                      ;
                      ; *****************************************************;
                      BUFADM:
019A  22F803    R             LD      (CORADR),HL     ; CORE ADDRESS
019D  ED5BF603  R             LD      DE,(SEGPTR)     ; CHECK IF FIRST TIME
01A1  7A                      LD      A,D
01A2  B3                      OR      E
01A3  201B                    JR      NZ,NOTFST       ; NOT FIRST TIME

                      ; INIT VARIABLES:
                      ;       SEGPTR - SEGMENT TABLE POINTER
                      ;       HIGADR - HIGH ADDRESS FOR FILE ATTRIBUTES
                      ;       LOWADR - LOW ADDRESS FOR FILE ATTRIBUTES
                      ;       BOTRNG - LOW ADDRESS OF CURRENT FILE RECORD
                      ;       TOPRNG - TOP ADDRESS OF CURRENT FILE RECORD
                      ;       TOPUSD - FILBFR ADDR. WHERE LAST BYTE DATA STORE
                      ;
                      ;  ALSO SET UP ENTRY IN SEGMENT TABLE
                      ;       1ST WORD - LOW CORE ADDRESS OF CURRENT SEGMENT
                      ;       2ND WORD - HIGH CORE ADDRESS OF CURRENT SEGMENT
                      ;                (TEMPORARY, SHOULD SET TO SEGMENT SIZ

01A5  116703    R             LD      DE,FILSMT
01A8  ED53F603  R             LD      (SEGPTR),DE     ; SEGMENT TABLE POINTER

01AC  CDA602    R             CALL    GENRNG          ; SET UP ADDRESSES FOR F
                                                      ;  RECORD, SEGMENT TABLE
01AF  22BB03    R             LD      (HIGADR),HL     ; ADDRESS FOR FILE ATTRI
01B2  2AFA03    R             LD      HL,(BOTRNG)
01B5  22B903    R             LD      (LOWADR),HL

                      BLKBFR:
01B8  CD3B02    R             CALL    BLKBUF          ; INIT FILBFR WITH FF
01BB  210004    R             LD      HL,FILBFR
01BE  1816                    JR      COMADM          ; FIGURE HOW FAR TO FILL
                                                      ;  FILBFR (TOPUSD)

                      ; CHECK IF CORE ADDRESS OF RECEIVING DATA IS BEYOND THE
```

```
 LOC    OBJ CODE M STMT SOURCE STATEMENT                                   ASM 5.9
                    520  ;  RANGE OF CURRENT FILE RECORD ADDRESS, IF SO, WRITE
                    521  ;  OUT DATA LEFT IN FILBFR TO FILE, AND CREATE NEW SEGMI
                    522
                    523
01C0   ED5BFC03 R   524  NOTFST: LD      DE,(TOPRNG)      ; TOP ADDRESS OF CURREN
01C4   ED52         525          SBC     HL,DE            ;  FILE RECORD
01C6   302C         526          JR      NC,NXTRNG        ; NEED TO WRITE FILE
                    527
                    528  ; FIGURE THE ADDRESS OF FILBFR FOR STORING RECEIVING DA
                    529
01C8   A7           530          AND     A
01C9   2AF803   R   531          LD      HL,(CORADR)      ; CORE ADDRESS IN RECORI
01CC   ED5BFA03 R   532          LD      DE,(BOTRNG)
01D0   ED52         533          SBC     HL,DE
01D2   110004   R   534          LD      DE,FILBFR
01D5   19           535          ADD     HL,DE            ; FILBFR ADDRESS FOR NE
                    536                                   ;  BYTE = (CORADR)-(BOTI
                    537                                   ;        +FILBFR
                    538
                    539  ; COMPUTE THE TOP ADDRESS OF FILBFR FOR STORING THE LAS
                    540  ;   RECEIVING DATA IN  (TOPUSD)
                    541  ; ALSO SET UP THE CORE ADDRESS OF THE LAST RECEIVING
                    542  ;  DATA IN 2ND WORD OF SEGMENT TABLE ENTRY
                    543
                    544  COMADM:
01D6   ED4BF403 R   545          LD      BC,(RECBYE)
01DA   48           546          LD      C,B              ; BYTE COUNT
01DB   0600         547          LD      B,0
01DD   0B           548          DEC     BC
01DE   E5           549          PUSH    HL               ; FILBFR ADDR FOR NEXT I
01DF   09           550          ADD     HL,BC
01E0   22FE03   R   551          LD      (TOPUSD),HL      ; ADDRESS OF FILBFR FOR
                    552                                   ;  STORING LAST DATA
                    553
01E3   2AF803   R   554          LD      HL,(CORADR)      ; CORD ADDRESS
01E6   09           555          ADD     HL,BC
01E7   DD2AF603 R   556          LD      IX,(SEGPTR)
01EB   DD7502       557          LD      (IX+2),L         ; SET 2ND WORD SEGMENT E
01EE   DD7403       558          LD      (IX+3),H         ;  WITH ADDRESS OF LAST
                    559                                   ;  DATA OF RECEIVING BUI
01F1   E1           560          POP     HL               ; FILBFR ADDR FOR 1ST DA
01F2   BF           561          CP      A                ; RETURN Z FOR OK
01F3   C9           562          RET
                    563
                    564
                    565  ; WRITE OUT DATA IN FILBFR ( MAX.80 BYTES), AND MOVE UP
                    566  ;  THE REST OF DATA, RECOMPUTE ADDRESS OF NEXT FILE RECC
                    567  ;  IN  (BOTRNG)  & (TOPRNG)
                    568
                    569  NXTRNG:
01F4   CD5D02   R   570          CALL    WRTDSK
01F7   C0           571          RET     NZ               ; WRITE FILE ERROR
                    572
                    573
01F8   CD6F02   R   574          CALL    ADJDSK           ; MOVE DATA UP TO FILBFI
01FB   08           575          EX      AF,AF'           ; SAVE CARRY FOR BUF EMI
                    576
01FC   2AFC03   R   577          LD      HL,(TOPRNG)      ; RESET ADDRESS OF NEW
```

```
01FF    22FA03    R  578            LD      (BOTRNG),HL      ;  FILE RECORD
0202    ED5B4E03  R  579            LD      DE,(FILRL)       ; RECORD LENGTH
0206    19           580            ADD     HL,DE
0207    22FC03    R  581            LD      (TOPRNG),HL      ; TOP ADDR FOR NEW FILE
                     582
                     583   ; CHECK IF CORE ADDRESS OF RECEIVING DATA IS BEYOND THE
                     584   ;  RANGE OF THE NEXT NEW FILE RECORD ADDRESSES, IF NOT,
                     585   ;  KEEP FILLING DATA IN FILBFR
                     586
020A    DD2AF603  R  587            LD      IX,(SEGPTR)
020E    DD6E02       588            LD      L,(IX+2)
0211    DD6603       589            LD      H,(IX+3)
0214    19           590            ADD     HL,DE            ; CORE ADDRESS FOR LAST
                     591                                     ; BYTE + 80H
0215    ED5BF803  R  592            LD      DE,(CORADR)      ; CORE ADDR OF RECIVING
0219    A7           593            AND     A
021A    ED52         594            SBC     HL,DE
021C    EB           595            EX      DE,HL            ; HL = CORE ADDR
021D    3806         596            JR      C, WRTS          ; WRITE OUT DATA
                     597
021F    08           598            EX      AF,AF'
0220    DC3B02    R  599            CALL    C,BLKBUF         ; BLANK FILBFR IF EMPTY
0223    189B         600            JR      NOTFST           ; USE SAME FILE RECORD
                     601
                     602   ; ADDRESS OF NEXT RECORD DATA IS NOT IN THE RANGE OF
                     603   ;  CURRENT FILE RECORD, WRITE OUT DATA IN FILBFR, AND
                     604   ;  CREATE A NEW SEGMENT
                     605
                     606   WRTS:
0225    08           607            EX AF,AF'                ;WAS THERE MORE IN BUFFE
0226    3804         608            JR C,NXTRN2              ;NO
0228    CD5D02    R  609            CALL WRTDSK              ;GET RID OF REMAINDER
022B    C0           610            RET     NZ               ; WRITE FILE ERROR
                     611
022C    CDBC02    R  612   NXTRN2:  CALL    SETCNT           ; SET SEGMENT SIZE IN SE
                     613                                     ;  MENT TABLE
022F    2AF803    R  614            LD      HL,(CORADR)
0232    CDA602    R  615            CALL    GENRNG           ; SET FILE RECORD ADDRES
0235    CD4D02    R  616            CALL    TSTMXR           ; SET HIGHADDRESS FOR FI
                     617                                     ;  ATTRIBUTES
0238    C3B801    R  618            JP      BLKBFR           ; BLANK FILBFR & START
                     619
```

```
                    620  *HEADING BLKBUF, TSTMXR, WRTDSK, ADJDSK
                    621
                    622  ; *****************************************************
                    623  ;
                    624  ; BLKBUF FILL THE FILBFR BUFFER WITH DATA FF.
                    625  ;
                    626  ; *****************************************************
023B  E5            627  BLKBUF: PUSH    HL
023C  C5            628          PUSH    BC
023D  210004     R  629          LD      HL,FILBFR       ; BUFFER ADDR
0240  110104     R  630          LD      DE,FILBFR+1
0243  01FF03        631          LD      BC, 3FFH        ; SIZE OF BUFF
0246  36FF          632          LD      (HL),0FFH       ; FILL 1ST BYTE WITH FF
0248  EDB0          633          LDIR
024A  C1            634          POP     BC
024B  E1            635          POP     HL
024C  C9            636          RET
                    637
                    638
                    639  ; *****************************************************
                    640  ;
                    641  ; TSTMXR SET THE HIGH ADDRESS FOR THE FILE ATTRIBUTES.
                    642  ;  IF (TOPUSD) TOP ADDRESS OF CURRENT FILE RECORD IS
                    643  ;  GREATER THAN THEVALUE IN(HIGADR),THEN SET HIGADR=TOP
                    644  ;
                    645  ; *****************************************************
                    646
                    647
024D  2ABB03     R  648  TSTMXR: LD HL,(HIGADR)
0250  ED5BFC03   R  649          LD DE,(TOPRNG)
0254  A7            650          AND A
0255  ED52          651          SBC HL,DE
0257  D0            652          RET NC                  ;OK
0258  ED53BB03   R  653          LD (HIGADR),DE          ;RESET
025C  C9            654          RET
                    655
                    656
                    657  ; *****************************************************
                    658  ;
                    659  ; WRTDSK WRITE THE FIRST 80 BYTES OF DATA IN FILBFR
                    660  ;  BUFFER TO THE DISK.
                    661  ;
                    662  ; OUTPUT: RETURN NZ FOR WRITE FILE ERROR
                    663  ;         RETURN Z FOR NO ERROR
                    664  ;
                    665  ; *****************************************************
                    666
                    667
025D  2A4E03     R  668  WRTDSK: LD HL,(FILRL)
0260  22C303     R  669          LD      (FILDL),HL      ; FILE RECORD LENGTH
0263  FD21BF03   R  670          LD      IY,FILVEC
0267  CD0000     X  671          CALL    SYSTEM          ; WRITE DATA TO FILE
026A  FDCB0A76      672          BIT     6,(IY+10)
026E  C9            673          RET                     ; RETURN Z FOR NO ERROR
                    674                                  ; RETURN NZ FOR WRITE E
                    675
                    676
                    677  ; *****************************************************
```

```
                      ;
                      ; ADJDSK DELETES THE DATA IN FILBFR THAT HAS BEEN WRITTI
                      ; TO FILE, THEN MOVES UP THE REST OF DATA AND BLANK OUT
                      ; THE UNUSED PART OF FILBFR.  ALSO RESET  (TOPUSD)
                      ; FOR ADDRESS OF FILBFR LAST USED.
                      ;
                      ; OUTPUT:  RETURN NC IF THERE IS VALID DATA IN FILBFR
                      ;              RETURN C IF FILBFR EMPTY
                      ;
                      ; ****************************************************:


026F   2AFE03   R     ADJDSK: LD HL,(TOPUSD)
0272   110004   R             LD DE,FILBFR
0275   A7                     AND A
0276   ED52                   SBC HL,DE         ; # OF BYTE OF DATA IN FILBFR
0278   D8                     RET C             ; NO DATA IN FILBFR

0279   ED5B4E03 R             LD DE,(FILRL)     ;THIS MUCH WAS WRITTEN
027D   ED52                   SBC HL,DE
027F   D8                     RET C             ;NOTHING MORE IN FILBFR

                      ; MOVE UP DATA

0280   23                     INC HL
0281   44                     LD B,H
0282   4D                     LD C,L            ;READY FOR LDIR
0283   2A4E03   R             LD HL,(FILRL)
0286   110004   R             LD DE,FILBFR
0289   19                     ADD HL,DE         ;FROM TOP TO BOTTOM
028A   C5                     PUSH    BC        ; # OF BYTE GOT MOVE
028B   EDB0                   LDIR
028D   C1                     POP BC
028E   21FF03                 LD HL,400H-1
0291   A7                     AND A             ; BLANK REST OF FILBFR WITH FF
0292   ED42                   SBC HL,BC
0294   44                     LD B,H
0295   4D                     LD C,L
0296   D5                     PUSH DE           ; FILBFR ADDRESS FOR NEXT DATA E

0297   62                     LD H,D
0298   6B                     LD L,E
0299   36FF                   LD (HL),0FFH
029B   13                     INC DE
029C   EDB0                   LDIR              ;CLEAR REMAINDER WITH FF
029E   D1                     POP DE
029F   1B                     DEC DE
02A0   ED53FE03 R             LD (TOPUSD),DE    ; DATA ARE FILL UP TO THIS ADDRE
02A4   A7                     AND A             ;FORCE NO CARRY ON RETURN
02A5   C9                     RET


                      ; ****************************************************
                      ;
                      ; GENRNG SET UP SEGMENT ADDRESS IN SEGMENT TABLE. ALSO
                      ;  SET   BOTRNG - LOW ADDRESS OF NEW FILE RECORD
                      ;        TOPRNG - HIGH ADDRESS OF NEW FILE RECORD
                      ;
```

```
                         ; INPUT:  HL - CORE ADDRESS
                         ;         SEGPTR - POINTER TO SEGMENT TABLE
                         ;
                         ; ******************************************************:

02A6    22FA03     R     GENRNG: LD (BOTRNG),HL
02A9    DD2AF603   R             LD IX,(SEGPTR)
02AD    DD7401                   LD (IX+1),H
02B0    DD7500                   LD (IX),L
02B3    ED5B4E03   R             LD DE,(FILRL)
02B7    19                       ADD HL,DE               ;TOP
02B8    22FC03     R             LD (TOPRNG),HL
02BB    C9                       RET



