EMS 8000

Zilog

User's Guide

September 1982

EMS 8000

User's Guide

EMS 8000
TABLE OF CONTENTS

Chapter 1. Introduction

Chapter 2. Unpacking and Installation

Chapter 3. EMS 8000 Features and Capabilities

Chapter 4. User Screens

Appendix A. EMS 8000 Tutorial

Appendix B. Load/Send Protocol

List of Illustrations

List of Tables

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

EMS 8000 is a high-end Emulation subsystem for emulating Z8001, Z8002, and Z8003 microprocessors in real time at clock speeds up to 6 MHz. The Z-LAB™ concept of emulation treats the in-circuit emulator as an intelligent peripheral. This technique allows independent selection of the software development and emulation debug environments. Zilog offers the S8000™ Z-LAB as a sophisticated software development system that can serve as the host for EMS.

The following chapters are provided in this manual. Chapter 4 serves as the user's reference manual for operation of EMS and provides various examples of EMS commands.

Chapter 1	Introduction
Chapter 2	Unpacking and Installing the EMS 8000
Chapter 3	Features and Capabilities
Chapter 4	Commands and Examples

Each different host that is used with EMS requires a special communications package on a media (tape, diskette, etc.) that the host can read. Host communications packages are currently available for the following hosts:

- S8000 Z-LAB
- PDS 8000™
- MCZ-1
- MCZ-2
- VAX UNIX[1]

A Field Applications Engineer (FAE) from your local Zilog sales office should be your first point of contact regarding any questions or problems with EMS.

---

[1]UNIX is a trademark of Bell Laboratories; Zilog is licensed by Western Electric Company.

1.2 COMMAND SYNTAX

The following command syntax is used throughout this manual:

- A single set of square brackets [*n*] indicates that n is optional.
- A set of square brackets with options [Opt 1, Opt 2, Opt 3] indicates that one option should be chosen from within the brackets.
- <CTRL> indicates that the terminal keyboard control key is to be used in conjunction with another key.

CHAPTER 2

UNPACKING AND INSTALLATION

2.1 OVERVIEW

The various hosts, terminals, and target systems that can be selected for use with the EMS 8000 are discussed in this chapter. In addition, basic cable connections and installation procedures are also described. The EMS 8000 needs to be unpacked in such a way that it can be reshipped if service becomes necessary (unpacking procedures are described in Section 2.3). The initial power-up and booting up of the system is also discussed, and the overall specifications of EMS are provided.

2.2 UNITS AND SYSTEMS THAT CAN BE CONNECTED TO EMS

A CRT terminal (from our list of supported terminals in Table 2-2) and a host system are required for the minimal EMS configuration. A target system can be connected to EMS, or programs can be debugged in EMS's internal mappable memory. In addition, a logic analyzer with an external trigger can be connected to EMS. The trigger output (located on the EMS front panel) provides a high-going pulse when user-programmed trigger conditions have been satisfied. Figure 2-1 illustrates the most common EMS 8000 configuration; the functions of the units included in this configuration are shown in Table 2-1.

Table 2-1. EMS 8000 System Units and Functions

Unit	Function
CRT Terminal	The CRT terminal provides the user interface for EMS as well as to the host system during Transparent mode.
Host System	The Host system is used initially to download the EMS 8000 monitor. The user can communicate directly with the host (via Transparent mode) to develop software that can be downloaded into the user's target memory. Target memory can also be uploaded into the host computer and saved for future use. The S8000 ZLAB is an example of a host computer that is supported by EMS.
Target System	The target system is the system to be emulated.
Logic Analyzer	Optional; may be used with the EMS 8000 to record digital signals synchronized with programmed triggers.

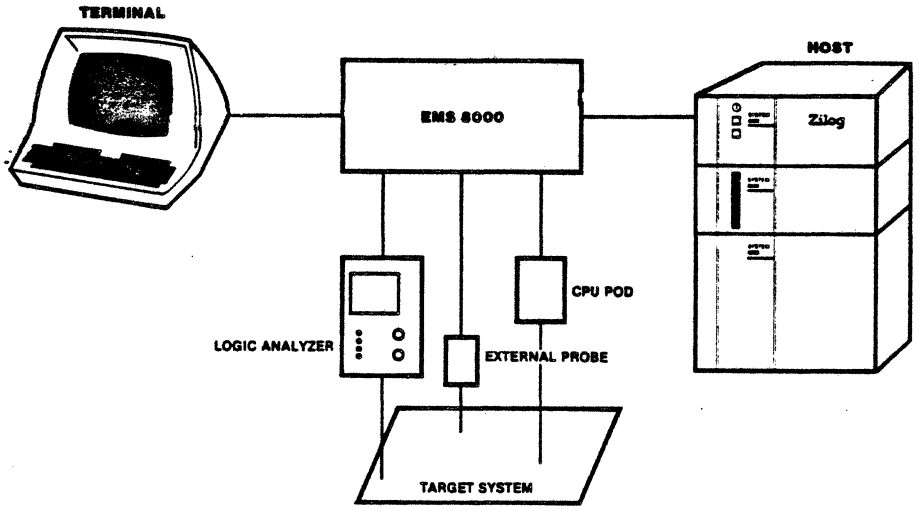


Figure 2-1. EMS 8000 System Configuration

2.2.1 CRT Terminals Compatible with EMS 8000

The number of CRT terminals that are supported by EMS are listed in Table 2-2.

Table 2-2.
CRT Terminals for Use with the EMS 8000

Terminal	Manufacturer
ADM 31	Lear Siegler
VT100	Digital Equipment Corp.
VT-Z 2/10	Zilog

2.2.2 Recommended Host Systems

The MCZ-1, MCZ-2, and S8000 are Zilog development systems that can serve as a host for the EMS 8000. The MCZ-1 and MCZ-2 use the RIO operating system, whereas the S8000 uses the ZEUS™ (UNIX) operating system.

2.2.3 Target System Requirements

EMS currently supports any target system that uses either a Z8001 or Z8002 microprocessor.

2.3 UNPACKING

Every EMS 8000 is fully inspected and tested before shipment to ensure that it meets Zilog specifications. All equipment is packaged for safe transit under normal freight-handling conditions and should arrive ready to be installed. The packing material has been specially designed to protect the EMS 8000 unit during shipment. **Should reshipment be necessary, repack the EMS 8000 unit in the original carton using the original packing material received with the unit.**

Before unpacking the system, inspect the shipping containers for signs of possible damage of the primary unit or the CPU Pod during transit. **If shipping damage is suspected, claims with the freight carrier should be filed immediately. Your Field Applications Engineer (FAE) should be notified of such action.**

The complete EMS 8000 system can be shipped in more than one carton. A minimum operational system consists of the following three items, which are each ordered separately:

- EMS basic unit
- CPU Pod module for the processor in your application
- A host software package with manual

Any optional items ordered, such as CPU Pod modules, a Probe Interface board, External Probes or an additional Mappable Memory board will be present with your order.

Unpack all items carefully and inspect them for external damage, such as dents, broken switches, loose connections or damaged cables. Any sound of loose items inside the cabinet is evidence of damage. **If damage is evident or suspected, make no further attempt to install or operate the system. Notify your FAE of the problem.**

2.4 INSTALLATION

The procedures for installing the EMS 8000 system include:

- Mounting the front panel
- Connecting EMS to the CRT terminal
- Connecting EMS to the host
- Connecting the POD cables to the EMS emulator board
- Connecting the target cable to the target
- Setting the DIP switches on the EMS rear panel
- Installing the power cord

Refer to Sections 2.5 and 2.6 for the initial power-up procedure and operational checkout of the EMS 8000 subsystem. Before beginning the installation procedures, be sure that the EMS 8000 unit is placed securely on a level surface, and that ample space is allowed for ventilation on the sides and back portion of the unit.

2.4.1 Front Panel

Before installing the front panel (shown in Figure 2-2), check the following items:

1. All PC boards should be securely seated in the EMS Backplane. The ends of the ejector tabs should be pointing towards the center of the PC board.

2. The Emulator board (eighth card slot from the top) contains two connectors for the two CPU Pod cables. The Probe Interface board (optional) also contains connectors for its External Probe cables. Make sure that the ejector tabs on these connectors are positioned straight (forward) so that they will not interfere with the mounting of the front panel; failure to do so could result in permanent damage to the ejector tabs.

3. Grasp the front panel by each end with the labels legible. As you are positioning the front panel in place, make sure that all four of the ejector tabs come through the appropriate opening, labeled CPU POD CABLES.

4. Press on each corner of the front panel to snap it into place.

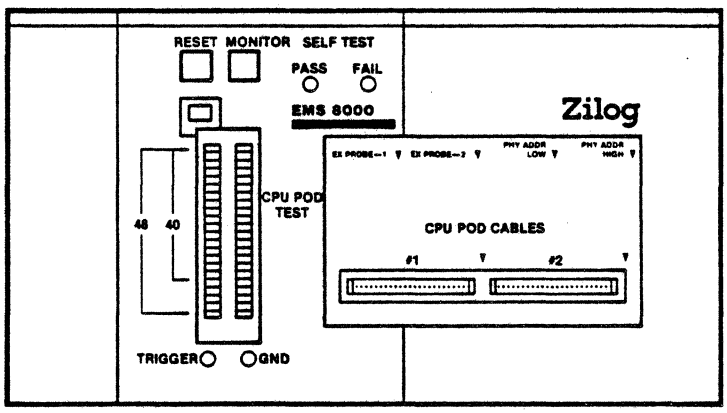


Figure 2-2. EMS Front Panel

2.4.2 Cable Connections

The following subsections describe the cables and the procedures used to connect the various parts of the EMS system.

2.4.2.1 CPU Pod

The first cables to be connected are the CPU Pod cables that connect the CPU Pod to the EMS 8000. These six-foot cables are permanently attached to the Z8001/2 CPU Pod and are marked CPU-POD-CABLE-01 and CPU-POD-CABLE-02 at the 50-pin connectors. Use the following instructions to connect these two cables through the front panel of the EMS unit to the emulator board (see Figure 2-2).

1. Plug the top Pod cable connector (marked CPU-POD-CABLE-01) into the EMS front-panel 50-pin header labeled CPU POD CABLE #1 (left connector). This cable is keyed and should be oriented so that the connector marked CPU POD CABLE 1 faces up.

2. Plug the bottom Pod cable connector (marked CPU-POD-CABLE-02) into the EMS front-panel 50-pin header labeled CPU POD CABLE #2 (right connector). This cable is also keyed and should be connected so that the connector marked CPU POD CABLE 2 faces up.

2.4.2.2 Target Cable

The target cable that connects the target to the CPU Pod is installed next. This cable is identified by the blue wire along its side, which indicates the location of Pin 1.

The size of the DIP connector on this cable differs for the Z8001 and Z8002 CPU Pod. The connector on the target cable (on the EMS Pod side) is marked with a part number and a triangle that indicates Pin #1. The other end of the target cable has a special molded plastic protector plugged into the pins of the connector.

1. Plug the target cable connector (50-pin for Z8001, 40-pin for Z8002) into the EMS Pod, ensuring that Pin 1 of the cable matches Pin 1 of the EMS Pod. Pin 1 of the EMS Pod is located on the top-right side of the Pod (see Figure 2-3).

2. Carefully remove the plastic protector from the connector pins on the other end of the target cable and plug the connector pins into the target system (ensuring that Pin 1 is in the proper position). (Pin 1 of the target cable is indicated by the blue wire running along the side of the cable.) The plastic protector should be placed over the target system plug whenever the plug is removed from the target system.

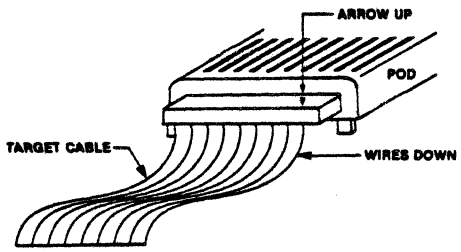


Figure 2-3. CPU Pod and Target Cable Connection

2.4.2.3 Terminal Cable

The terminal cable that connects the terminal to EMS is connected next. A flat cable with 25-pin "D" connectors on each end is supplied to connect the terminal to the EMS 8000. (Refer to Table 2-2 for a list of supported terminals.) The only lines that are actually required in this cable are the Transmit, Receive, and Ground signals (so three wire cables are acceptable).

To connect the terminal to EMS, perform the following steps:

1. Unscrew the knurled knob of the rear-panel door (see Figure 2-4). This uncovers the area used to connect the various cables to the EMS unit.

2. Connect one end of the RS-232 cable to connector J4 located (upper-right) on the rear panel of the EMS 8000 unit. This portion of the rear panel is shown in Figure 2-5.

3. Connect the other end of the RS-232 cable to the selected terminal.

4. See Section 2.4.3 for instructions on setting the EMS 8000 baud rate used for EMS terminal and host communications.

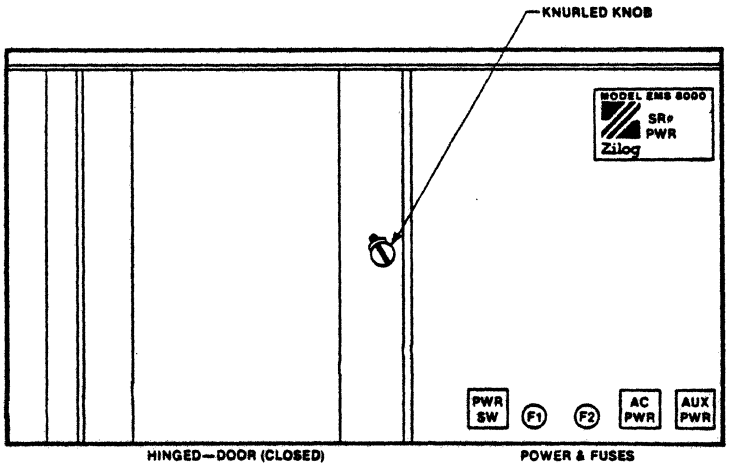


Figure 2-4. EMS 8000 Rear Panel, Before Installation

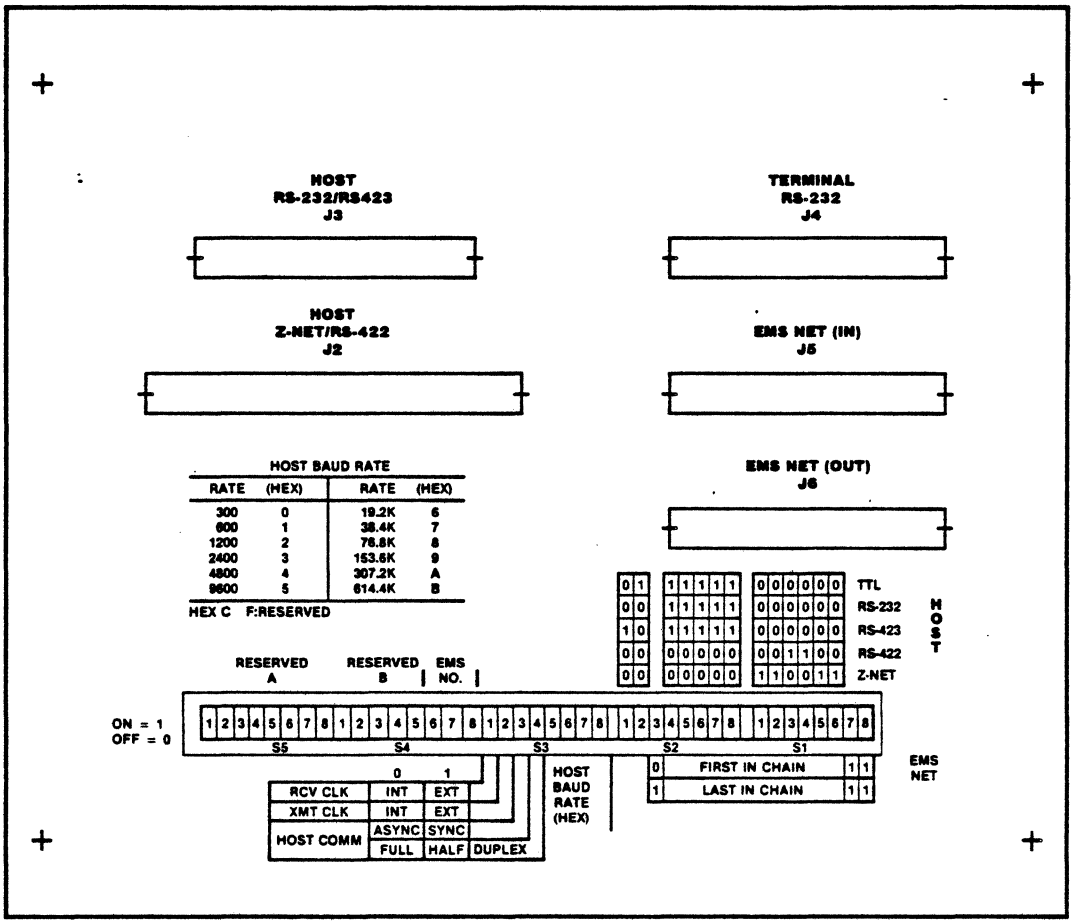


Figure 2-5. EMS Rear Panel (Access Door)

2.4.2.4 Host Cable

The host cable is identical to the terminal cable and can be connected as follows:

1. Connect one end of the 25-pin "D" connector to the EMS rear-panel jack J3, which is located on the top left of the EMS 8000 rear panel (see Figure 2-5).

2. Connect the other end of the 25-pin "D" connector to the appropriate port on the host system.

3. Refer to Section 2.4.3 for instructions on setting the EMS 8000 rear-panel switches for EMS and host communications.

2.4.3 Switch Settings

Rear-panel switch settings recommended for the MCZ-1, MCZ-2 and S8000 hosts are given in Table 2-3.

1. Set bits 5-8 of S3 to represent the hexadecimal value that corresponds to the desired baud rate for the host computer (see Table 2-4). The terminal should always be operated at 9600 baud.

Table 2-3. Suggested Rear-Panel Switches for MCZ-1, MCZ-2 and S8000 Hosts

Switch	Bit	State	Comments
Switch S1	Bits 1-6	Off	Select RS232
Switch S2	Bits 1-2	Off	Select RS232
Switch S2	Bits 4-8	On	Select RS232
Switch S1	Bits 7-8	On	ON for EMS Net
Switch S2	Bit 3	Off	MS first in Net

Table 2-4. Host Baud Rates and Corresponding Values for the "S3" Switch Settings

Baud Rate	Bits 5 6 7 8	Hex
300	0 0 0 0	(0 Hex)
600	0 0 0 1	(1 Hex)
1200	0 0 1 0	(2 Hex)
2400	0 0 1 1	(3 Hex)
4800	0 1 0 0	(4 Hex)
9600	0 1 0 1	(5 Hex)
19.2K	0 1 1 0	(6 Hex)

2.4.4 Power Cord and AC Power Selection

Connect the power cord that was shipped with the EMS 8000 unit to the power receptacle on the rear panel of the EMS unit (refer to Figure 2-4). If your EMS 8000 is being used outside of the United States, contact your local FAE before performing the initial power-up procedures or the initial operation checkout.

2.4.5 External Probes

The External probes are connected through the front panel of EMS with the red stripe of the cable on the right. The External Probe option requires the External Probe board, and one or more External Probes. External Probes can be used for supporting Physical addressing from Memory Management Units (MMUs), tracing logical levels (sampled on Data Strobe), and detecting glitches.

The locations for Physical Address probes (two are required) and the external Logic probes are shown on the front panel as EXT PROBE 1, EXT PROBE 2, PHYS ADDR LOW and PHYS ADDR HIGH. When using the EXT PROBE 2 field to trace CPU signals as configured in the Allocation screen (see Section 4.4.8), the External Probe board is not required. The usage and requirements of the External Probes are shown in Table 2-5.