                         ; ******************************************************
                         ;
                         ; SETCNT COMPUTES THE SEGMENT SIZE IN THE SEGMENT TABLE
                         ;   THE SECOND WORD IN THE SEGMENT TABLE HAS BEEN KEPT
                         ;   UPDATED WITH THE HIGHEST ADDRESS-1 SO FAR.  IT SHOUI
                         ;   REPLACE WITH THE ACTUAL SEGMENT SIZE.
                         ;
                         ; INPUT: SEGPTR - POINTER TO THE SEGMENT TABLE
                         ;
                         ; OUTPUT: SEGPTR - UPDATED BY 4 TO POINT TO NEXT SEGMEN
                         ;
                         ; ******************************************************


02BC    DD2AF603   R     SETCNT: LD IX,(SEGPTR)
02C0    DD6603                   LD H,(IX+3)
02C3    DD6E02                   LD L,(IX+2)             ; 1ST WORD OF SEGMENT
02C6    DD5601                   LD D,(IX+1)
02C9    DD5E00                   LD E,(IX)               ; 2ND WORD OF SEGMENT
02CC    A7                       AND A
02CD    ED52                     SBC HL,DE
02CF    23                       INC HL                  ; CONVERT TO SIZE
02D0    DD7403                   LD (IX+3),H
02D3    DD7502                   LD (IX+2),L
02D6    110400                   LD DE,4
02D9    DD19                     ADD IX,DE
02DB    DD22F603   R             LD (SEGPTR),IX          ; UPDATE SEGMENT TBL POI
02DF    C9                       RET
```

```
                  *HEADING SUBROUTINE.S


                  ; ****************************************************:
                  ;
                  ; CONINP GET INPUT LINE FROM CONSOLE INPUT
                  ;
                  ; ****************************************************:


                  CONINP:
02E0  218009   R          LD      HL, CHRBUF       ; RECEIVE CHARACTER STR:
02E3  22D703   R          LD      (CONDTA),HL
02E6  218000              LD      HL,CHRSIZ        ; SIZE OF INPUT CHAR
02E9  22D903   R          LD      (CONLEN),HL      ; LENGTH OF BUF
02EC  FD21D503 R          LD      IY, CONVEC
02F0  C30000   X          JP      SYSTEM



                  ; ****************************************************
                  ;
                  ; PUTSTR OUTPUT A STRING OF CHARACTER TO CONSOLE OUTPUT
                  ;
                  ; INPUT:  HL - POINT TO A MESSAGE IN DEFT FORMAT
                  ;
                  ; ****************************************************


02F3  7E          PUTSTR: LD A,(HL)                ;GET COUNT
02F4  23                  INC HL
02F5  22E203   R          LD (STRDTA),HL
02F8  32E403   R          LD (STRLEN),A            ;OTHER PART IS ALWAYS 0
02FB  FD21E003 R          LD IY,STRVEC
02FF  C30000   X          JP SYSTEM

```

```
 LOC    OBJ CODE M STMT SOURCE STATEMENT

                   816  *HEADING SYSTEM CALL VECTOR
                   817
                   818
                   819  ; ASSIGN FILE
                   820
0302    05         821  FILASV: DEFB    FILDEV
0303    02         822          DEFB    ZDASGN          ; ASSIGN REQUEST CODE
0304    0000       823  FILPTR: DEFW    0               ; FILENAME POINTER
0306    0000       824          DEFW    0
0308    0000       825          DEFW    0
030A    0000       826          DEFW    0
030C    00         827          DEFB    0
030D    0F03     R 828          DEFW    FILSPV          ; SUPPL.PARAMETER VECTO
                   829
030F    00         830  FILSPV: DEFB    0               ; FLAG BYTE
0310               831          DEFS    34
                   832
                   833  ; OPEN FILE
                   834
0332    05         835  FILOPN: DEFB    FILDEV          ; SAME UNIT AS ASSIGN
0333    04         836          DEFB    ZDOPEN          ; OPEN REQUEST CODE
0334    4B03     R 837          DEFW    FILPRM          ; FILE ATTRIBUTES AREA
0336    7400       838          DEFW    116             ; LENGTH OF FILE ATRIBU
0338    0000       839          DEFW    0
033A    0000       840          DEFW    0
033C    00         841          DEFB    0
033D    0F03     R 842          DEFW    FILSPV          ; SUPPL.PARAMETER VECTO
                   843
                   844  ; CLOSE FILE
                   845
033F    05         846  FILCLS: DEFB    FILDEV          ; SAME UNIT AS ASSIGN
0340    06         847          DEFB    ZDCLOS          ; CLOSE REQUEST CODE
0341    0000       848          DEFW    0
0343    0000       849          DEFW    0
0345    0000       850          DEFW    0
0347    0000       851          DEFW    0
0349    0000       852          DEFW    0
                   853
                   854  FILPRM:
034B    80         855          DEFB    80H     ; FILE TYPE ( MUST BE PROC.)
034C    0000       856          DEFW    0       ; SIZE
034E    0000       857  FILRL:  DEFW    00H     ; RECORD LENGTH,DEFAULT
0350    0000       858          DEFW    0       ; BLOCKING
0352    00         859          DEFB    0       ; PROPERTIES
0353    0000       860  FILSA:  DEFW    0       ; STARTING ADDRESS
0355    0002       861          DEFW    200H    ; BYTE IN LAST RECORD
                   862                          ; WILL CHANGE TO 80H IF FLOPPY
0357               863          DEFS    16      ; CREATION DATE
0367               864  FILSMT: DEFS    82      ; SEGMENT POINTER TABLE
03B9    0000       865  LOWADR: DEFW    0       ; LOWEST ADDRESS USED
03BB    0000       866  HIGADR: DEFW    0       ; HIGHEST ADDRESS UES
03BD    8000       867          DEFW    80H     ; STACK SIZE
                   868
                   869
                   870  ;FILE WRITE VECTOR
                   871
03BF    05         872  FILVEC: DEFB FILDEV
03C0    0E         873          DEFB ZDWRTB
```

```
 LOC    OBJ CODE M STMT SOURCE STATEMENT
03C1    0004     R                DEFW FILBFR
03C3                     FILDL:   DEFS 2
03C5    0000                      DEFW 0
03C7    0000                      DEFW 0
03C9    00                        DEFB 0

                         ; CLOSE FOR PROCEDURE FILE

03CA    05               FILCLP:  DEFB    FILDEV
03CB    06                        DEFB    ZDCLOS
03CC    4B03     R                DEFW    FILPRM
03CE    7400                      DEFW    116
03D0    0000                      DEFW    0
03D2    0000                      DEFW    0
03D4    00                        DEFB    0


                         ;CONSOLE READ AND WRITE

03D5    01               CONVEC:  DEFB CONIN
03D6    0C                        DEFB ZDREDA
03D7    0000             CONDTA:  DEFW 0
03D9    0000             CONLEN:  DEFW 0
03DB    0000                      DEFW 0
03DD    0000                      DEFW 0
03DF    00                        DEFB 0

03E0    02               STRVEC:  DEFB CONOUT
03E1    0E                        DEFB ZDWRTB
03E2    0000             STRDTA:  DEFW 0
03E4    0000             STRLEN:  DEFW 0
03E6    0000                      DEFW 0
03E8    0000                      DEFW 0
03EA    00                        DEFB 0

03EB    02               ACKMSG:  DEFB    2
03EC    30                        DEFB    '0'
03ED    0D                        DEFB    0DH
03EE    02               NAKMSG:  DEFB    2
03EF    37                        DEFB    '7'
03F0    0D                        DEFB    0DH
03F1    02               ABRMSG:  DEFB    2
03F2    39                        DEFB    '9'
03F3    0D                        DEFB    0DH


03F4    0000             RECBYE:  DEFW    00          ; RECORD BYTE COUNT
03F6    0000             SEGPTR:  DEFW    0           ; SEGMENT TABLE POINTER
03F8    0000             CORADR:  DEFW    0           ; ADDRESS IN LOAD RECOR
03FA    0000             BOTRNG:  DEFW    0           ; LOW ADDRESS OF CURREN
                                                      ;  FILE RECORD
03FC    0000             TOPRNG:  DEFW    0           ; HIGH ADDRESS OF CURRE
                                                      ;  FILE RECORD
03FE    0000             TOPUSD:  DEFW    0           ; TOP ADDRESS OF FILBFR
0400                     FILBFR:  DEFS    500H        ; FILE BUFFER
0900                     OUTBUF:  DEFS    128         ; BUF WITH TEKTRONIX FO
0980                     CHRBUF:  DEFS    128         ; INPUT BUF FROM CONSOL
                         CHRSIZ   EQU     $-CHRBUF
```

```
SYMBOL  VAL M DEFN REFS

ABRCKS 0181 R  425   81  195
ABRMSG 03F1 R  915  425
ACKMSG 03EB R  909  419
ADJDSK 026F R  690  237  574
BADCKS 017B R  422  177  276
BLKBFR 01B8 R  513  618
BLKBUF 023B R  627  149  514  599
BOTRNG 03FA R  923  509  532  578  741
BUFADM 019A R  482  191
CEND   0132 R  298  292
CHKBAD 0090 R  176  165
CHKCKS 0187 R  448  164  173
CHMORE 0017 R   59   55
CHRBUF 0980 R  930  267  791  931
CHROK  012F R  294  290
CHRSIZ 0080    931  793
CK1    0189 R  449  451
CLSE   00F9 R  242  157
CLSS   00F6 R  241  238
COMADM 01D6 R  544  516
COMSY1 000D R   49   46
CONDTA 03D7 R  895  792
CONIN  0001      9  893
CONINP 02E0 R  790  264
CONLEN 03D9 R  896  265  794
CONOUT 0002     10  901
CONVEC 03D5 R  893  795
CORADR 03F8 R  922  483  531  554  592  614
FILASV 0302 R  821  101
FILBFR 0400 R  928  515  534  629  630  691  706  874
FILCLP 03CA R  882  243
FILCLS 033F R  846   77
FILDEV 0005      8  821  835  846  872  882
FILDL  03C3 R  875  669
FILOPN 0332 R  835  109
FILPRM 034B R  854  837  884
FILPTR 0304 R  823  100
FILRL  034E R  857  579  668  696  705  745
FILSA  0353 R  860  224
FILSMT 0367 R  864  144  145  503
FILSPV 030F R  830  108  828  842
FILVEC 03BF R  872  670
FINLOD 00C1 R  219  185
GENADL 016B R  396  190  223
GENRNG 02A6 R  741  506  615
GODCKS 0175 R  419  152  241
GOTLSH 011B R  279  273
HEXBAD 0169 R  379  370  374
HEXDCD 014D R  362  324  335
HEXUPR 0155 R  367  365
HIGADR 03BB R  866  232  234  508  648  653
LODBYL 0136 R  324  206  396  398  449  454
LODCON 00AE R  202  192
LODF   0071 R  153  178
LODFRO 006E R  151  212
LODFR3 00B6 R  206  211
LODOK  0095 R  183  168  175
LODREC 0102 R  264  154  277
```

SYMBOL	VAL	M	DEFN	REFS						
LOWADR	03B9	R	865	510						
NAKMSG	03EE	R	912	422						
NCHR	010C	R	269	275						
NOTFST	01C0	R	524	487	600					
NXT	011F	R	282	291	296					
NXTRN2	022C	R	612	608						
NXTRNG	01F4	R	569	526						
OPNEWF	0002		14	107						
OPNFIL	0038	R	99	67						
OUTBUF	0900	R	929	155	162	189	205	222	281	
PUTSTR	02F3	R	809	420	423	426				
RECBYE	03F4	R	920	187	204	545				
RETNZ	00AB	R	196							
SED9	00A8	R	194	236	240					
SEGPTR	03F6	R	921	143	484	504	556	587	742	766	778
SEND9	0034	R	81	60	62	64	68			
SETCNT	02BC	R	766	225	612					
SKIPSP	000F	R	53	57						
STRDTA	03E2	R	903	811						
STRFIL	0058	R	141	71						
STRLEN	03E4	R	904	812						