Table 2-5. External Probes Usage and Requirements

Use	Requirements
Trace CPU	None
Trace Ext 1 and 2	Ext Probe board and two pods
Physical Memory	Ext Probe board and two pods
Trace Ext 1 and 2 and Physical Memory	Ext Probe board and four pods

2.4.6 Mappable Memory

The standard EMS 8000 configuration includes one Mappable Memory board with 64K bytes of high-speed static RAM. An optional Mappable Memory board can be purchased to increase mappable RAM capacity to 126K bytes. This board is placed in the bottom card slot. Table 2-6 shows the jumper placements for the standard and optional Mappable Memory boards.

Table 2-6. Jumper Placements for Mappable Memory

Board	Jumper Placement
Mappable Memory #1 (64K bytes)	Jumper is factory-placed between E1 and E2.
Mappable Memory #2 (62K bytes)	Place jumper between E2 and E3.

2.5 INITIAL POWER-UP PROCEDURES

1. Turn ac power on for the EMS 8000.
2. Turn on the CRT terminal.
3. Turn ac on for the target.

2.6 BOOTING THE SYSTEM UP

1. Make sure that the host communications package is resident on the host computer and accessible from the port connected to EMS. Host communications packages for the MCZ-1, MCZ-2, and S8000 are available from Zilog.

2. Press the RESET button on EMS (located on the left side of the front panel next to the MONITOR button). The following message should appear immediately on the terminal screen:

   Please select one of the following terminal types and enter the terminal code.
   0) ADM-31 1) TVI-920 2) VT-100 3) VT-Z 2/10
   ?

3. At this point, select one of the above terminals by number. EMS will respond with the version number of the monitor software, and will automatically place EMS into Transparent mode. The following message should appear on the screen:

   EMS 8000 Version X.X
   Transparent Mode

4. The user's terminal is now linked directly with the host. If the host has not yet been initialized, it should be done now.

   Execute the host communications program "HOST". This program is available on floppy disk for the MCZ-1 and MCZ-2 host systems and on tape for the S8000 host system (this program takes a few seconds to load). HOST is an active program that runs on the host computer to monitor EMS requests to the host for loading or saving functions.

5. Press the BREAK key on the EMS terminal, or the MONITOR button on EMS. This action initiates downloading of the EMS monitor software from the host system. After downloading is completed, the user is in the Change screen (see Chapter 4 for a description of EMS commands). The following messages should appear on the screen:

   Loading EMS
   Loading EMSA
   Loading EMSB
   Loading EMSC
   Loading EMSD
   Load completed...Stand by

A few seconds after the "Load completed" message, EMS should display the Change screen. If an error occurs during loading, the initial terminal request will be repeated. The links and the host should be checked for problems, and to make sure that the EMS files are present on the host. The procedure would then be repeated.

2.7 RETURNING TO TRANSPARENT MODE

After booting the monitor, Transparent mode can be reentered at any time from any screen by typing the control character "<CTRL> T" (for Transparent mode). The EMS monitor responds with "Transparent Mode". At this time the EMS terminal behaves as if it were directly connected to the host system. If the communications program HOST is still active on the host system, it should be terminated by an "X <RETURN>". This character must be an upper case "X" (otherwise the host will appear not to respond to terminal commands). EMS remains in Transparent mode until either the BREAK key on the terminal or the MONITOR button on the EMS front panel is pressed. Before pressing the BREAK key or the MONITOR button, the HOST program must be re-executed if any loading or saving of files is to be done. Pressing either the BREAK key or the MONITOR button immediately returns control to the EMS screen that was active before entering Transparent mode.

2.8 INITIAL OPERATIONAL CHECKOUT

In order to verify the proper attachment to the Target system, the user should type the following sequence.

As the user becomes more familiar with the operation of EMS, some of these steps can be eliminated.

1. Type a "<TAB> D" to go to the Debug screen.

2. Type an "E" to select the "Edit" command, "R" to select Register mode, "P" to select PC, and <RETURN> to allow the PC and FCW to be entered. This allows the beginning of emulation to be at the point desired in the correct location. Otherwise, the CPU will begin execution at location 0 with a status of zero as if it were code.

3. Type <CTRL> G to begin emulation. A reverse video feedback area in the lower right-hand portion of the screen should read "Running".

4. Verify that the target operation is identical to operation with an actual Z8000 CPU. If the target has components needing a reset not achieved during a power-up, it should be reset manually after start of emulation.

5. After identical operation is verified, type "<CTRL> C" to terminate emulation and return control to EMS. Note that the feedback area in the lower right-hand portion of the screen displays "User Break".

2.8.1 Bad Clock or Timeout Messages

If the message "BAD CLOCK! Fix it" appears, check the following:

- The Z8000 CPU clock may not meet the Z8000 ac or dc specifications given in the "Z8001/Z8002 CPU Product Specification" (document number 00-2045-A0).
- A bad connection may exist between the target CPU socket and EMS.
- Power may not be present in the target system.
- The EMS CPU Pod or the EMS Emulator board may be faulty.

If the message "TIMEOUT!..." appears, check the CPU bus signals BUSRQ-, RESET-, or WAIT-. If any of these signals are shorted or held Low for very long periods of time, system emulations cannot be completed. These emulations are necessary to fetch user registers, display user memory and to access target resources. If the problem persists after checking the items listed above, call your FAE.

2.9 EMS 8000 SPECIFICATIONS

EMS Terminal	25-pin "D" connector(using Txd, Rxd, and Gnd). RS-423 buffering is provided. Terminal baud rate is 9600 baud.
Host Computer	25-pin "D" connector (using Txd, Rxd, and Gnd). RS-423 buffering is provided. Host baud rate can be 38.4K baud maximum.
EMS Group Break	25-pin "D" connectors in a daisy chain.

Electrical Specifications:

Target Clock Rate	500 KHz to 6 MHz
Voltage	90-140 V 180-260 V
Fuse	5 A
Power	670 W
Line Frequency	47-63 Hz
Phase	1

Note

Voltage conversion **MUST** be performed by authorized Zilog personnel. Contact your Zilog Sales Office to request this service.

Dimensions of the EMS 8000 Unit:

Width	20.0 in.	(50.8 cm)
Depth	25.0 in.	(63.5 cm)
Height	7.0 in.	(17.8 cm)
Unit Weight	65.0 lbs.	(29.5 kg)
Shipping Weight	80.0 lbs.	(36.4 kg)

Transportation/Storage Environment

Temperature	-50° - +125°C
Humidity	5%-95% (No condensation)

Operating Environment: 0-40° C

Humidity	10%-90% (No condensation)

CHAPTER 3

EMS 8000 FEATURES AND CAPABILITIES

3.1 OVERVIEW

This section describes the basic hardware and software features of the EMS 8000. A description of the EMS screens and commands is provided in Chapter 4 of this manual.

The EMS 8000 is a state-of-the-art in-circuit emulation subsystem that supports the hardware and software design engineer in developing products using the Zilog Family of microprocessors and peripheral components. EMS provides a CPU plug to replace the CPU chip in the user's target system, which makes installation simple and immediate.

The EMS 8000 is modular in design, with a friendly screen-oriented, self-prompting user interface. The user has full access to the target microprocessor's registers, memory, and I/O space. Special I/O and Memory Management Unit (MMU) functions are supported with substitution of physical addresses for logical addresses. The user can start, stop, monitor, and step execution in real time.

The EMS 8000 interfaces with either UNIX or RIO-based Zilog systems. The following Zilog microcomputers can be used as the EMS Host system: System 8000, MCZ-2, PDS 8000 or the MCZ-1.

The EMS 8000 is an intelligent peripheral whose monitor software is downloaded during the initial power-up. This feature allows the EMS software to be upgraded easily. Other features, such as complex triggering, a large real-time trace buffer, and a large mappable memory space provide the user with powerful debugging tools during the development cycle.

Additional features of the EMS 8000 include:

- A real-time partitionable trace module for multiple recordings of program execution. The partitioned trace feature also enables the user to capture events that are separated in time. Trace fields that are recorded include CPU address, data, status, control and External Probe bits: 64 bits of target information are recorded each cycle.

- Three parallel trigger resources are provided as building blocks in configuring the trigger, trace and timing functions. These comparators provide an effective aid for specific debugging strategies and support address ranging, sequential conditions, Enable and Disable of other triggers, ORing, and bit-masking.

- Performance measurements are supported with a General-Purpose counter designed to aid in benchmarking critical software routines.

- 64K bytes (expandable to 126K bytes) of high-speed, static mappable memory can be mapped with a resolution of 2K bytes anywhere in the user's memory space. Each 2K block can be declared unprotected, write-protected, non-existent, code or data only, or normal or system only.

- A pulse output feature permits a logic analyzer to be used as part of the development system allowing logic data to be captured synchronous to program execution.

- A Group Break feature is provided to allow several EMS units to begin and terminate emulation together. This feature is useful in debugging network systems.

3.2 HARDWARE DESCRIPTION

EMS is a full-featured emulation peripheral. The heart of EMS 8000 is a Central Controller Unit (CCU) with 256K of dynamic memory and 16K of ROM. The CCU contains the monitor program that provides a screen-oriented user interface. The CCU operates continuously, allowing the user to follow the progress of an emulation and monitor breakpoints during the emulation process in real-time.

The other EMS modules include a two-board Trigger module, a real-time Trace module, an External Probe interface module, a Mappable Memory module, and a microprocessor "Personality" module with a CPU Pod. A block diagram of the EMS 8000 hardware is provided in Figure 3-1.

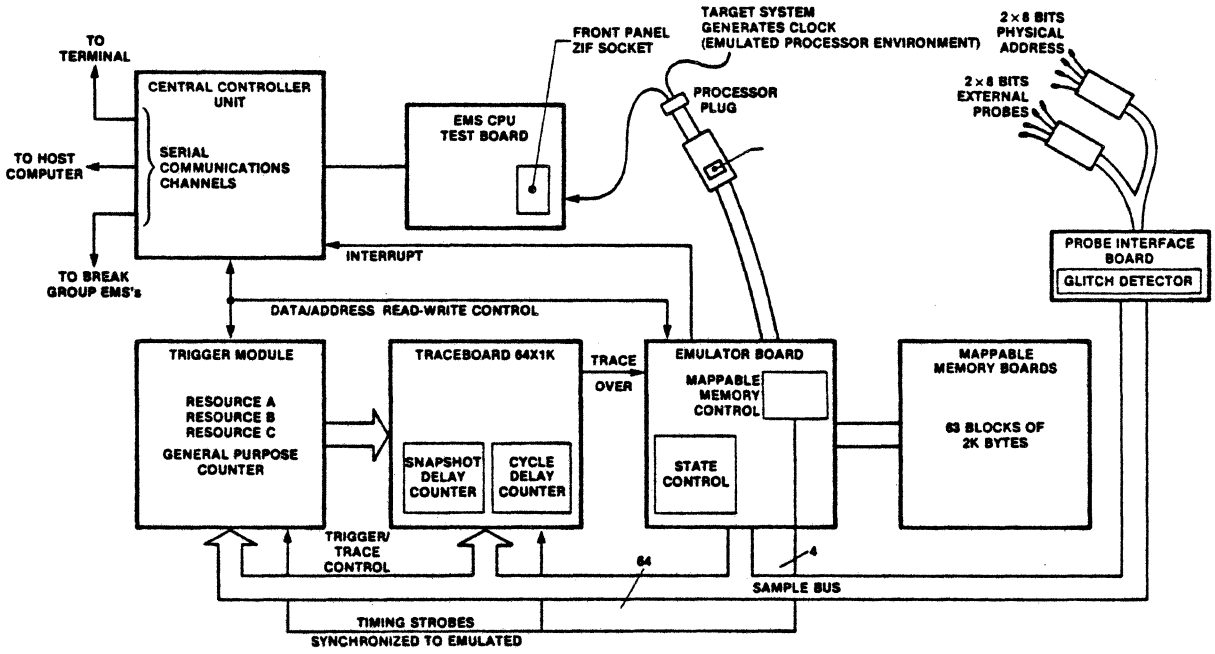


Figure 3-1. EMS 8000 System Block Diagram

3.2.1 Central Control Unit (CCU)

The CCU module contains a 4 MHz Z80A microprocessor with up to 16K bytes of EPROM for bootstrapping and low-level software routines. Up to 256K bytes of bank-switched RAM on this board is reserved for the system software. The CCU maintains all communications to and from the user terminal and the development system host via the Rear Panel board. In response to user inputs, the CCU programs the other EMS modules that are reponsible for specific EMS tasks. The CCU also contains the necessary hardware for the EMS Group Break (a feature that allows several EMS units to start and stop emulation simultaneously).

3.2.1.1 The Rear Panel

There are six connectors and five DIP switches on the rear panel (see Figure 2-5). Table 3-1 describes the functions of the connectors.

Table 3-1. Rear Panel Connector Assignments and Functions

Connector Label	Functions
J1	Internal connector to connect the Rear Panel board to the EMS Backplane. This connector is not visible to the user.
J2	Reserved for future connection to Z-NET network transceiver or a high-speed host computer (RS-422 compatible).
J3	Used to connect EMS to a host computer with RS-232 or RS-423 electrical buffering (RS-232 pin assignments using 25 pin "D" connector).
J4	Used to connect EMS with the terminal.
J5	Daisy-chain input for EMS NET (used for Group Break feature).
J6	Daisy-chain output for EMS NET (used for Group Break feature).

3.2.2 Sample Bus

EMS 8000 uses a 64-bit-wide sample bus. With the Z8000, these bits are assigned as follows:

- 8 bits for segment number
- 16 bits for offset address
- 16 bits for data
- 8 bits for CPU Control (ST_0-ST_3, N/$\overline{S}$, R/$\overline{W}$, B/$\overline{W}$, $\overline{BUSACK}$)
- 16 bits for External Probes or 8 bits for External Probes and 8 bits for CPU signals

A detailed diagram of the sample bus is shown in Figure 3-2.

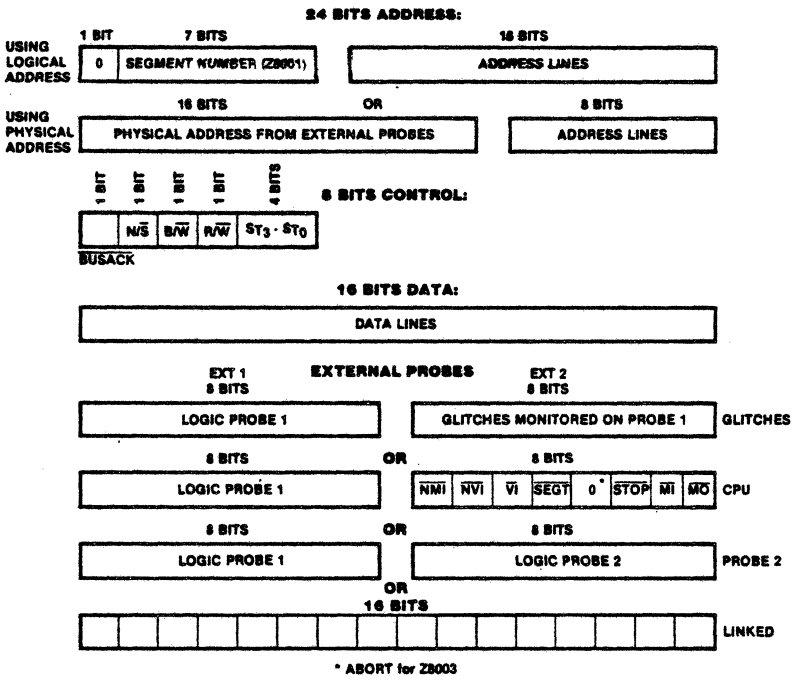


Figure 3-2. EMS 8000 Sample Data Bus

3.2.3 Trigger Module

Three separate resources monitor the 64 bits of sample data each cycle, looking for preprogrammed conditions. The resources can be programmed to look for "0", "1", or "don't care" for each of the 64 bits. The address, data and External Probe fields can each be used in equal, not equal, greater than, or less than comparison modes. Each of the three resources can be programmed to look for a different pattern.

Resource B is simple and produces an output whenever its pattern is found.

Resource A is more sophisticated and can look for a sequence of up to five different cycles, each with its own pattern-matching qualifications. Resource A also has two Address fields that can be logically ANDed to provide in-range or out-of-range cycle detection. These fields can also be ORed to provide two addresses per pattern. Resource A also includes a Cycle Limit counter that allows the user to specify that this sequence must occur within a certain number of cycles.

Resource C can look for a sequence of up to eight different cycles, each with its own pattern-matching qualifications. Resource C can be used the same way as Resource A or Resource B, or it can be partitioned into an Enable/Disable resource. When used as an Enable/Disable resource, Resource C serves as a "window" to qualify other resources.

A 16-bit General-Purpose counter resource can be used in conjunction with the above resources. The general-purpose counter can be a pass counter or can count cycles or elapsed time. As a pass counter it waits for an event of the trigger resources to occur a certain number of times before taking any action. As a cycle or elapsed time counter, the number of machine cycles or T-States between two events is accumulated. This feature aids in performance measurements such as benchmarking.

All of the above resources can be used as building blocks to control the Trace module. These resources specify which cycles are traced, trigger the recording of trace snapshots, and terminate emulation. These resources can be used for any of the trace functions. For example, Resource A can trigger the trace while all cycles matching Resource B are traced. The resources can also be combined logically; for instance emulation can end if any cycle meets the conditions of Resource A OR Resource B OR Resource C.

3.2.4 Trace Module

The Trace module is a 64 x 1024 bit-partitionable memory that records 64 bits of sample data from the CPU bus each time a qualified trace event is recognized. The Allocation screen (discussed in Chapter 4) allows the user to specify which resources affect this recording process. In addition, a delay feature allows cycles to be recorded long after the trigger resource has become true.

A group of cycles recorded relative to a trigger is called a "snapshot" or "partition." A snapshot provides a small window of execution history. The trace memory can be partitioned into several smaller sized snapshots or the entire trace memory can be allocated to a single snapshot. Multiple snapshots are discussed in Section 3.2.4.2. One snapshot is recorded for each occurrence of the Trigger. This feature allows many occurrences of subroutines or specific CPU instructions to be recorded without disturbing real-time operation. The emulation can be terminated after a specified number of snapshots (determined by the user) has been recorded.

The number of snapshots and bus cycles can be combined as shown in Table 3-2.

Table 3-2. Possible Snapshot and Bus Cycle Combinations

Number of Snapshots	Number of Qualified Bus Cycles
1 snapshot	1024 qualified bus cycles
2 snapshots	512 qualified bus cycles
4 snapshots	256 qualified bus cycles
8 snapshots	128 qualified bus cycles
16 snapshots	64 qualified bus cycles
32 snapshots	32 qualified bus cycles
64 snapshots	16 qualified bus cycles
128 snapshots	8 qualified bus cycles
256 snapshots	4 qualified bus cycles

Two special counters (different from the General-Purpose counter) can be used in conjunction with the trace functions. They are the Cycle Delay counter and the Snapshot Delay counter:

- The Cycle Delay counter gives the user the ability to position cycles within a snapshot relative to a trigger (much like post-, center- and pre-trigger recording with a logic analyzer).

- The Snapshot Delay counter allows the user to stop emulation after a specified number of snapshot triggers after the emulation breakpoint has been recognized.

Collecting cycles before a trigger event is called pre-trigger recording. Collecting cycles around a trigger event is called center-trigger recording. Collecting cycles after a trigger event is called post-trigger recording.