STRVEC	03E0	R	901	813						
SYSTEM	0000	X	3	78	102	110	244	671	796	814
TOPRNG	03FC	R	925	524	577	581	649	747		
TOPUSD	03FE	R	927	551	690	725				
TSTMXR	024D	R	648	231	616					
USEDIG	0166	R	376	372						
WRTDSK	025D	R	668	235	239	570	609			
WRTS	0225	R	606	596						
ZDASGN	0002		19	822						
ZDCLOS	0006		21	847	883					
ZDOPEN	0004		20	836						
ZDREDA	000C		23	894						
ZDREDB	000A		22							
ZDWRTB	000E		24	873	902					

APPENDIX D

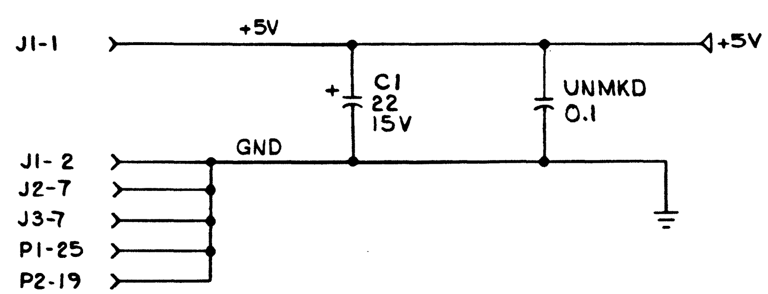
Z-SCAN 8000 SCHEMATICS

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
A	ECN 2120	6-18-81	

DEVICE TABLE							
TYPE	+5V	-5V	+12V	-12V	GND	REF DES	UNUSED OUTPUTS
74LS04	14				7	U1,61,128,138	U128-10,12
74LS32	14				7	U2,46,64,95,108,155	
74LS161	16				8	U3,70,71,136	
74LS373	20				10	U4,6,84-87	
74S85	16				8	U5,8,9,11,12,22,23,30,101,103,104	
74S373	20				10	U7,105	
74LS175	16				8	U10,13	
1489	14				7	U14,35	
1488	—		14	1	7	U15,16	U15-3: U16-3
74LS125	14				7	U17	
74S260	14				7	U18	
74S00	14				7	U140	U140-8,11
74S32	14				7	U20,149	U149- 8,11, U20-6
74S02	14				7	U96,126	U126-4
74LS00	14				7	U24,63,133,135,19	U19-6
74LS138	16				8	U25,26,36,47,49	
74LS74	14				7	U28,29,58,68,92,94	U68-8,9
Z80A-PIO	26				11	U31-33,73-75	
Z80A-SIO/2	9				31	U34	
TL497	14				5	U37	
40L44-20	18				9	U38-45,50-57	
74S133	16				8	U48,67	
74LS10	14				7	U59,60,132,137	U132-6: U137-8,12
74LS08	14				7	U62,65,131	
74LS20	14				7	U66,109	U109-6
74S10	14				7	U69	
Z80A-CTC	24				5	U72	
2114-2	18				9	U76-79,151-154	
(2732) PROM	24				12	U81-83,89-91	
74S11	14				7	U93	
93S16	16				8	U98,99	
OSC 19.6608MHZ	14				7	U100	
4702	16				8	U102	
74LS241	20				10	U106,107,117-123,145,146,148,150	
74S08	14				7	U111,129	U129-3,8
74S64	14				7	U112	
74LS253	16				8	U113-116	
Z8002-CPU	10				31	U124	
Z8001-CPU	11				36	U125	
74LS02	14				7	U21	
74S240	20				10	U127	U127-3,5,7,9,18
74S241	20				10	U134	
Z8000 CLK DRVR	1				5	U139	
RES NTWK 33Ω	—				—	U142-144,147	
RES NTWK 10K	16				—	U27,97,130,141	
74S74	14				7	U110	

NOTES. UNLESS OTHERWISE SPECIFIED:

1. ALL RESISTANCE VALUES ARE IN OHMS, ±5%, 1/4 WATT.
2. ALL CAPACITANCE VALUES ARE IN UF.



REFERENCE DESIGNATORS	
HIGHEST USED	NOT USED
C13	
CR3	
E16	
J4	
L1	
P2	
PB1	
R19	
S3	
U155	U80,88

For reference only.
Subject to change without notice.

ITEM | QTY | TYPE | ZILOG PART NO | DESCRIPTION | REF DES