For the pre-trigger trace (see Figure 3-3), recording stops immediately on the trigger condition. For the center-trigger trace (see Figure 3-4), after the trigger occurs there is a delay equal to half the size of the snapshot before recording is stopped. At that time, half of the snapshot contains a record of what happened before the trigger occurred, and the other half contains a record of what happened afterwards. For the post-trigger trace (see Figure 3-5), after the trigger occurs, there is a delay equal to the size of the snapshot memory and then recording is stopped. The snapshot contains all of the bus transactions beginning with the trigger event itself.

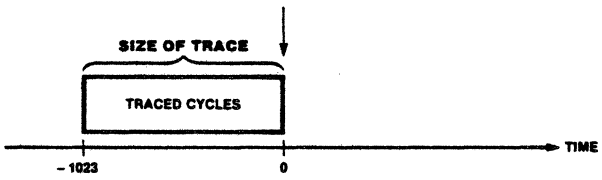


Figure 3-3. Pre-Trigger Trace

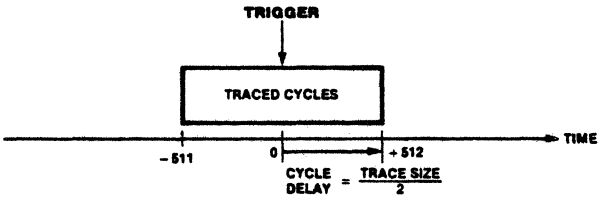


Figure 3-4. Center-Trigger Trace

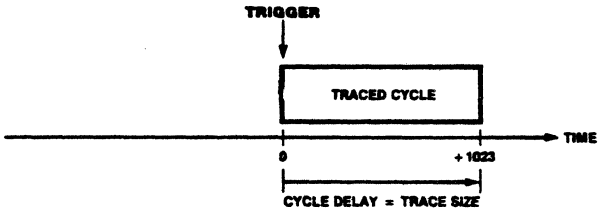


Figure 3-5. Post-Trigger Trace

The only difference between pre-trigger, center-trigger and the post-trigger tracing is the number of cycles after the trigger event before recording is stopped. In EMS, this number can be continuously adjusted: for instance, the trace memory, which is 1024 cycles long, can contain 923 cycles before and 100 cycles after the trigger (note that the trigger cycle is also included, see Figure 3-6). It is also possible to have a delay that is longer than the trace snapshot size (see Figure 3-7). This feature is useful if the user's software program seems to "get lost" after a fixed amount of time following a certain identifiable point in the program. As an example, if a delay of 9024 is used to stop tracing, the number of cycles traced would be in the range from 8000 to 9024 after the trigger. Note that the trigger cycle is not included in this case (see Figure 3-7).

If the trigger condition never occurs, or no trigger condition is specified, then recording occurs until the user intervenes with a manual break "<CTRL> C". Cycles are numbered with respect to the trigger point. The trigger itself is cycle number 0. The fifth cycle before the trigger is cycle number -5. The 7046th cycle after the trigger (in this case the trigger buffer is written over many times) is cycle number +7046. Cycle numbering is shown in Figures 3-3 through 3-7.

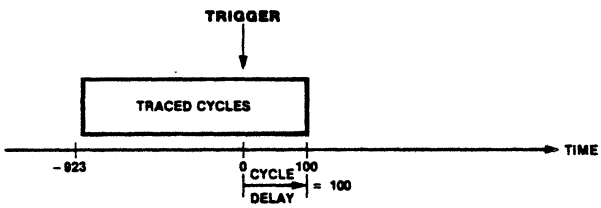


Figure 3-6. Center Trigger with Arbitrary Cycle Delay

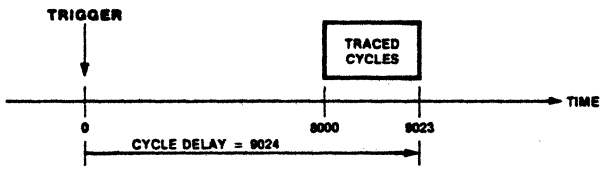


Figure 3-7. Post-Trigger with Long Cycle Delay

3.2.4.1 Trace Qualification

Trace qualification is the ability to restrict the cycles recorded in the trace memory to those that meet specified requirements. This is particularly useful for monitoring all of the accesses to a certain device, memory address, or range of addresses. The various forms of trace qualifications in EMS are described in the following paragraphs.

Any of the three resources (A, B, or C) can be used as trace qualifiers (see Figure 3-8).

It is possible to use Enable and Disable (part of Resource C) to define time slots during which tracing occurs. Selecting Enable means that cycles are qualified only after the Enable point. Selecting Disable means that cycles are not qualified after the Disable point (see Figure 3-9).

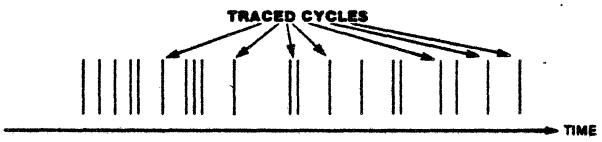


Figure 3-8. Qualified Cycles

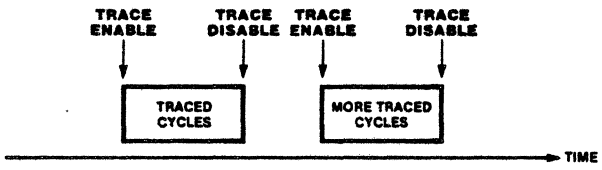


Figure 3-9. Enable/Disable Trace

Enable and Disable can be allocated to form a "qualification window" for cycles to be recorded in the trace memory. This allocation requires Resource C to be used in Enable/Disable mode. For example, Figure 3-10 shows the trace memory recording only those cycles that meet certain conditions and that occur in a certain qualification window.

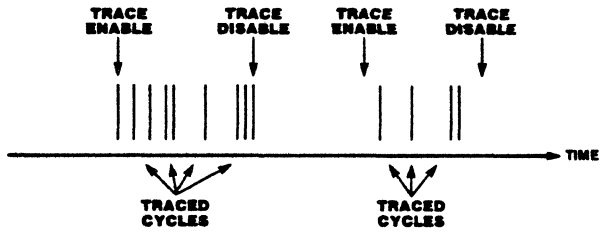


Figure 3-10. Qualification with Enable/Disable

NOTE

Triggers and final triggers are always traced, regardless of qualification.

When a trigger condition is used with a delay for center- or post-trigger tracing, and this condition is combined with trace qualification, the delay is a number of qualified cycles. This situation is illustrated in Figure 3-11.

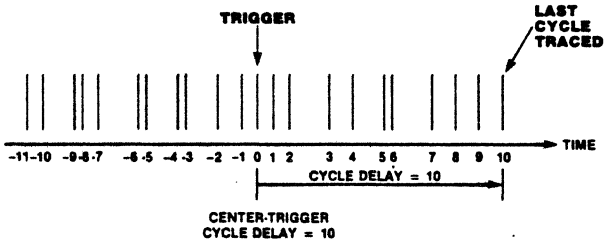


Figure 3-11. Numbering of Qualified Cycles

3.2.4.2 Multiple Snapshots

If the user has an intermittent hardware or software problem it may be desirable to collect, in one trial, execution data before or after each of several triggers. The ability to collect execution data in this way (called multiple snapshots) is a unique feature of EMS.

In order to capture multiple snapshots, the 1024-cycle trace memory is partitioned into a collection of smaller trace memories. The number and size of the partitions is adjustable (some situations require a larger number of triggers, and other situations require a larger amount of data around each trigger). As indicated in Section 3.2.4, the memory can be divided into 2 partitions of 512 cycles, 4 partitions of 256 cycles, and so on, down to 256 partitions of 4 cycles. When emulation begins, instead of overwriting the entire trace memory, recording is restricted to the first partition. At some point, a trigger condition occurs, followed by the cycle delay (for center- or post-trigger tracing). After the cycle delay occurs, recording in the first partition stops and proceeds on to the second partition. Each trigger

condition that occurs thereafter causes cycles to be traced in the next partition. When the last partition has been filled, the first one will be overwritten (as long as triggers keep occurring). This process continues until a condition occurs that stops emulation. This condition can be one of several things:

- A certain number of triggers have occurred (shown in Figure 3-12).
- A breakpoint or a manual break has occurred (shown in Figure 3-13).
- A breakpoint has occurred, followed by a certain number of triggers (shown in Figure 3-14).

Snapshots are numbered relative to the emulation breakpoint. If there is no emulation breakpoint, they are numbered relative to the end of emulation; therefore, the oldest one is -n and the most recent one is 0. Snapshot numbering is shown in Figures 3-12 through Figures 3-14.

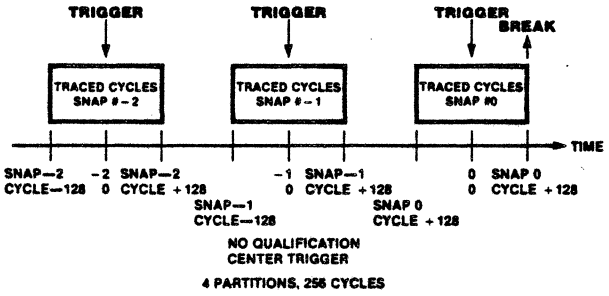


Figure 3-12. Multiple Snapshots, Breaking After a Certain Number of Triggers (Break After 3)