PARTS LIST

CONFIDENTIAL

09-0229-00 ZSCAN
NEXT ASSY | USED ON
APPLICATION

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ± 1°
FRACTIONS ± 1/64
3 PLACE DEC ± .010
2 PLC DEC ± .02

CODE IDENT: 56766

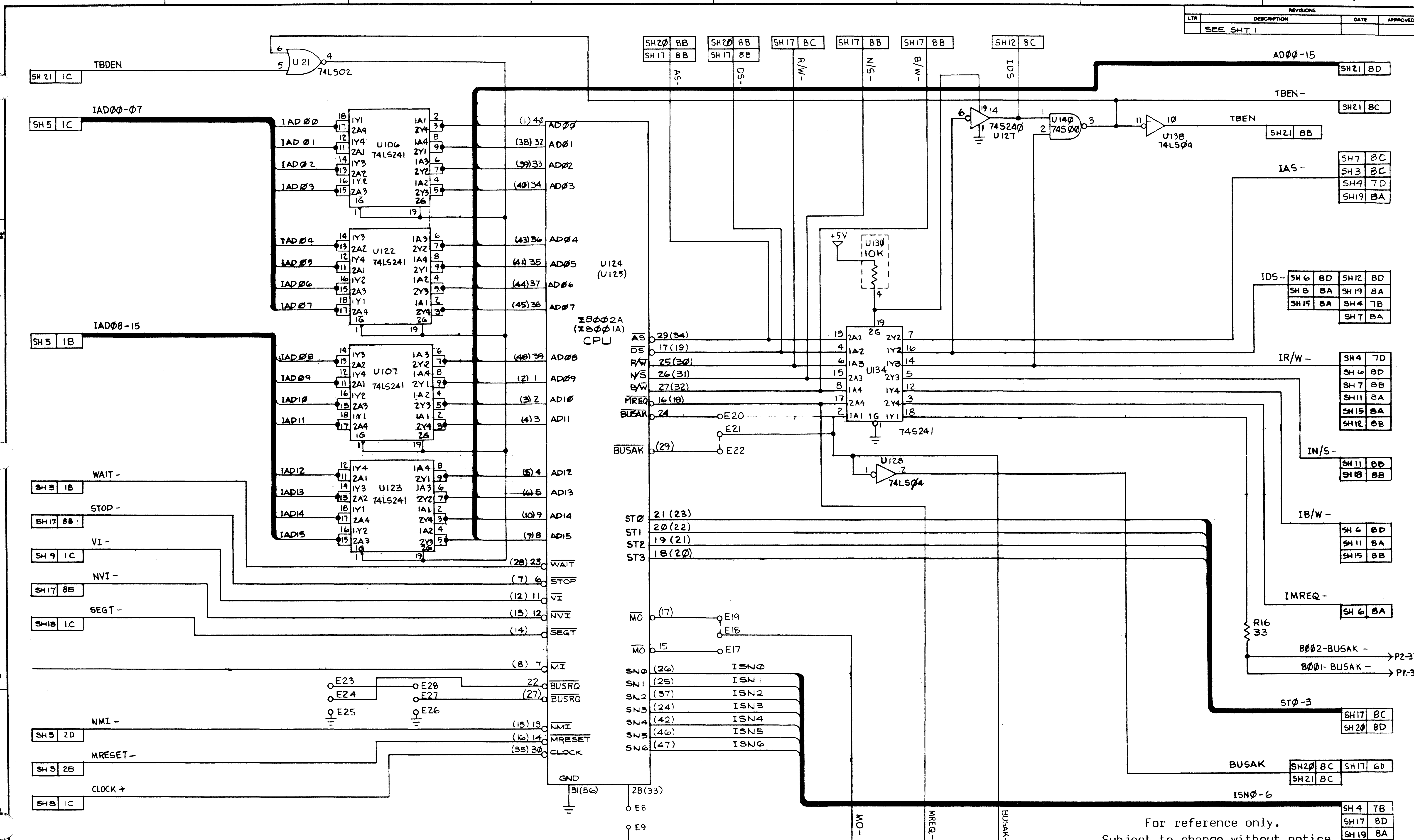
SIGNATURE AND DATE
DRAWN 7-80
DRFTG CHK
APVD

ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA

TITLE LOGIC DIAGRAM ZSCAN 8000

SIZE D | DRAWING NO DZ-0229-00 | ISSUE A

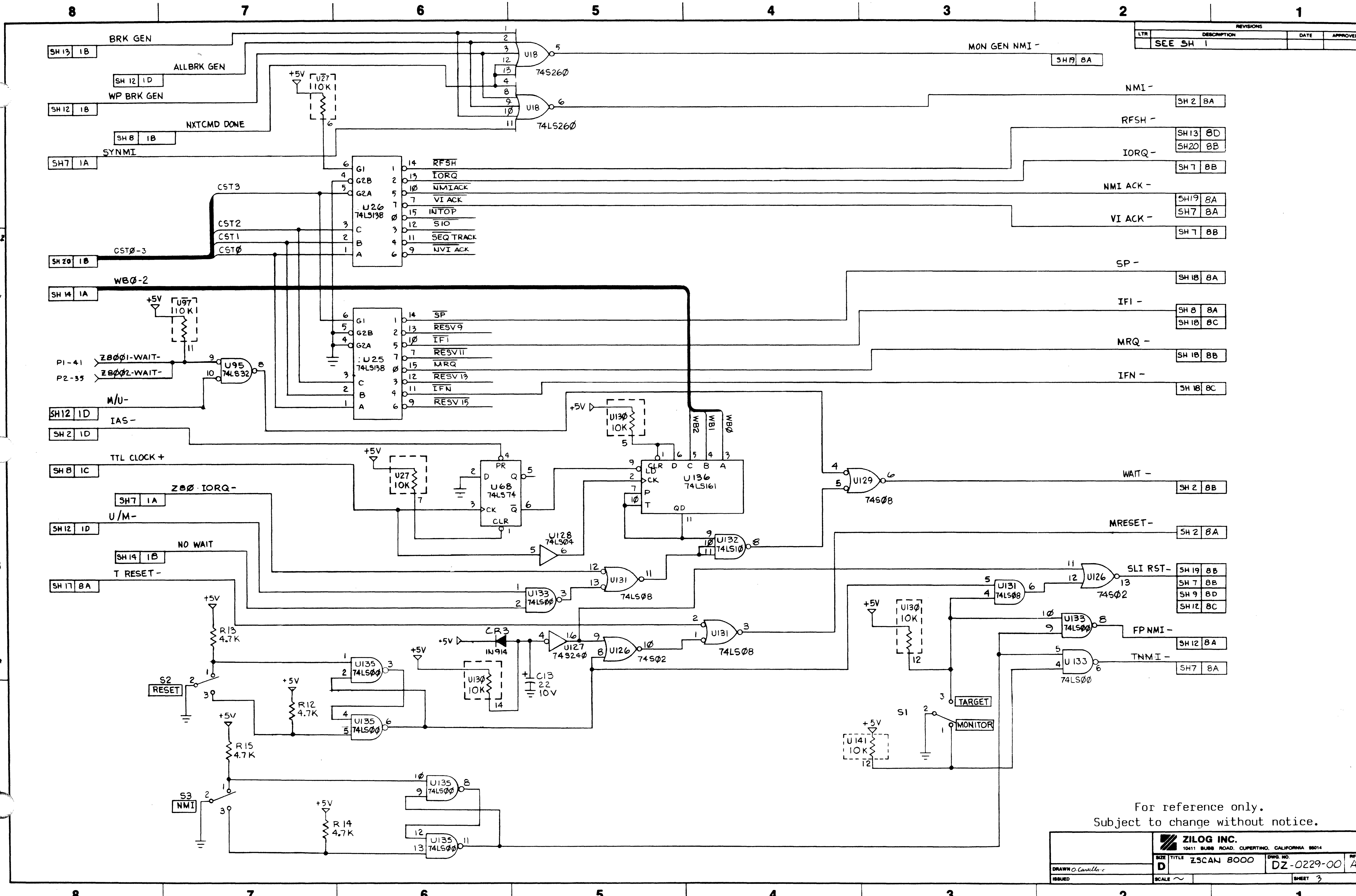
SCALE ~ | SHEET 1 OF 21



For reference only.
Subject to change without notice.

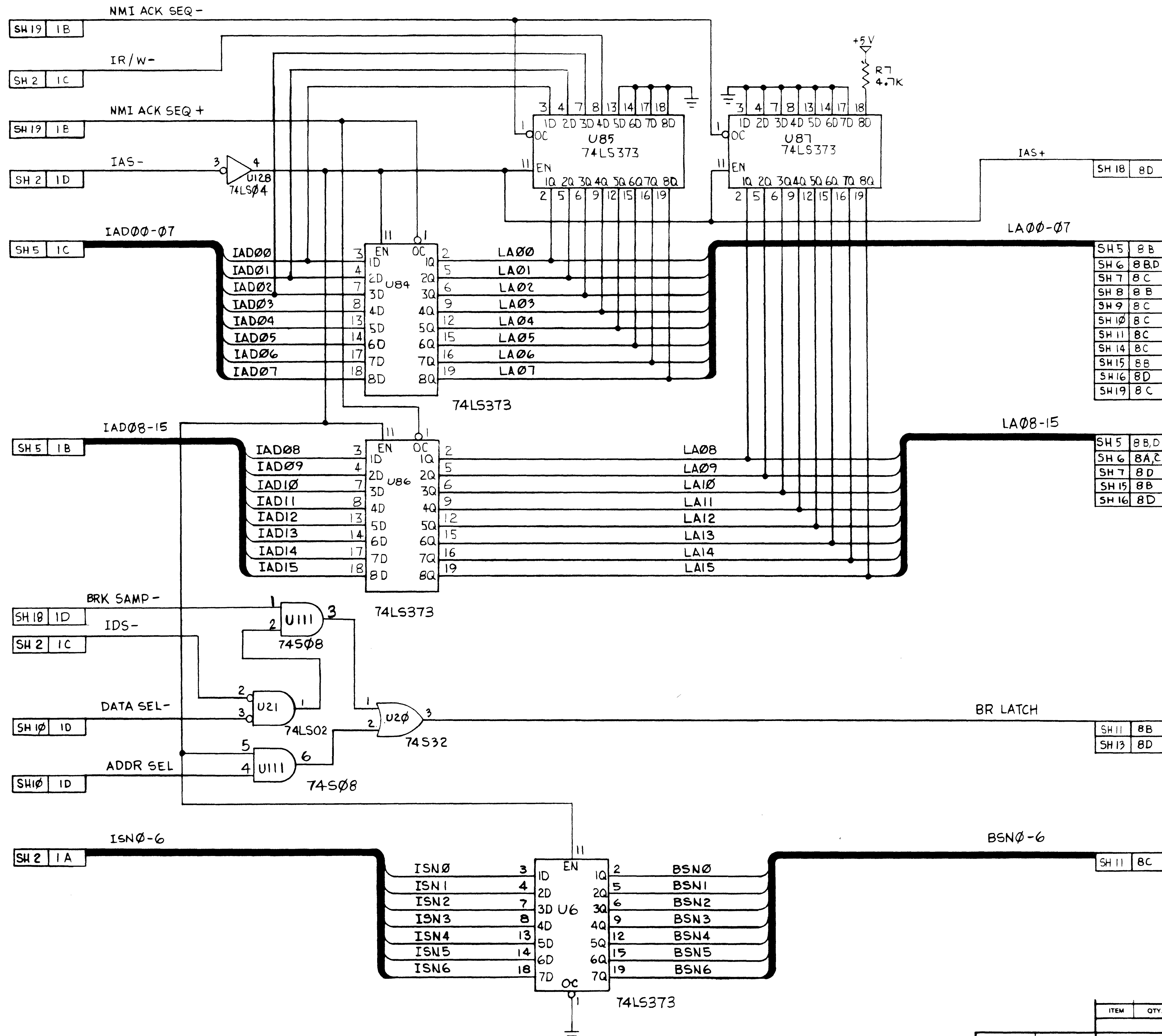
ZILOG INC. 10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

DRAWN	Gemini Graphics 3/80
ISSUED	
SIZE	D
TITLE	ZSCAN 8000
DWG. NO.	DZ-0229-00
REV.	A
SCALE	~
SHEET	2



For reference only.
Subject to change without notice.

LTR	DESCRIPTION	DATE	APPROVED
	SEE SH 1		



For reference only.
Subject to change without notice.

ITEM	QTY	TYPE	ZILOG PART NO.	DESCRIPTION	REF. DES.

PARTS LIST

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ± 1°
FRACTIONS ± 1/64
3 PLACE DEC ± .010
2 PLC DEC ± .02

CODE IDENT: 56708

MAT'L

FINISH

	SIGNATURE AND DATE
DRWN	James Hogues 3/80
DRFTG CHK	
APVD	

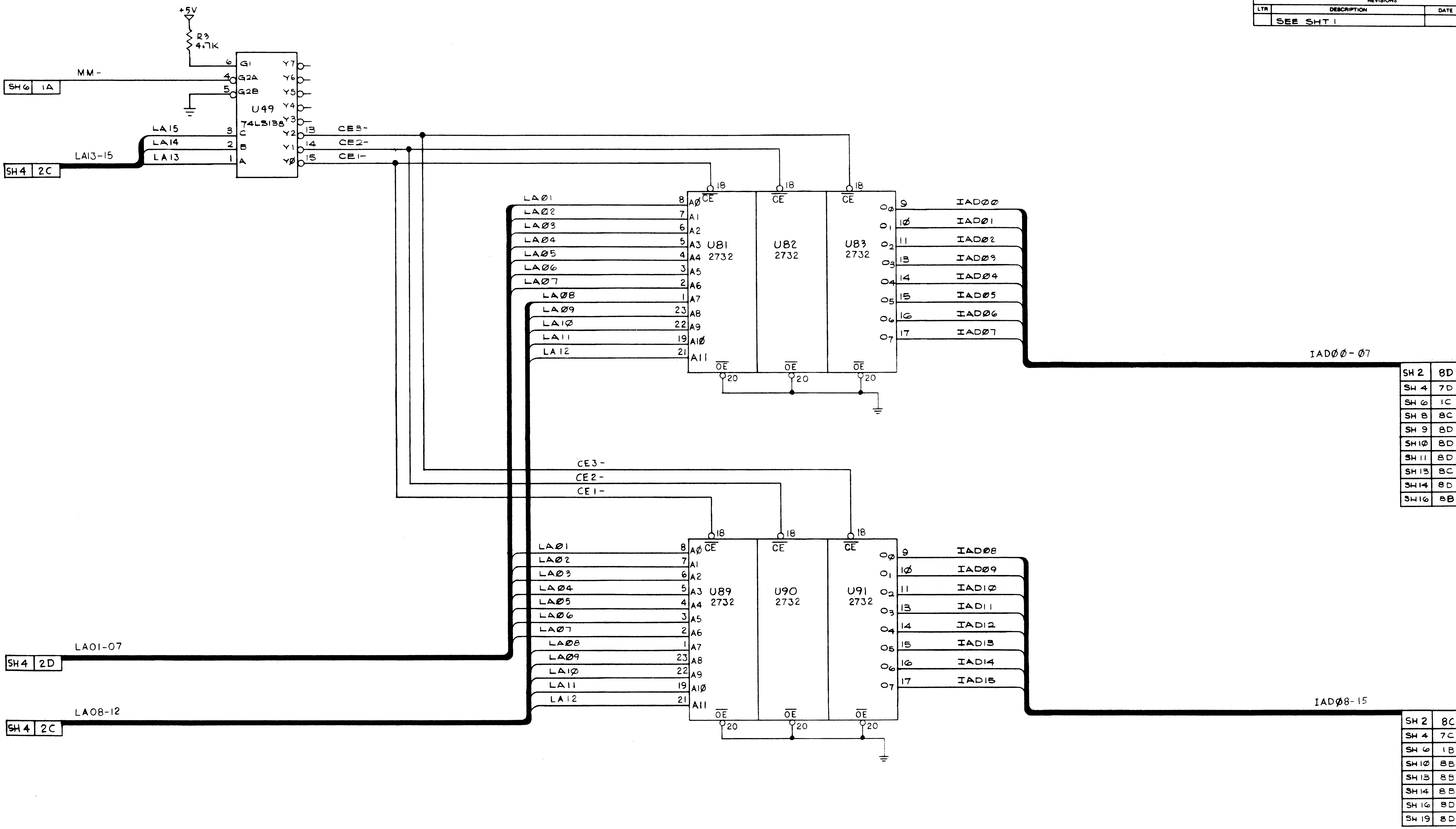
NEXT ASSY | USED ON
APPLICATION

ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

TITLE: Z SCAN 8000

SIZE: D | DRAWING NO: DZ-0229-00 | ISSUE: A

SCALE: ~ | SHEET 4 OF



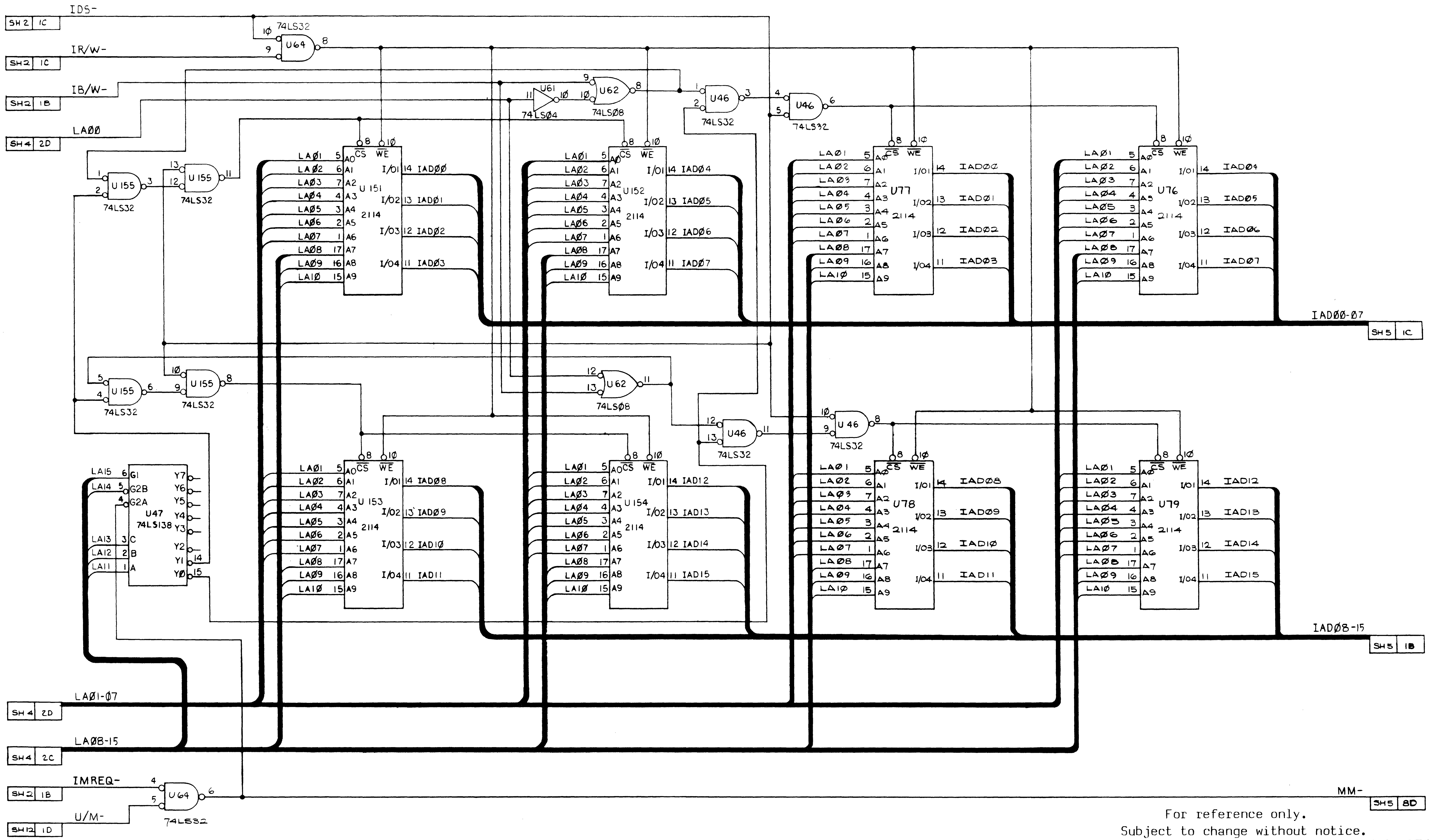
REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
	SEE SHT 1		

For reference only.
Subject to change without notice.

ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

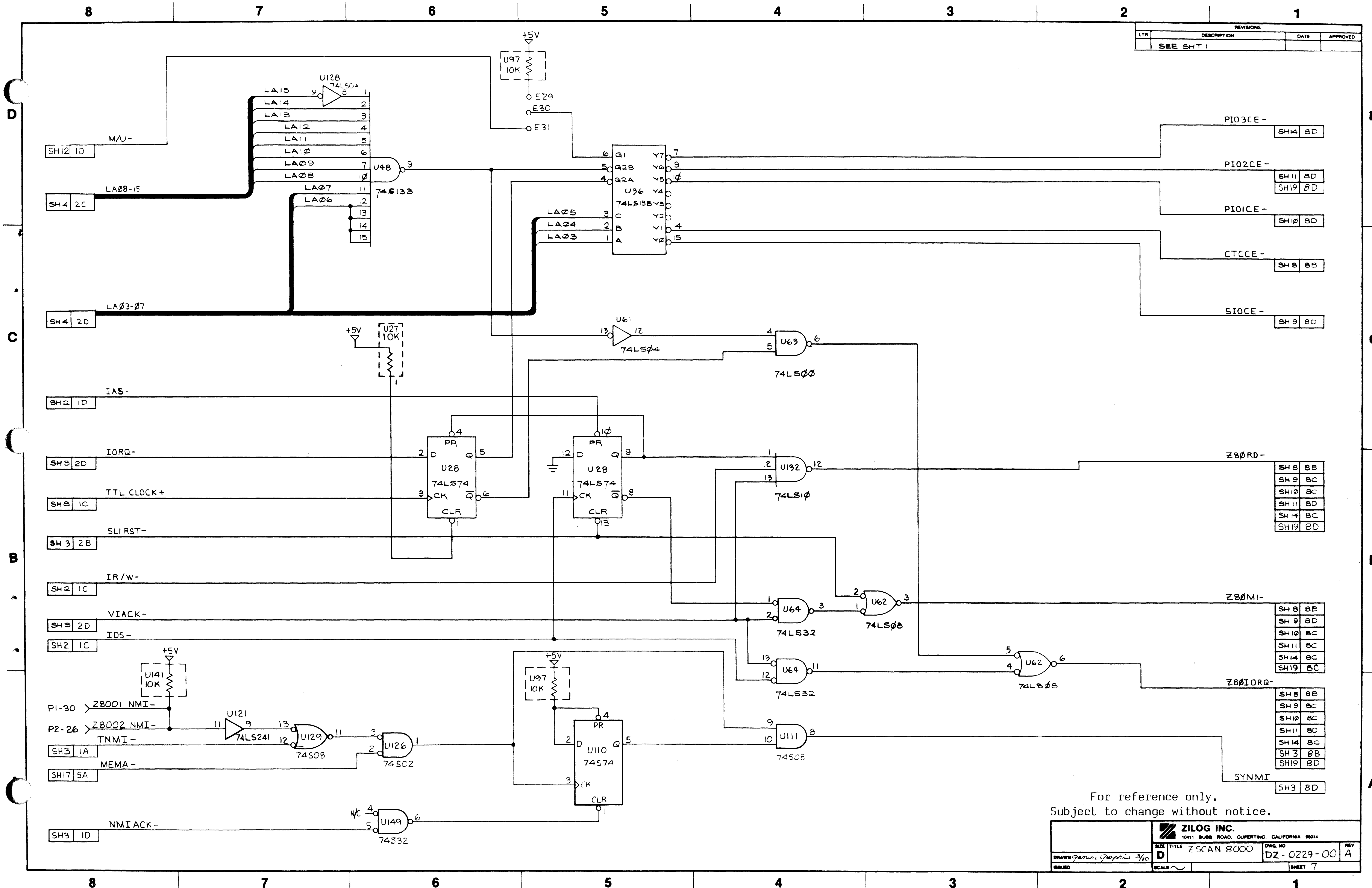
SIZE	TITLE	DWG. NO.	REV
D	ZSCAN 8000	DZ-0229-00	A

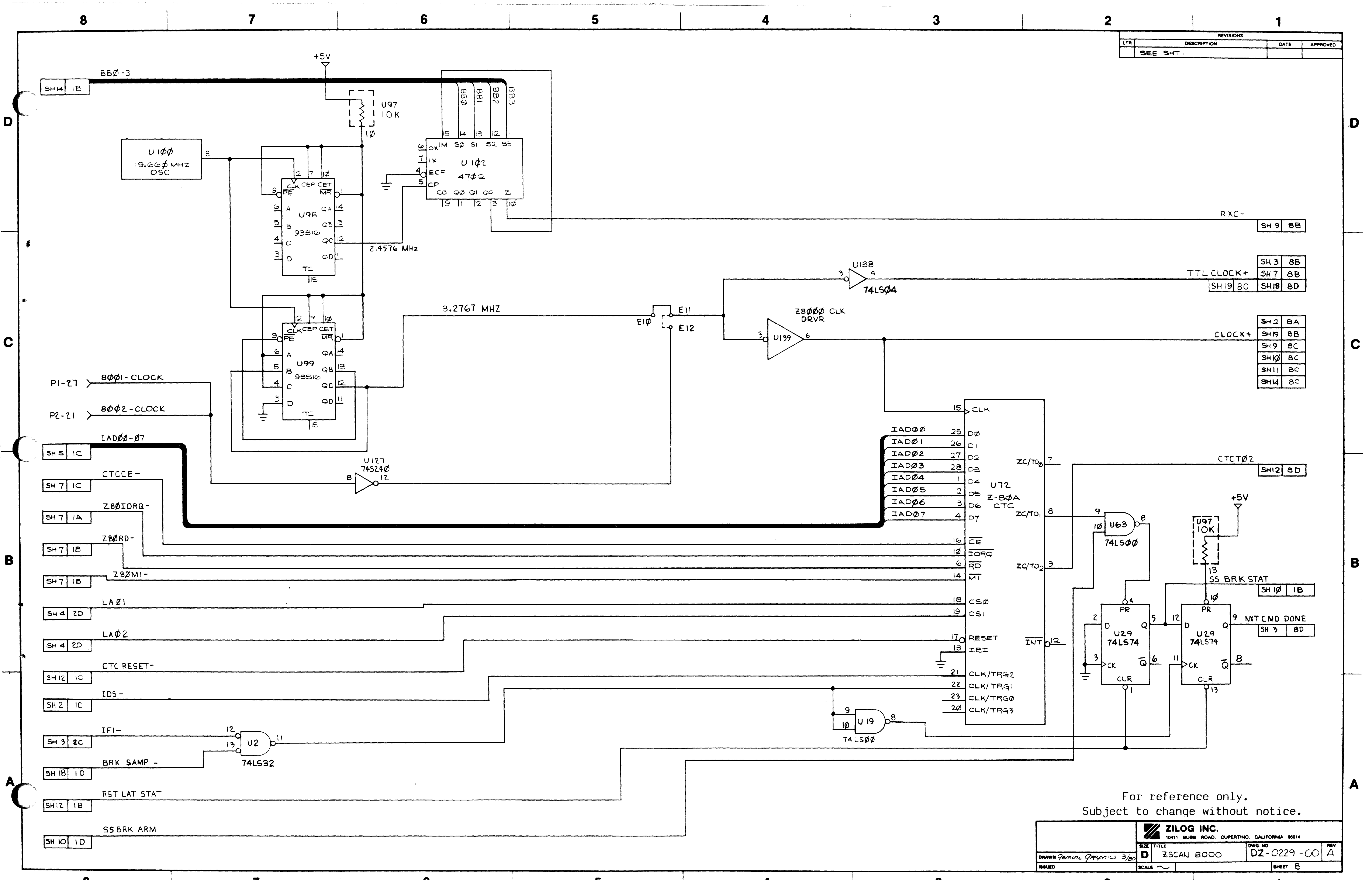
DRAWN
ISSUED
SCALE ~
SHEET 5



For reference only.
Subject to change without notice.

REVISIONS	SEE SHT 1
ZILOG INC.	10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014
SIZE	D
TITLE	ZSCAN 8000
DWG. NO.	DZ-0229-00
REV.	A
DRAWN	Gemini Graphics 3/80
SHEET	6


REVISIONS
LTR
DESCRIPTION
DATE
APPROVED
SEE SHT 1
+5V
U97
10K
E29
E30
E31
U128
74LS04
LA15
LA14
LA13
LA12
LA11
LA1Ø
LAØ9
LAØ8
LAØ7
LAØ6
U48
74S133
M/U-
SH12 1D
LAØ8-15
SH4 2C
LAØ3-Ø7
SH4 2D
G1
G2B
G2A
U36
74LS138
LAØ5
LAØ4
LAØ3
PIO3CE-
SH14 8D
PIO2CE-
SH11 8D
SH19 8D
PIO1CE-
SH1Ø 8D
CTCCE-
SH8 8B
SIOCE-
SH9 8D
U61
74LSØ4
U63
74LSØØ
U27
10K
IAS-
SH2 1D
IORQ-
SH3 2D
TTL CLOCK+
SH8 1C
SL1RST-
SH3 2B
IR/W-
SH2 1C
VIACK-
SH3 2D
IDS-
SH2 1C
U28
74LS74
PR
CLR
CK
U132
74LS1Ø
Z8ØRD-
SH8 8B
SH9 8C
SH1Ø 8C
SH11 8D
SH14 8C
SH19 8D
U64
74LS32
U62
74LSØ8
Z8ØMI-
SH8 8B
SH9 8D
SH1Ø 8C
SH11 8C
SH14 8C
SH19 8C
Z8ØIORQ-
SH8 8B
SH9 8C
SH1Ø 8C
SH11 8D
SH14 8C
SH3 8B
SH19 8D
U141
10K
P1-30
Z8001 NMI-
P2-26
Z8002 NMI-
U121
74LS241
U129
74S08
TNMI-
SH3 1A
U126
74S02
MEMA-
SH17 5A
U110
74S74
U111
74S08
SYNMI
SH3 8D
N/C
U149
74S32
NMIACK-
SH3 1D
For reference only.
Subject to change without notice.
ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014
SIZE D
TITLE ZSCAN 8000
DWG. NO DZ-0229-00
REV. A
DRAWN Gemini Graphics 3/80
ISSUED
SCALE
SHEET 7

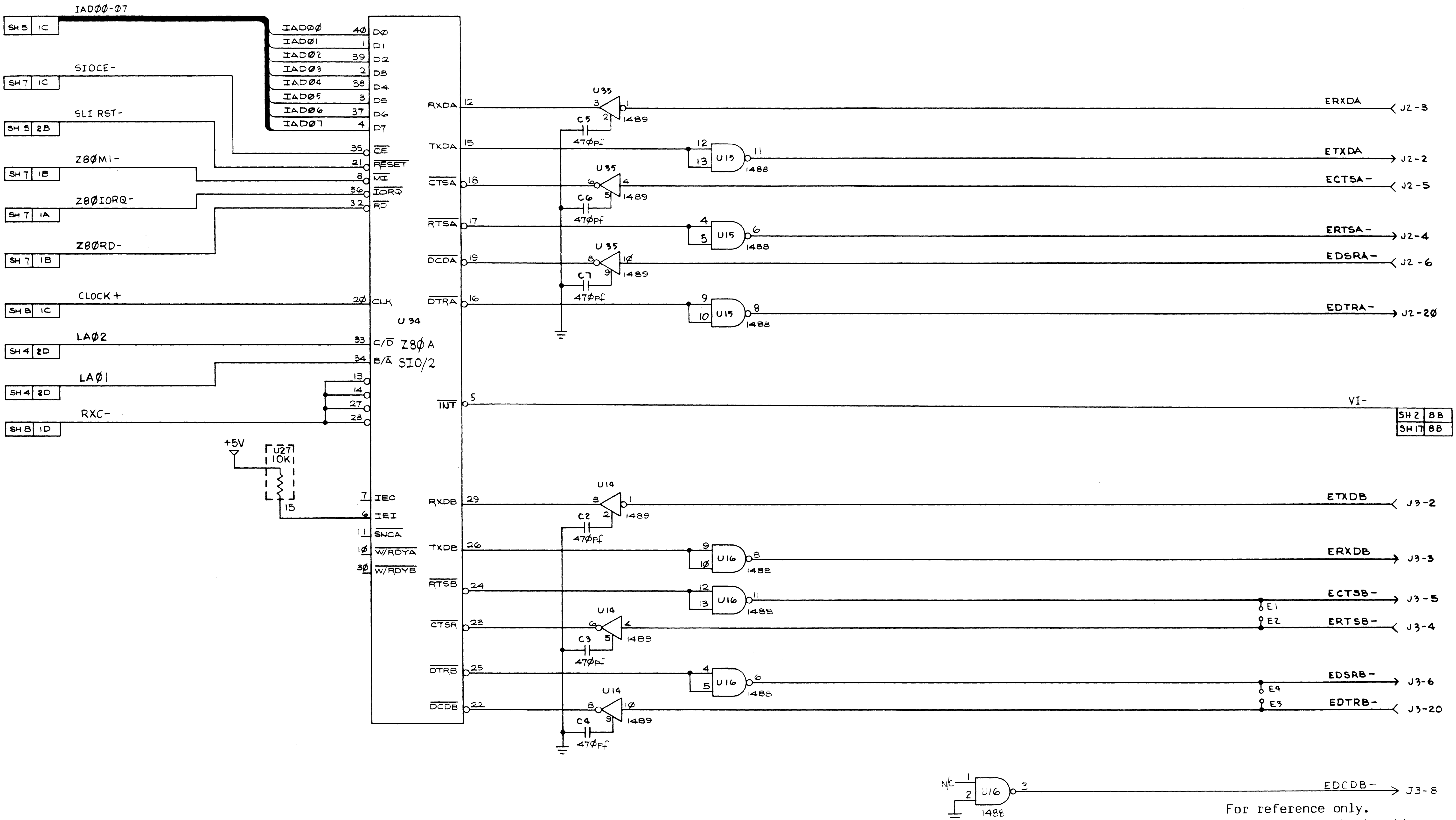


For reference only.
Subject to change without notice.

ZILOG INC. 10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

DRAWN	Gemini Graphics 3/80
SIZE	D
TITLE	ZSCAN 8000
DWG. NO.	DZ-0229-00
REV.	A
SHEET	8

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
	SEE SHT 1		

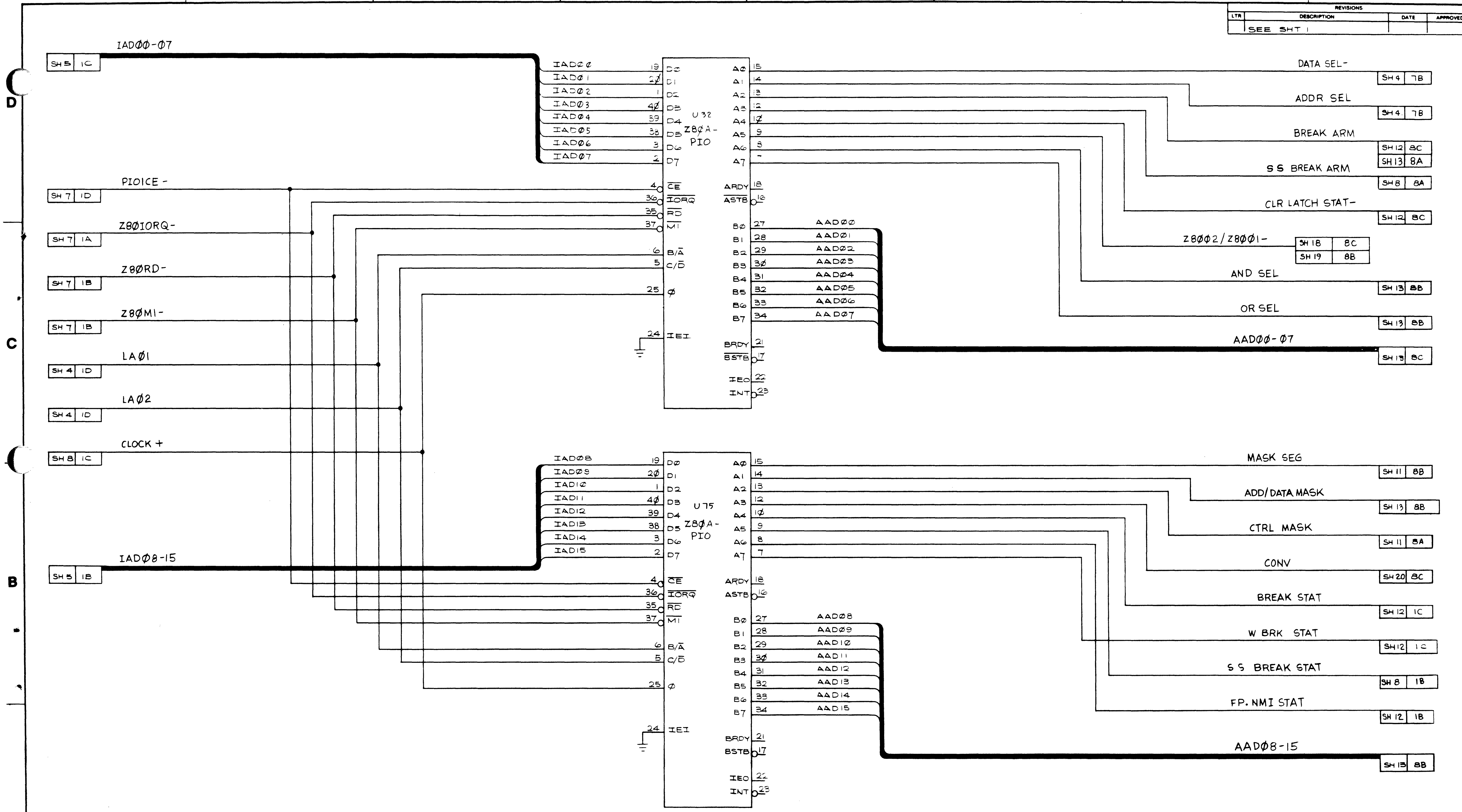


For reference only.
Subject to change without notice.

ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

SIZE	TITLE	DWG. NO.	REV.
D	ZSCAN 8000	DZ-0229-00	A

DRAWN Gemini Graphics 3/80
ISSUED
SCALE ~
SHEET 9

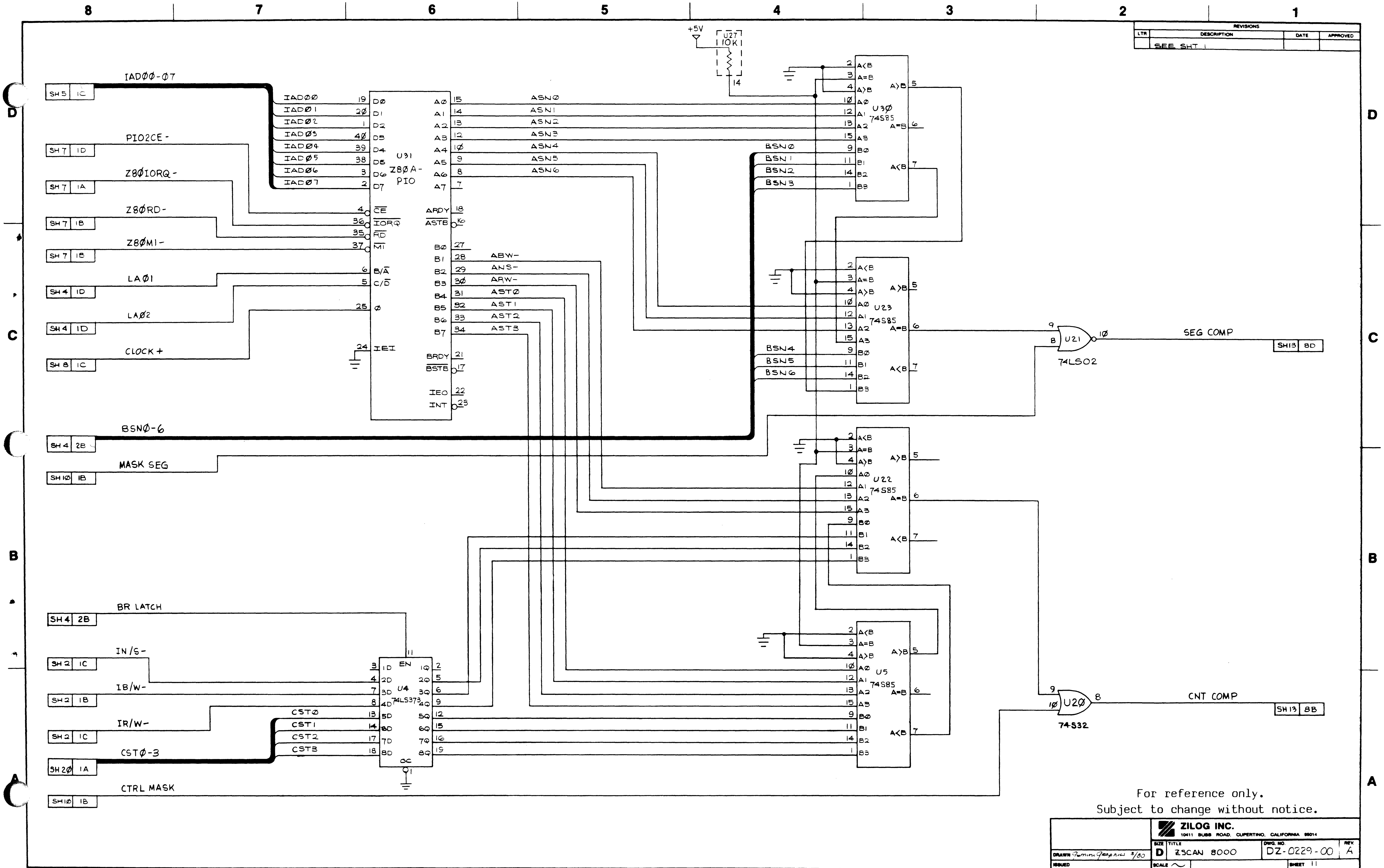


For reference only.
Subject to change without notice.

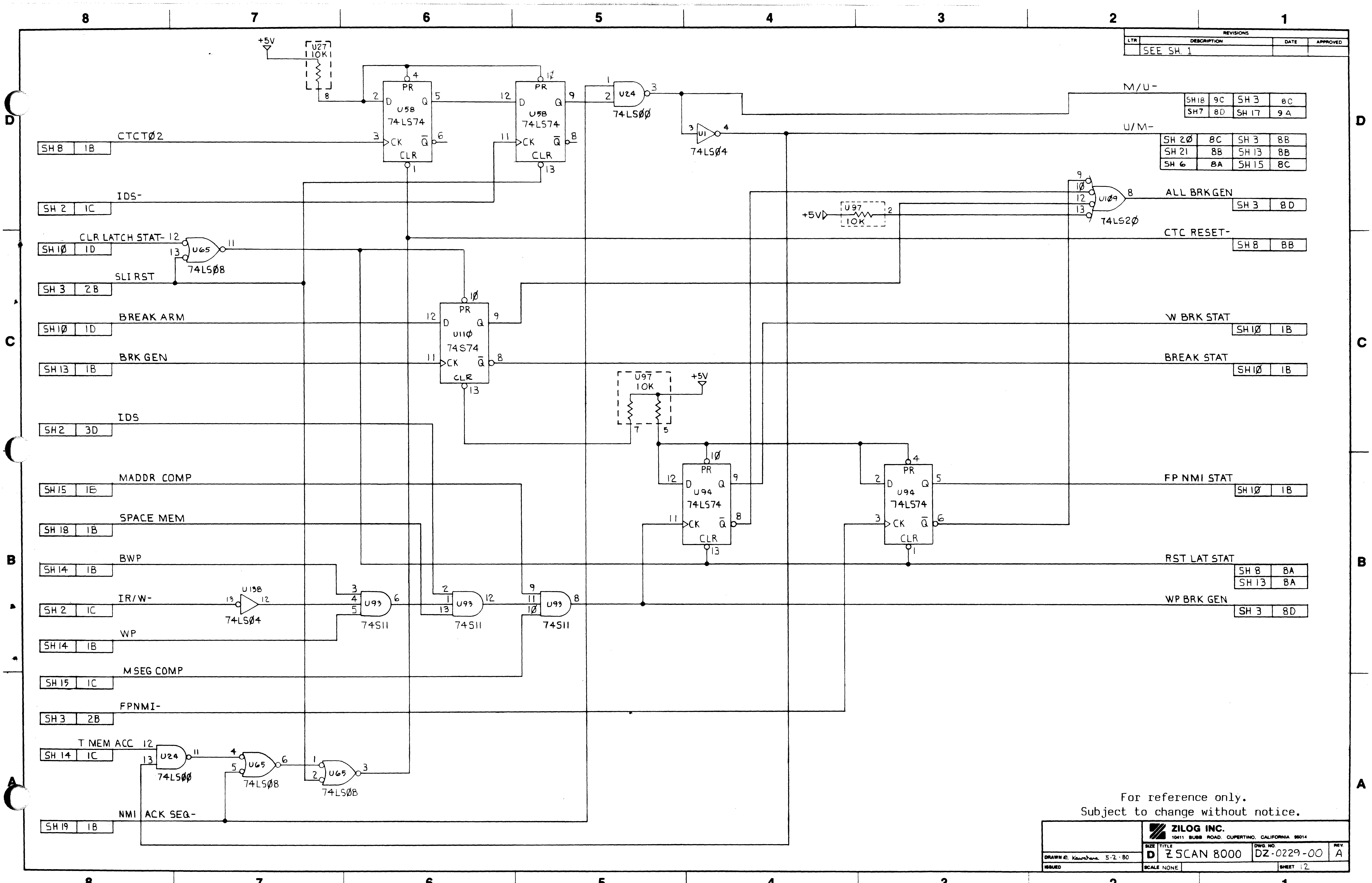
ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

SIZE	TITLE	DWG. NO.	REV.
D	ZSCAN 8000	DZ-0229-00	A

DRAWN Gemini Graphics 3/80
ISSUED
SCALE ~
SHEET 10



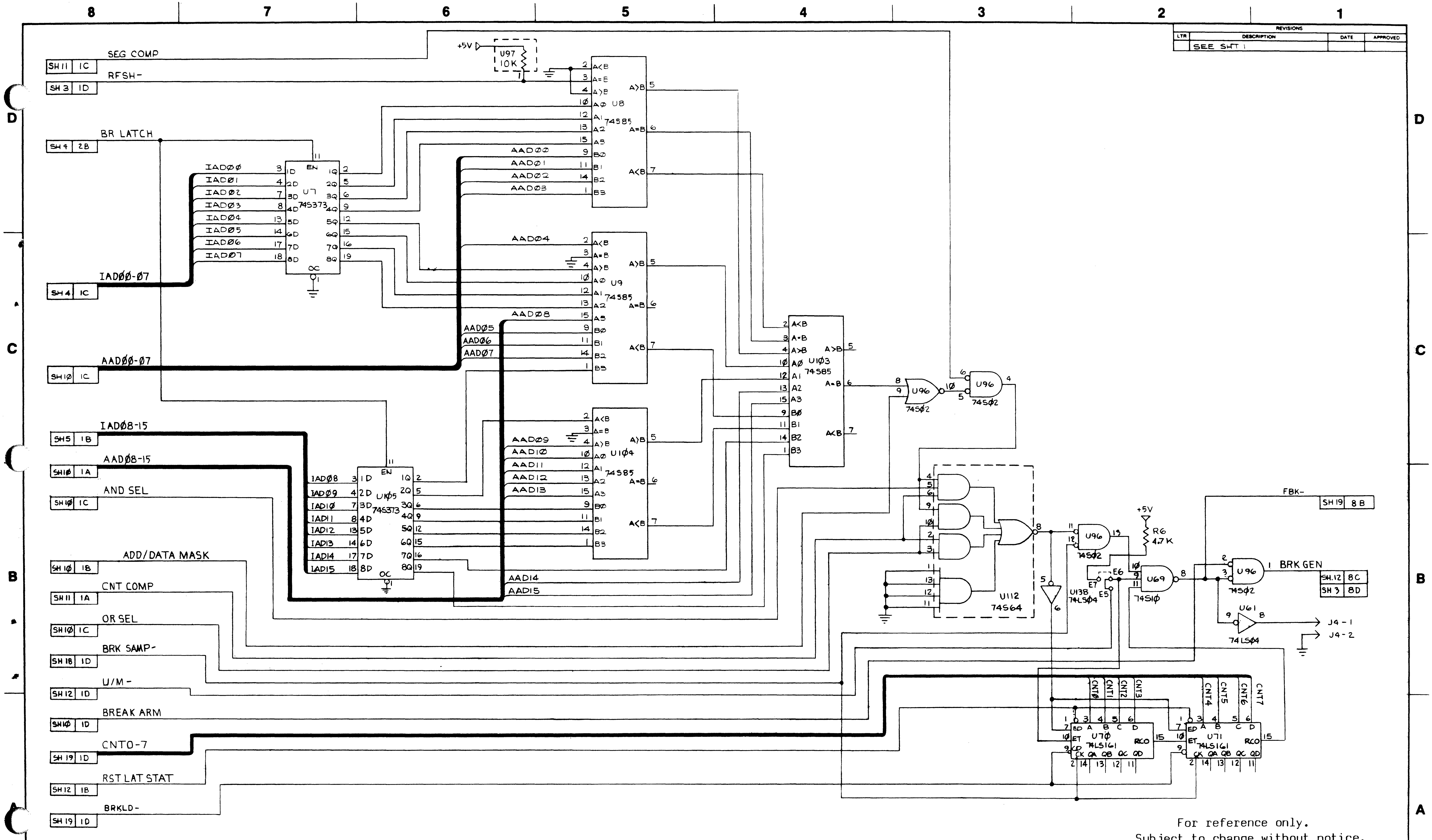
For reference only.
Subject to change without notice.



For reference only.
Subject to change without notice.

ZILOG INC. 10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

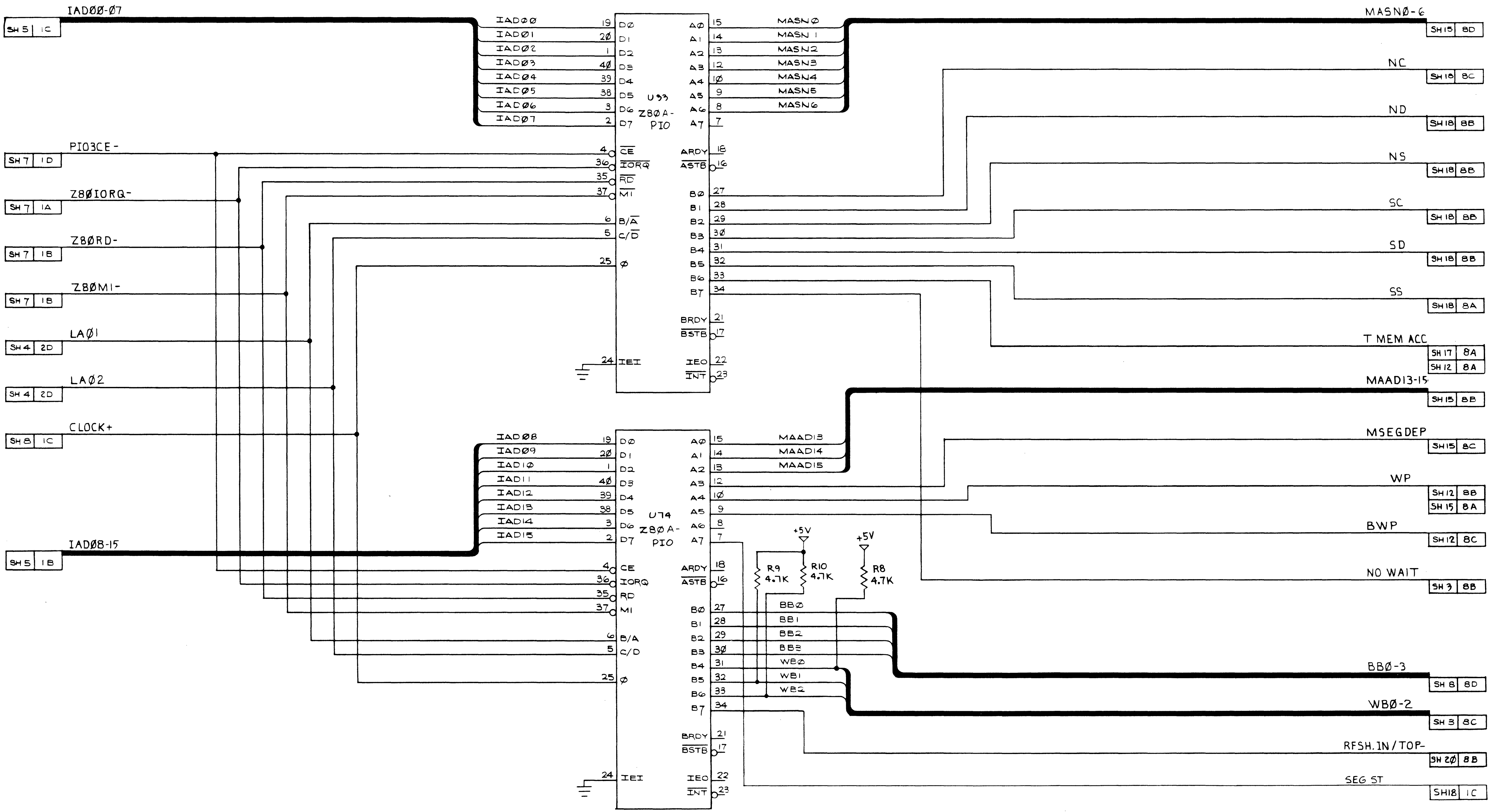
SIZE	D
TITLE	ZSCAN 8000
DWG. NO.	DZ-0229-00
REV.	A
DRAWN	R. Kawahara 5-2-80
SCALE	NONE
SHEET	12



For reference only.
Subject to change without notice.

ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

SIZE D	TITLE Z SAN 8000	DWG. NO. DZ-0229 -00	REV. A
DRAWN Gemini Graphics 3/80	ISSUED	SCALE ~	SHEET 13



REVISIONS

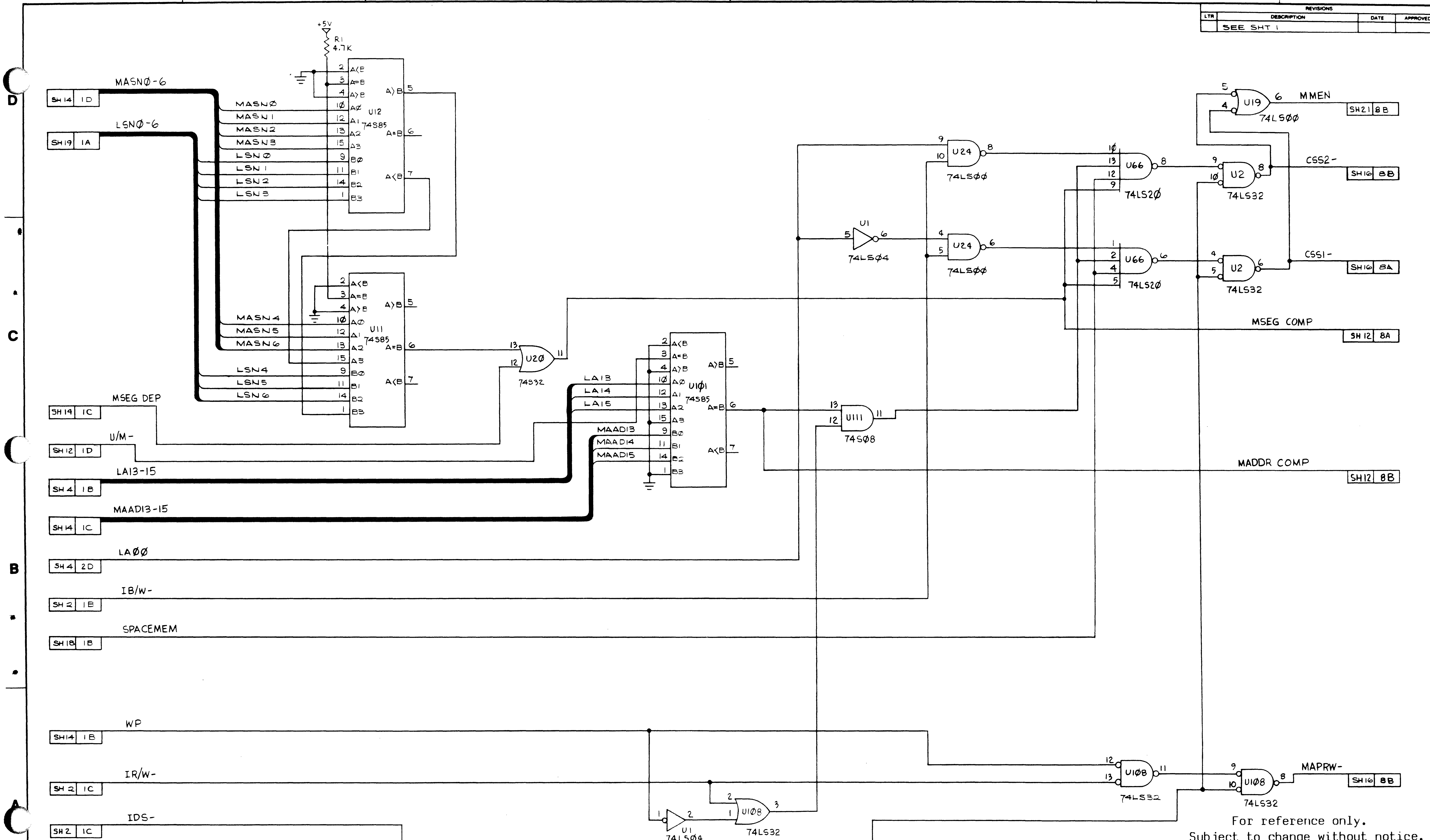
LTR	DESCRIPTION	DATE	APPROVED
	SEE SHT 1		

For reference only.