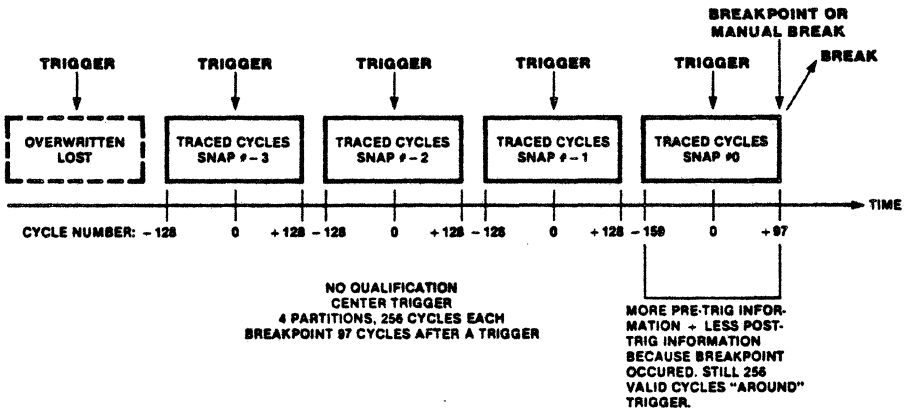


Figure 3-13. Multiple Snapshots Break with Breakpoint or Manual Break

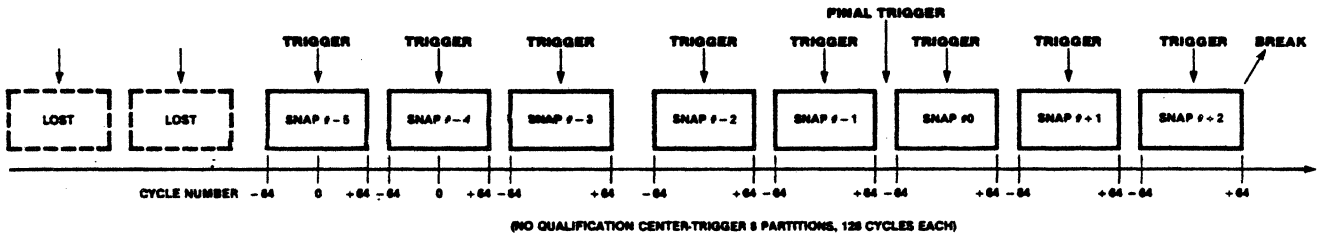


Figure 3-14. Multiple Snapshots, Break n Snapshots after Final Trigger (n = 3)

If multiple snapshots are used in center- or post-trigger mode and if the trigger occurs again while the cycle delay is in progress, then recording in that partition is stopped immediately and continues in the next partition. This process is shown in Figure 3-15.

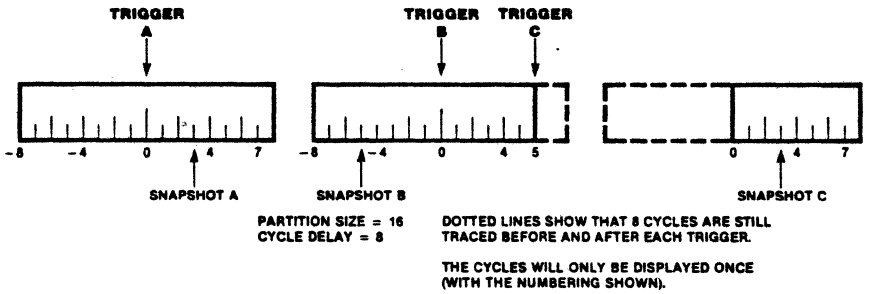


Figure 3-15. Triggers Occurring During a Cycle Delay

During the cycle delay after the last trigger, any extra triggers that occur will not cause recording in that partition to be stopped. This allows the user to gain full post-trigger information from the last trigger that occurs. The extra triggers appear in the trace display as "Triggers", and the numbering of the cycles is from the first trigger. This is shown in Figure 3-16.

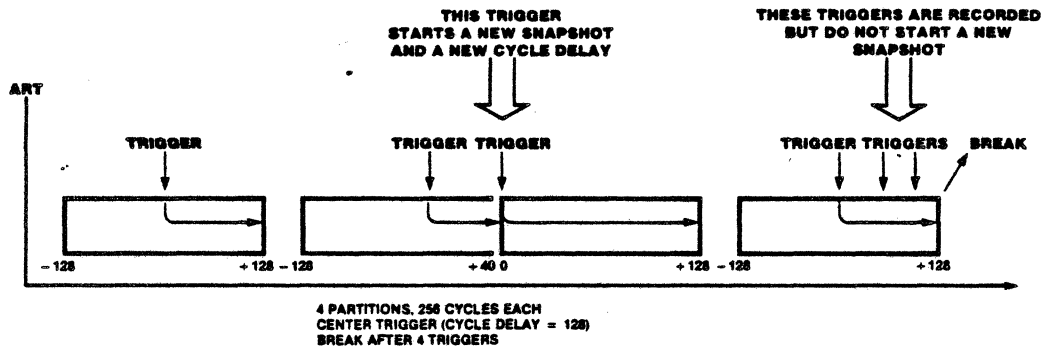


Figure 3-16. Triggers Occurring During the Last Cycle Delay

3.2.5 Emulator Module

The EMS Emulator module is the "Personality" module located between the user's Z8001 or Z8002 socket and the rest of EMS. The user's Z8000 socket is connected to a pod by a short cable. The Z8000 Pod is connected to EMS by two long cables. The pod contains a high-speed Z8000 CPU and all of the time-critical circuitry necessary to connect or isolate the target system from the pod's Z8000 CPU. It also contains buffer circuitry to drive signals to the Z8000 Emulator board.

The Z8000 Emulator board contains a controller to start and stop the Z8000 CPU. The CPU can be stopped by any one of the following:

- A command from the terminal, "<CTRL> C".
- An occurrence of a trigger breakpoint, or a trace-full condition.
- An EMS Group Break.
- A memory protect violation.

The Emulator board transfers data between the CCU board and the Z8000 CPU. Local memory is provided to the Z8000 to run system emulations. These emulations are responsible for fetching the Z8000's internal registers and accessing the various resources of the target system. Another function of the Emulator board is to gather and supply the address, data, and cycle status to the Sample Data bus for the Trigger and Trace modules. Timing signals for the sample data are also supplied. The Emulator board produces the memory timing signals and protection bits for the mappable memory array.

3.2.6 Mappable Memory

Mappable memory allows regions of target memory space to be substituted by internal high-speed static RAM located in EMS. EMS is shipped with 64K of mappable memory and is expandable to 126K (2K of the second memory board is reserved for EMS mapping operation). The memory mapper provides mappable memory for up to two Z8001 segments at the same time. Mapping can be performed separately for Code and Data, and for Normal and System modes. It also allows these areas to be write-protected, or designated as nonexistent. These protection modes can apply to target memory or to EMS memory. No Wait states are introduced by the use of mappable memory (except when Physical Address probes are used with a slow MMU).

3.2.7 External Probes

External Probes are an optional feature offered with EMS. These probes allow the user to record logic transitions (sampled with the trailing edge of Data Strobe of the target CPU) along with instruction execution. For example, the user may wish to verify that an output port has changed state based upon a specific CPU output instruction. The External Probes can also be used to detect short transitions (as small as 50 ns) that may occur between Data Strobes.

Up to four probes can be connected to the External Probe board. Two of these probes are dedicated to the above logic state functions and two are dedicated to supporting memory management addressing. EMS can trigger, trace, map (substitute) memory, and break instruction execution based upon physical addresses (as opposed to logical addresses from the Z8001 CPU). This allows EMS to support the debugging of large memory systems with MMUs. The External Probe cables are connected through the EMS front panel (see Section 2.4.5).

3.3 SOFTWARE DESCRIPTION

The EMS 8000 software provides a friendly debug environment for microprocessor-based systems. The user interacts with EMS through screens with menu prompts. Each screen fills a standard 24 x 80 CRT display, and is devoted to a particular function. The information contained in this section provides an overview of the EMS 8000 software; refer to Chapter 4 for a complete description of the screens and their commands.

In general, the user enters parameters and commands by moving the cursor to a desired field and selecting an option from a constantly displayed menu line. An error message is displayed in reverse video when an option is selected that conflicts with a previously-selected option.

The user interface consists of five user screens, a "Help" screen, and a "Change" screen. The five user screens are: Allocation, Pattern, Configuration, Debug, and Map. The user can move from one screen to another by typing <TAB> followed by the letter capitalized in the desired screen name. The <TAB> character exits the current screen and returns control back to the Change screen.

- The Change screen allows the user to change from one screen to another. This screen is entered when the EMS monitor is downloaded and whenever the <TAB> character is typed. The available screens are indicated on the menu line.

- The Allocation screen is used to allocate EMS resources to specific actions.

- The Pattern screen is used to program patterns for allocated resources.

- The Configuration screen is used to configure global parameters for the system.

- The Debug screen allows the user to set Instruction Breakpoints, Display memory, Edit Memory, Edit Registers, Edit I/O, Edit Special I/O, Begin Emulation, Upload and Download from the Host computer, Compare Memory, Fill Memory, Move Memory, Search Memory, Step Execution, Display the Trace Memory, Setup Watch area, and Zero triggers.

- The Map screen is used to substitute EMS mappable memory in place of target memory.

- The Help screen (which is invoked by typing the question mark character "?") provides a helpful reminder of some system-wide EMS commands that are not provided by the menu line. The Help screen can be entered from any other screen by typing "?" and is exited by typing <RETURN>.

CHAPTER 4

USER SCREENS

4.1 OVERVIEW

The user communicates with EMS through five menu-driven screens and two support screens. A screen is designed to fit on a standard display terminal, 80 columns wide by 24 lines long. Each main screen in EMS is dedicated to a particular function. Descriptions of all the screens are given in Table 4-1.

Table 4-1. EMS Screen Descriptions

Screen	Function
	Menu-Driven Screen
Configuration screen	Used to configure global features of EMS.
Allocation screen	Used to allocate the EMS event recognition resources to specific actions such as tracing and breakpoints.
Pattern screen	Used to enter event patterns for the EMS recognition resources.
Map screen	Used to substitute EMS mappable memory in place of target memory.
Debug screen	Used to examine and edit memory, registers, I/O; to display the trace results; to begin emulation; to set software breakpoints; to turn on and off the "watch" area; to upload and download files to and from the host computer; and to single/multiple step through program execution.
	Support Screens
Change screen	Serves as a "dispatch" screen to allow the user to change from one screen to another.
Help screen	Provides a helpful reminder of some system-wide EMS command characters that are not provided by the menu line.

4.1.1 Selecting Screens

The user can move between screens at any time by typing the "<TAB>" key to enter the Change screen, followed by the capitalized letter of the screen to be entered.