Subject to change without notice.

ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

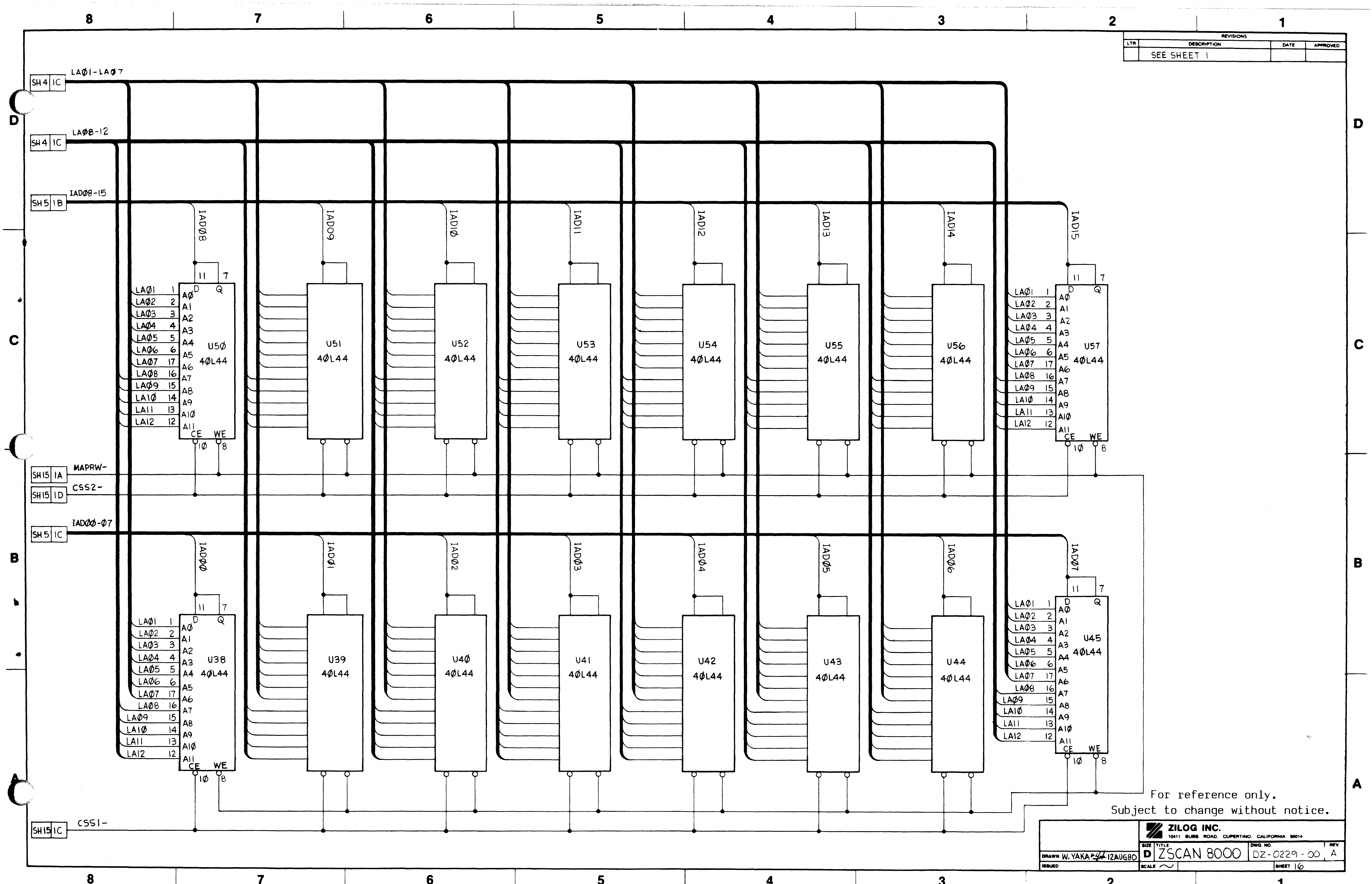
SIZE	TITLE	DWG. NO.	REV
D	ZSCAN 8000	DZ-0229-00	A

DRAWN Gemini Jaegar 3/80
ISSUED
SCALE ~
SHEET 14



For reference only.
Subject to change without notice.

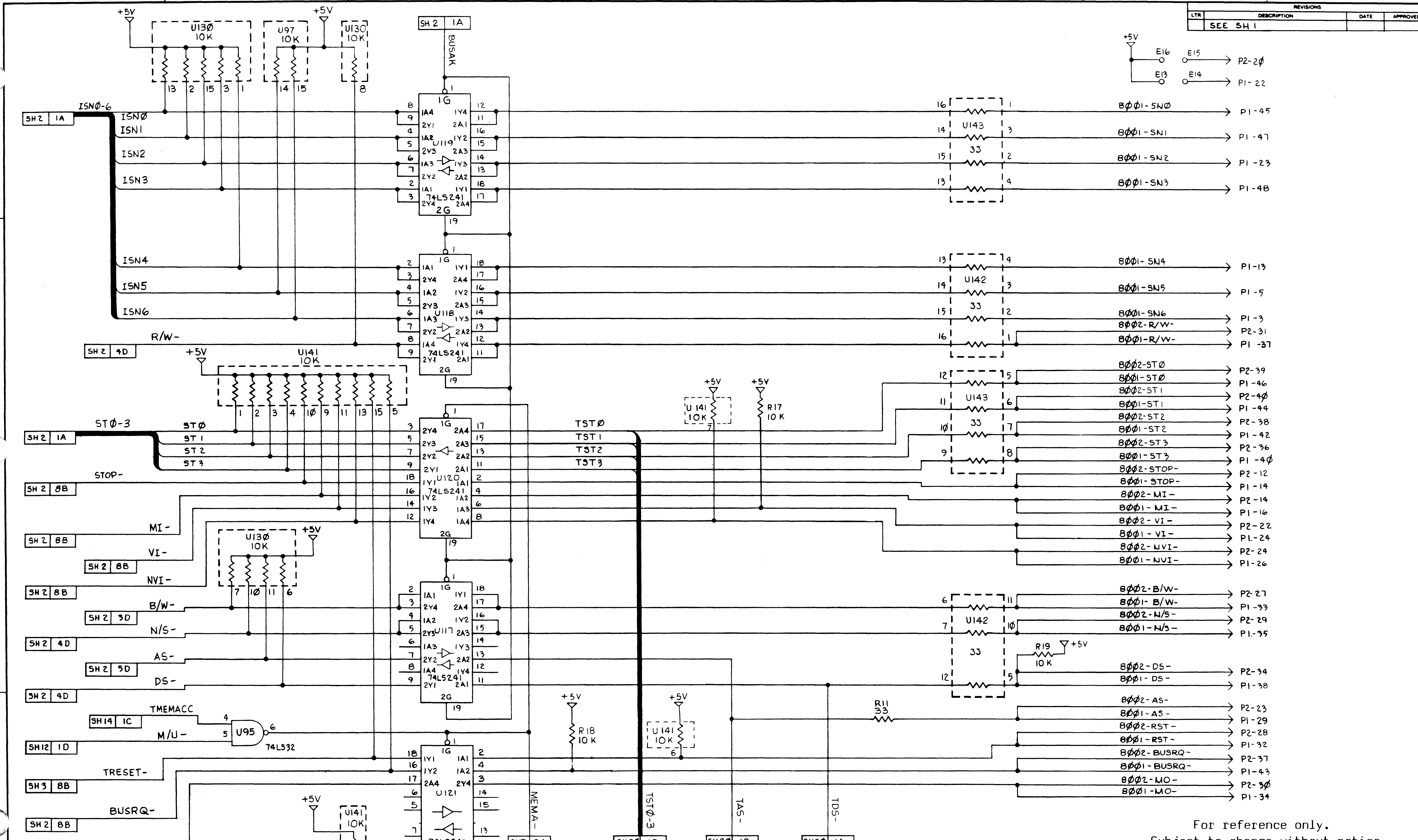
ZILOG INC. 10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014	
SIZE D	TITLE ZSCAN 8000
DWG. NO. DZ-0229-00	REV. A
SHEET 15	



For reference only.
Subject to change without notice.

ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

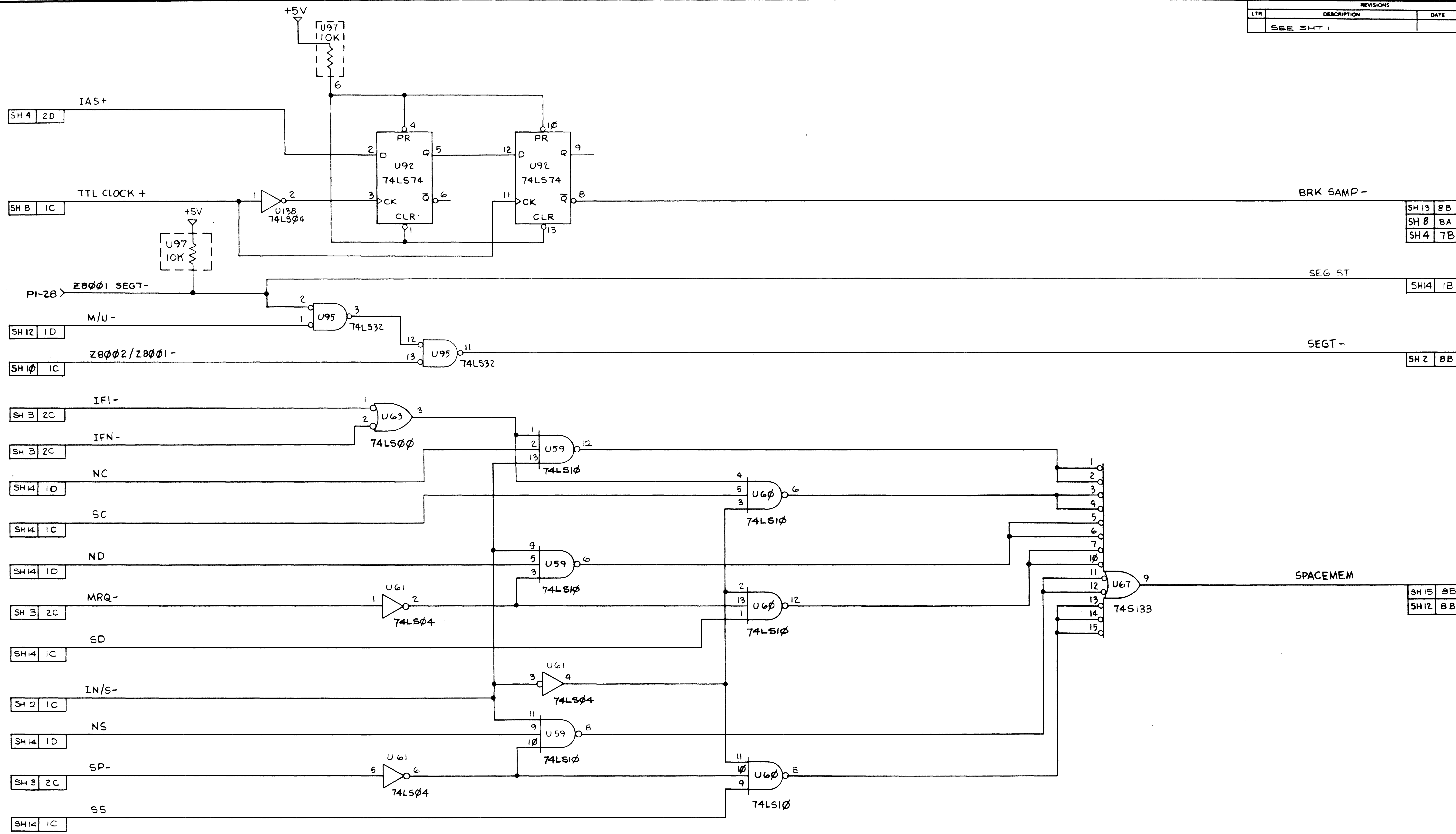
DRAWN W. YAKA 12AUG80
ISSUED
SIZE D
TITLE ZSCAN 8000
DWG. NO. DZ-0229-00
REV. A
SCALE ~
SHEET 16



For reference only.
Subject to change without notice.

ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

SIZE	D
TITLE	ZSCAN 8000
DWG. NO.	DZ-0229-00
REV.	A
DRAWN	O. Cavallaro 6-3-80
ISSUED	
SCALE	~
SHEET	17



REVISIONS

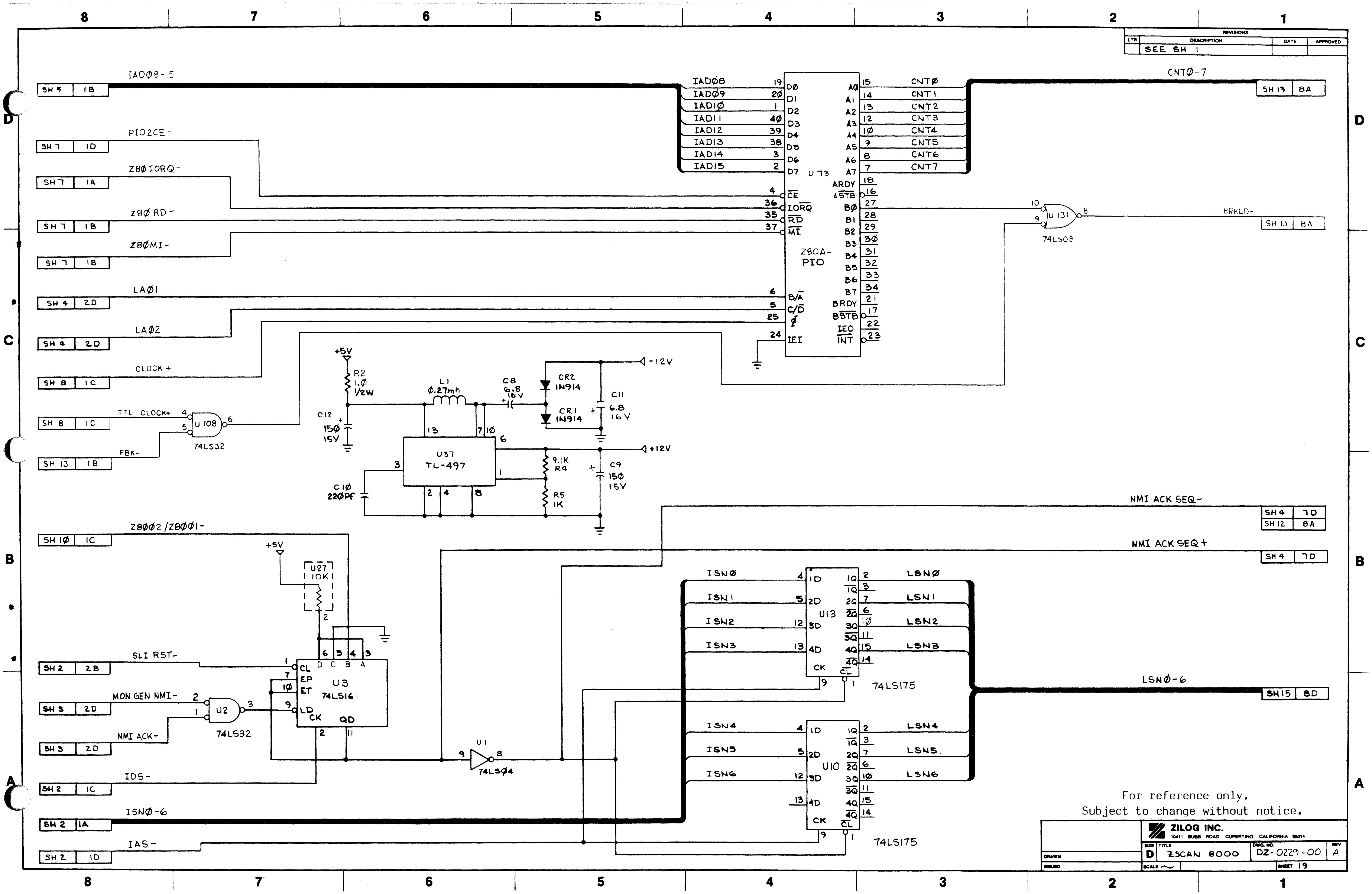
LTR	DESCRIPTION	DATE	APPROVED
	SEE SHT 1		

For reference only.
Subject to change without notice.

ZILOG INC. 10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

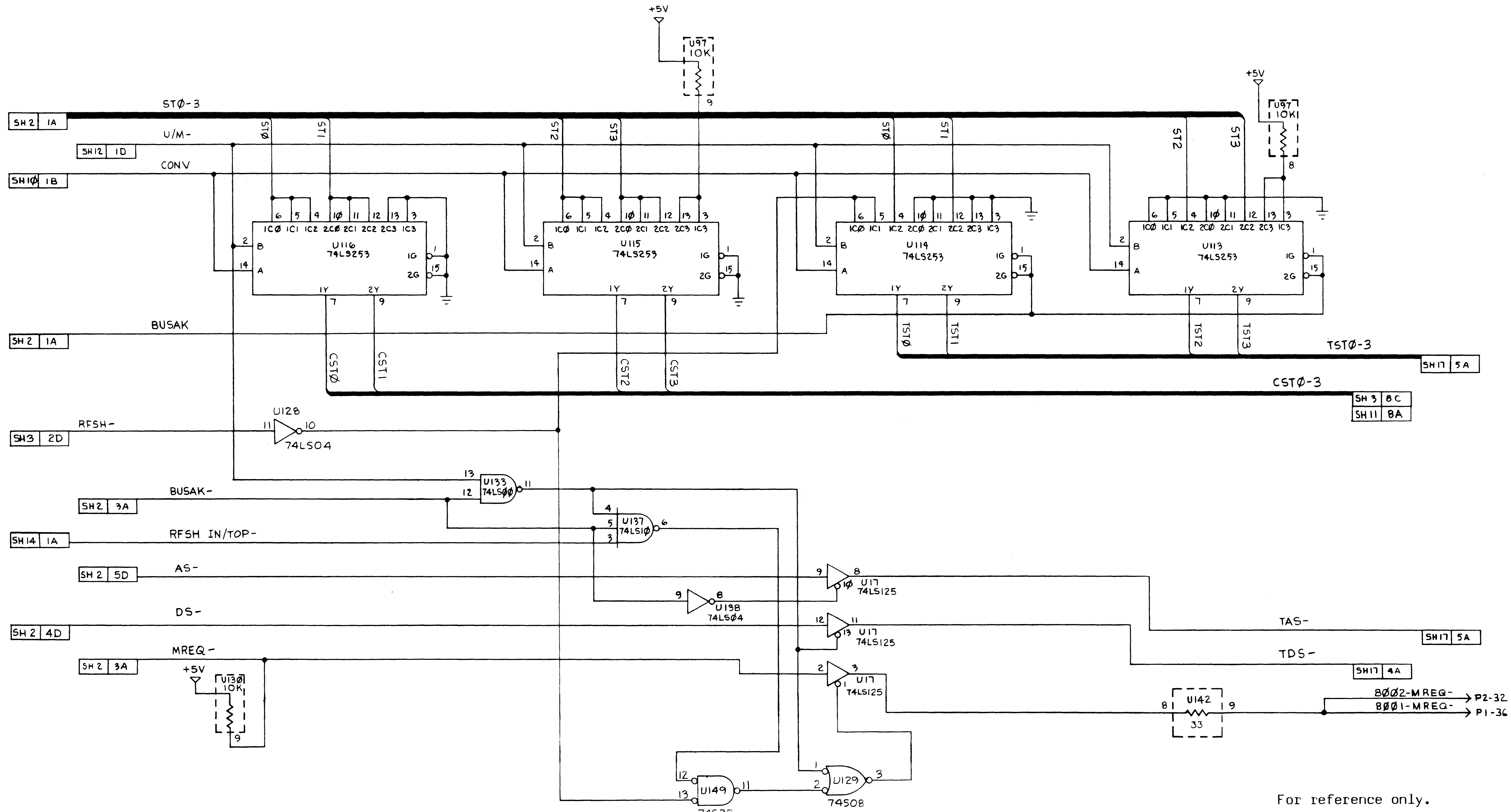
SIZE D — TITLE: ZSCAN 8000 — DWG. NO. DZ-0229-00 — REV A

DRAWN 3/80 — SCALE ~ — SHEET 18



REVISIONS
LTR DESCRIPTION DATE APPROVED
SEE SH 1
IADØ8-15
SH 5 1B
IADØ8 IADØ9 IAD1Ø IAD11 IAD12 IAD13 IAD14 IAD15
DØ D1 D2 D3 D4 D5 D6 D7
U73
Z80A-PIO
AØ A1 A2 A3 A4 A5 A6 A7
ARDY ASTB BØ B1 B2 B3 B4 B5 B6 B7 BRDY BSTB IEO INT
CE IORQ RD MI B/A C/D Ø IEI
CNTØ CNT1 CNT2 CNT3 CNT4 CNT5 CNT6 CNT7
CNTØ-7
SH 13 8A
PIO2CE-
SH 7 1D
Z8Ø IORQ-
SH 7 1A
Z8Ø RD-
SH 7 1B
Z8ØMI-
SH 7 1B
LAØ1
SH 4 2D
LAØ2
SH 4 2D
CLOCK +
SH 8 1C
TTL CLOCK+
SH 8 1C
U108
74LS32
FBK-
SH 13 1B
U131
74LS08
BRKLD-
SH 13 8A
+5V
R2 1.Ø 1/2W
C12 15Ø 15V
L1 Ø.27mh
C8 6.8 16V
CR2 1N914
CR1 1N914
C11 6.8 16V
-12V
+12V
U37 TL-497
9.1K R4
R5 1K
C9 15Ø 15V
C1Ø 22ØPf
NMI ACK SEQ-
SH 4 7D
SH 12 8A
NMI ACK SEQ+
SH 4 7D
Z8ØØ2/Z8ØØ1-
SH 1Ø 1C
+5V
U27 1ØK
SLI RST-
SH 2 2B
U3 74LS161
CL EP ET LD CK D C B A QD
MON GEN NMI-
SH 3 2D
U2
74LS32
NMI ACK-
SH 3 2D
U1
74LSØ4
IDS-
SH 2 1C
ISNØ-6
SH 2 1A
IAS-
SH 2 1D
ISNØ ISN1 ISN2 ISN3
U13
74LS175
1D 2D 3D 4D CK 1Q 2Q 3Q 4Q CL
LSNØ LSN1 LSN2 LSN3
ISN4 ISN5 ISN6
U1Ø
74LS175
LSN4 LSN5 LSN6
LSNØ-6
SH15 8D
For reference only.
Subject to change without notice.
ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014
SIZE D
TITLE ZSCAN 8000
DWG. NO. DZ-0229-00
REV. A
DRAWN
ISSUED
SCALE ~
SHEET 19

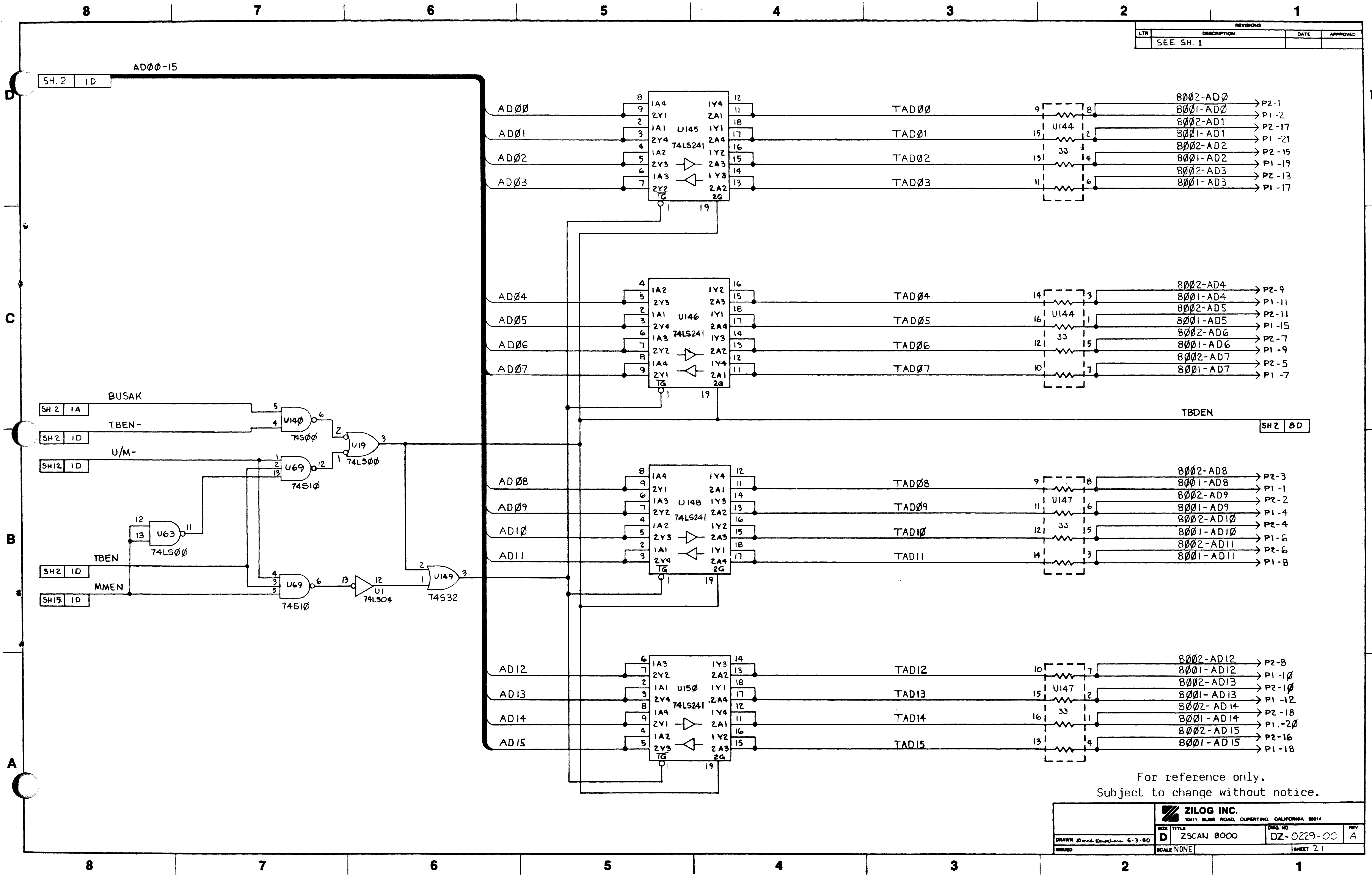
LTR	DESCRIPTION	DATE	APPROVED
	SEE SHT 1		



For reference only.
Subject to change without notice.

ZILOG INC. 10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

SIZE	D
TITLE	ZSCAN 8000
DWG. NO.	DZ-0229 -0C
REV.	A
DRAWN	O. Cavallino 6-4-80
ISSUED	
SCALE	~
SHEET	20



For reference only.
Subject to change without notice.

ZILOG INC.
10411 BUBB ROAD, CUPERTINO, CALIFORNIA 95014

SIZE D	TITLE ZSCAN 8000	DWG. NO. DZ-0229-00	REV. A
SCALE NONE			SHEET 21

DRAWN David Kawahara 6-3-80
ISSUED

INDEX

Z-SCAN INDEX

-A-

-B-

-C-

-D-

-E-

-F-

-J-

-K-

-L-

-M-

-N-

-O-

-P-

-Q-

-R-

-S-

-T-

-U-

-V-

-W-

-Z-

Zilog Sales Office

West

Sales & Technical Center
Zilog, Incorporated
1333 Lawrence Expressway
Suite 400
Santa Clara, CA 95051
Tele: (408) 446-9848

Sales & Technical Center
Zilog, Incorporated
18023 Sky Park Circle
Suite J
Irvine, CA 92714
Tele: (714) 549-2891
TWX: 910 595-2803

Sales & Technical Center
Zilog, Incorporated
15643 Sherman Way
Suite 430
Van Nuys, CA 91406
Tele: (213) 989-7484
TWX: 910-495-1765

Midwest

Sales & Technical Center
Zilog, Incorporated
890 East Higgins Road
Suite 147
Schaumburg, IL 60195
Tele: (312) 885-8080
TWX: 910 291 1064

South

Sales & Technical Center
Zilog, Incorporated
2711 Valley View, Suite 103
Dallas, TX 75234
Tele: (214) 243-6550
TWX: 910 860 5850

Technical Center
Zilog, Incorporated
1442 U.S. Hwy 19 South
Suite 135
Clearwater, FL 33516
Tele: (813) 535-5571

East

Sales & Technical Center
Zilog, Incorporated
Corporate Place
99 South Bedford St.
Burlington, MA 01803
Tele: (617) 273-4222
TWX: 710 332-1726

Sales & Technical Center
Zilog, Incorporated
110 Gibraltar Road
Horsham, PA 19044
Tele: (215) 441-8282
TWX: 510 665 7077

United Kingdom

Zilog (U.K.) Limited
Babbage House, King Street
Maidenhead SL6 1DU
Berkshire, United Kingdom
Tele: (628) 36131
TELEX: 848609

West Germany

Zilog GmbH
Zugspitzstrasse 2a
D-8011 Vaterstetten
Munich, West Germany
Tele: 08106 4035
TELEX: 529110 Zilog d.

Japan

Zilog, Japan KK
Linden Sky Heights
Bldg. 1F
13-2 Sakuragaoka-Machi
Shibuya-Ku Tokyo 105
Japan
Tele: (3) 476-3010
TWX: 781 23723 Lawright

Printed in USA

03-3166-03