Example: To enter the Configuration screen from any user screen, type "<TAB> C".

4.1.2 Cursor Manipulation

As each main screen is displayed, the cursor moves automatically to the first modifiable field on the screen. The cursor can be moved only to user-modifiable fields. The modifiable fields consist of numbers or key words selected from a menu line. The keywords have a single letter capitalized that identifies the option. Fields that consist of only capital letters are not modifiable. "All Caps" fields are used to clarify the user display and to visually locate option fields. The Right, Left, Up, and Down arrows move the cursor to the next user-modifiable field.

4.1.3 Menu Facility

When the user enters a field, a list of available options is displayed on the Menu line.

Example: If the cursor is positioned to the EXT 2 option field of the Allocation screen, the options available for that field (displayed on the bottom line of the screen) look like this:

Probe 2 Glitch Cpu

The user can then enter any one of the capitalized letters from the menu line (P, G, or C) to select the desired option. The menu line also contains status information indicating break conditions and macro status. In most cases conflicts between menu selections are impossible. If EMS detects a menu selection conflict (such as deselecting a resource with its Pass Count still enabled) this condition is displayed on the menu line. A carriage return clears the error condition and re-prompts the user for another menu selection.

4.1.4 Rules for Entering Data in a Field

There are two types of fields: multiple-choice Option fields, and Data fields in which a number or other string of characters (such as a filename) can be entered.

4.1.4.1 Option Fields

Options are selected by typing the single letter associated with an option, or by scrolling through the available options with the space bar. The available options are displayed in the menu line.

4.1.4.2 Data Fields

1. Addresses, data, and other strings are simply typed in data fields. A prompt in the menu line describes the type of data expected.

2. Characters entered into variable fields can be deleted with ASCII DEL.

3. A "<CTRL> U" removes a new entry and restores the previous value in a field.

4.1.4.3 Special Rules for Pattern Screen Data Entry

Masking Address and Data Bits

All numbers can contain an "X" for binary or hexadecimal digits that signify "don't care" values in those positions. Binary "don't care" digits can be expressed in a nibble of binary digits enclosed in parentheses. This nibble can be used as a hex digit. For example, 7F(1X01)3XD represents a valid number for a Pattern screen address or data entry.

When entering addresses or CPU data, the user can optionally enter a second number separated by an "&" (ampersand). The "&" symbol is used to specify a mask of bits to be used in the comparison.

Selecting Comparison Mode

Comparisons must be specified for address, data, and external fields on the Pattern screen. An address, data, or External field can be blanked out by setting the comparison field to "-".

The comparison operators are:

- = (equal to)
- # (not equal to)
- < (less than or equal to)
- > (greater than or equal to)

Note

For descriptions of the functions allocatable via the Allocation screen and their associated rules, refer to Section 4.3. For descriptions of the functions and Resource patterns defined via the Pattern screen (and their associated rules) refer to Section 4.4.

4.1.5 Using Control Keys for Starting, Stopping, and Stepping

One of the primary functions of EMS 8000 is to switch a CPU between a mode in which it is "emulating" or "running," (i.e., executing the user program and acting like a Z8000 CPU), and a mode in which it is suspended (i.e., "break" state). The latter mode allows EMS access to target registers, memory, and I/O. A system status indicator at the lower right-hand corner of the screen shows the current mode at all times.

Three keys can be used to control these modes at all times in all modes of all screens:

<CTRL> G — (Go) is used to start emulation, using all of the current register values including the PC. Emulation continues until a breakpoint or memory violation occurs, or until "<CTRL> C" is typed. After "<CTRL> G", the system status indicator shows "Running".

<CTRL> C — (Cancel) is used to stop emulation, unconditionally and immediately. After "<CTRL> C", the system status indicator shows "User Break".

If emulation breaks for some other reason, the terminal will beep, and the reason ("Trig Break", "Write Viol", "Mem Viol") is displayed by the system status indicator.

<CTRL> X — (Execute single instruction) executes one instruction from the current PC. The message "Trig Break" is then displayed by the system status indicator.

When the system is "Running," the user can:

- Switch freely between screens.
- Change options, such as enabling or disabling control signals, on the Configuration screen (the effect is immediate).
- Options can be selected on the Debug and Map screens.

However, Data fields may not be filled in, and commands may not be executed. To regain control, typing "<CTRL> C" stops emulation.

4.2 CONFIGURATION SCREEN

The Configuration screen is entered by typing "<TAB> C" from any screen. The user can move the cursor to any user-modifiable field on the screen by pressing the arrow keys on the terminal. To modify a field on the screen, the user enters the first letter of one of the available options, or a number. The space bar key also scrolls through the available options for a particular option field.

Example: When the Configuration screen is first entered, the cursor is positioned on the RESET field on the screen and the Menu line reads:

Enabled Disabled

The user can enable or disable the CPU Reset input by entering "E" for enable, or "D" for disable.

4.2.1 Configuration Screen Fields

The following fields on the Configuration screen display EMS status and allow the user to select default conditions for the EMS 8000.

4.2.1.1 CPU Type Field

EMS currently supports the Z8001, Z8002, or Z8003 microprocessors. EMS can automatically sense which processor pod is connected to EMS.

4.2.1.2 CPU Signals Fields

EMS allows the user to selectively enable or disable individual Z8000 control signals from the target. These CPU control signals are listed in Table 4-2.

Table 4-2. CPU Control Signals

Signals	Description
RESET	Reset
BUSRQ	Bus Request
WAIT	Wait
STOP	Stop
NMI	Nonmaskable Interrupt
NVI	Nonvectored Interrupt
VI	Vectored Interrupt
SEGT	Segment Trap (Z8001 or Z8003 only)
ABORT	Instruction Abort (Z8003 only)

4.2.1.3 Break Fields

The BREAK fields control whether or not an emulation break occurs in case of a memory write-protect violation or mode violation (see Table 4-3).

Table 4-3. BREAK Fields, Configuration Screen

Labels	Description
BREAK ON WRITE-PROTECT VIOLATION	If Yes, a write-protect violation will cause a Breakpoint.
BREAK ON MEMORY ACCESS VIOLATION	If Yes, access to memory mapped as nonexistent or incorrect access to memory protected as code or data will cause a break.

Memory access violations occur whenever a protected memory area is invalidly accessed, such as a data access to memory mapped as code protected when the code and data spaces are not configured as separate.

4.2.1.4 Memory Fields

The Memory fields are used to enter Wait states for systems that require longer memory access time. Each Wait state inserts one additional CPU clock cycle into the memory access. The user can enter up to seven Wait states for mappable memory accesses or user memory accesses. This field is useful when using External Probes for physical address substitution. If physical addresses are being used in the EMS system and mappable memory is required, at least one mapped access Wait state is required. Two Z8001 segments can be mapped to internal EMS memory. The EMS standard configuration comes with 32 blocks of 2K byte resolution. If an optional Mappable Memory board is ordered, the number of available mappable memory blocks is increased to 63.

The EMS can also be configured to allow separate System and Normal and separate Code and Data memories. Data and Stack memory can not be separated.

4.2.1.5 Address Fields

Logical — Selects logical addresses for Trace and Mapped Memory.

Physical — Selects physical addresses for Trace and Mapped Memory. (Physical addresses require the External Probe option.)

These fields of the Configuration screen are used to specify whether logical addressing or physical addressing is used as the address field for the trigger, trace, and memory mapper. Logical addressing is the default mode. Logical addressing means that the address from the CPU is used for the address fields automatically. Physical addressing can be used if an optional Probe Interface module is installed. The user connects these external probes to the output of the target MMU. EMS then substitutes these physical addresses for the CPU logical addresses. The probes should be connected (see Section 2.4.5) through the EMS front panel into the PHYS ADDR HI and PHYS ADDR LOW connectors. PHYS ADDR HI replaces the segment number (eight bits instead of seven) and PHYS ADDR LO replaces the upper byte of the 16-bit offset. The lower eight bits of the address are always supplied by the CPU.

4.2.1.6 Internal Operation and Refresh Cycles Field

The options are as follows:

Ignored — Causes internal CPU cycles and refresh cycles NOT to be recognized for triggering or tracing. This is the default mode, which causes only cycles relevant to the executing program to be traced, and uses the trace capacity effectively.

Traced — Trace internal CPU cycles and refresh cycles if they meet the other trace qualifications. This mode can be useful for initial hardware checkout, familiarization with the Z8000 CPU, or verification of refresh frequency.

4.2.1.7 Group Break Field

The Group Break is an option that can be enabled or disabled. If the Group Break is enabled then EMS starts and stops emulation together with any other EMS which also has its Group Break enabled. The EMS units are connected in a daisy chain via cables and 25-pin D connectors on the rear panel.

4.2.1.8 Mode Field

The Mode field allows the user to switch between an Emulator mode and an Analyzer mode. The Emulator mode causes EMS to act as a normal emulation system in which the CPU stops execution when breaks or user halt is issued. The Analyzer mode allows the CPU to continue running after the initial start of emulation and a break. This pseudo-halted state allows the user to examine trace history and change EMS setup while the emulated target is still running. This is useful in systems in which a shutdown in CPU functions is not desirable. A switch back from Analyzer to Emulator mode while EMS status is not "running" causes a halt to the Z8000 CPU. While in Analyzer mode, starting emulation and breaking from it results in associating trace resources to and away from the CPU. While in Analyzer mode, and with the status not "running," CPU registers are not available for viewing.

4.2.1.9 Clock Frequency Field

The Clock Frequency field displays the clock frequency of the target system. This is measured during the initialization of EMS and can be used by EMS to display timing results in μsec instead of T states at the user option.

4.3 ALLOCATION SCREEN

The Allocation screen is entered by typing "<TAB>" for the Change screen and then "A" from any screen. The Allocation screen is used to allocate EMS resources by assigning them to specific actions. Once the Allocation screen

has been set up, the user programs the resource match patterns by changing to the Pattern screen. The various modes of operation (Break/Trace mode, Timer mode, and Counter mode) are discussed below. The Allocation screen allocates the resources but they must be "activated" in the Pattern screen before they have an effect.

The Allocation screen is preset to a useful default that allows first-time users to skip the Allocation screen and to use the Pattern screen directly. This default allows trace qualification, trace triggering, a hardware breakpoint, and up to 16 instruction breakpoints. It is recommended that the user become familiar with the default Pattern screen and basic EMS capabilities before using the Allocation screen.

The Allocation screen must be used to select the following EMS features: performance evaluation modes (timer and event counter); trigger or qualifier enable/disable; ORing of resources; separate trace trigger and hardware breakpoint; alternate probe modes (glitches, CPU inputs, linked probes).

4.3.1 Mode Field

The Mode field is located at the top left of the Allocation screen and allows the user to select one of three modes: Break/Trace, Counter, or Timer mode. The first time that the Allocation screen is entered, the default value for the Mode field is "Break/Trace mode." Each successive time that the Allocation screen is entered, the Mode field indicates the mode that is presently selected. Table 4-4 lists the Mode selections for the Allocation screen. The Allocation screen can be reset to the default value or cleared of all settings by entering "<CTRL> Z" followed by "d" for default or "0" for clearing.

Table 4-4. Mode Selections, Allocation Screen

Mode	Description
Break/Trace mode	Used to allocate instruction breakpoints (using Resource B), trace triggers, hardware breakpoint, trace qualification, and pass counting.
Timer mode	Used for performance evaluation. Timer mode allows the user to determine the execution time between two events. Any of the three Resources (A, B, or C) can be used for the start and end points. Either the start or end point can be qualified by Enable/Disable. The execution times can be recorded in the trace memory. Emulation can be halted if the execution time exceeds a specified value. Time can be counted in T states, machine cycles (based on Address Strobe), or microseconds.
Counter mode	Used for performance evaluation. Counter mode allows the user to count the number of events from Enable to Disable. Either Resource A or B can be counted. In this mode, the Enable sequence is always the Start Event and the Disable sequence is always the Finish Event.

4.3.2 Resources and Actions

The basic architecture of EMS consists of several resources that can be assigned to specific debugging actions via the Allocation screen. The basic resources are listed with their attributes in Table 4-5.

Table 4-5. Resources and Their Associated Attributes

Resource	Attributes
A	Up to five patterns (bus patterns) deep; Address Ranging.
B	One cycle deep; can be allocated for software breakpoints.
C	Up to eight patterns deep; If Resource C is selected as a sequence event, Enable/Disable cannot be selected.
Enable/Disable	Used as a "window" to qualify Resource A or Resource B. Can be defined as "Enable only," or "Disable only."

The resources described in Table 4-5 can be configured to perform the following actions:

Trace qualifier	Filters what gets recorded in the trace memory. The default is "trace everything" (including internal and refresh cycles if tracing on internal and refresh cycles option is selected on the Configuration screen).
Trace Trigger	The event before, after, or around which each snapshot (partition) of cycles is recorded.
Breakpoint	Stops emulation either immediately or after a specified number of trace triggers.
Instruction Breakpoint	Stops emulation on instruction fetch at specified locations. Up to sixteen such locations can be specified. This action can be allocated by Resource B only, and can only be implemented with RAM target memory or mapped memory.

Refer to Section 4.3.3 for the specific rules that apply to configuring resources and actions via the Allocation screen.

4.3.3 Rules for Assigning Resources to Actions

Each resource can be associated with an action, but certain rules must be followed. In most cases, these rules are enforced by restrictions in cursor movement. The cursor can only be moved to places where options are available without "breaking the rules." These places are identified on the screen by the symbol "-" (which acts as a place holder). In a few cases, violations of the rules cause an error message to appear on the bottom line in reverse video. When an error occurs, type "<RETURN>" to return to the conflicting field and resolve the conflict. Usually an error is caused by the user attempting to create a configuration that cannot exist (e.g., removing a resource with pass counting still assigned).

1. Once a Resource (A, B, C, or Enable/Disable) is configured to perform a specified action, it cannot be used for another action. This rule is enforced by restricting cursor movement after each resource is assigned. Only one entry can appear in any column on the Allocation screen. The user can reassign a selected resource prior to leaving the Allocation screen, as long as the reassignment adheres to the rules. A resource can usually be reassigned by positioning the cursor to the currently-assigned location and typing the character "-" (or hitting the space bar). The cursor is then moved to the new location and the resource identifier (e.g., "A" for Resource A) or the spacebar is typed (hitting the space bar toggles the resource from active to inactive).

2. Only Resource B can be associated with instruction breakpoint.

3. Resource C can be a sequence event or Enable/Disable (but not both) and can be configured for any of the actions. Refer to rules 4 and 5 (below) for further information on the Enable/Disable event.

4. Enable or Disable can be used as a trace qualifier, causing the titles "trace enable" and "trace disable" to be displayed on the Pattern screen. If Enable/Disable is the only resource configured to trace qualification then all cycles are traced after the Enable sequence and before the Disable sequence. If either Resource A or Resource B is used as a trace qualifier, and Enable or Disable is selected, only qualified cycles after Enable and before Disable are traced.

5. Enable or Disable can also be used to qualify Resource A or B when they are configured as a trace trigger, breakpoint, timer start event, or timer finish event. In any of these cases, either Resource A or B must be selected in order for Enable or Disable to be selected. Likewise, Enable or Disable must be deselected before Resource A or Resource B can be deselected.

4.3.4 Pass Counting

Pass counting is a feature that is available only in Break/Trace mode. Pass counting allows the user to specify the number of occurrences (for example of Resource A) before the associated action takes place. When pass counting is selected, the Allocation screen changes to include parentheses around the events that can be pass counted.

4.3.4.1 General Rules for Selecting Pass Counting

The following rules apply to the use of the Pass Count option:

1. The Pass Count option is available in Break/Trace mode only.

2. Pass counting must be enabled before it can be applied to the actions. Pass counting is enabled by entering "Y" (for Yes) in the Pass Count field. Pass counting is disabled by entering "N" (for No) in the Pass Count field.

3. Only trace triggers and breakpoints can be pass counted. Trace qualifiers cannot be pass counted.

4. Pass counting for Resource A, B, or C is indicated by entering an "n" in the field next to the selected Resource, whereas pass counting for Enable (when Resource C is used as an enable/disable instead of a sequence event) is indicated by entering "m" next to the Enable field. Only one pass count symbol "n" and one enable pass count symbol "m" can be entered per screen.

5. The user can pass count part of a complex resource by entering "n" inside the parentheses (next to the pass-counted resource), or the user can pass count all of a complex Resource by entering "n" outside the parentheses. If Enable/Disable is selected, the Enable portion of this Resource can be pass counted by entering "m" next to the Enable Resource. The values for "n" and "m" can be different.

6. The value of "n" or "m" is programmed on the Pattern screen. If a value of zero or one is programmed for "n" or "m", no pass count occurs. Only values of two or more cause pass counting to occur.

Examples:

BREAKPOINT (nA or B)

This example means that "n" occurrences of Resource A or 1 occurrence of Resource B will cause the emulation to break. The value of "n" and the patterns that comprise Resource A and Resource B are programmed on the Pattern screen.

TRACE TRIGGER n (A or B)

This example means that "n" occurrences of either Resource A or B are required to trigger the trace once. The values of "n", Resource A and Resource B are programmed on the Programming screen.

TRACE TRIGGER (A or nB) after m Enable before Disable

This example means that the trace trigger occurs after the "m"th time that the Enable pattern happened when either Resource A's pattern occurred or the "n" occurrence of Resource B's pattern and the disable pattern had not yet occurred.

4.3.5 Timer Mode

The Timer mode option, which can be selected via the Allocation screen, is very similar to Break/Trace mode. In Timer mode, EMS records the amount of time from the start resource to the finish resource. The start and finish resource can be either Resource A, Resource B, or Enable and Disable. If the start resource occurs again after the finish resource, a new interval is begun, the count is reset to zero, and sequence begins again.

The Timer mode can be set to cause a break if the count exceeds a certain value or it can be set to halt after a number of finished events.

4.3.6 Counter Mode

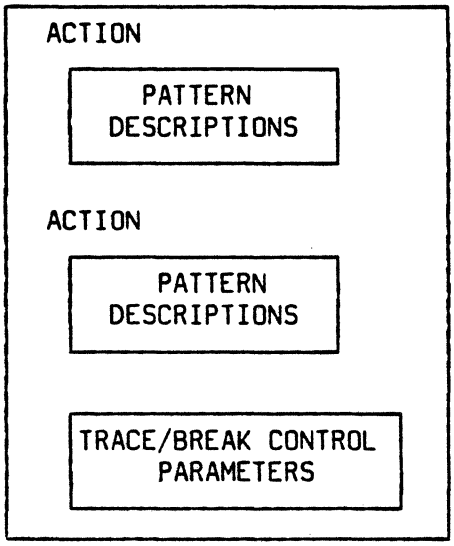
The Counter mode option, which is also selectable via the Allocation screen, is used to count the number of occurrences of Resource A or B after Enable and before Disable. If Enable occurs again after Disable, then a new interval begins and the count is reset to zero.

The counter can also halt emulation of the count exceeds a value or upon completion of a number of Disable events.

4.4 PATTERN SCREEN

The Pattern screen is used to enter the desired patterns for each of the resources that were assigned on the Allocation screen. The Pattern screen is dynamically created by the allocating of resources to actions on the Allocation screen.

The Pattern screen is generally laid out as shown:



The majority of the screen (i.e., starting from the top), is used to specify pattern descriptions consisting of the address, data, status, and external patterns of the events to be recognized with the pass counter. The remainder of the screen (i.e., lower/bottom portion) contains fields that contain trace information, such as breaking control snapshot size, and the number of snapshots to be traced.

Only actions with resources allocated on the Allocation screen are shown on the Pattern screen, in the same order as the Allocation screen. If Enable or Disable is used to restrict a resource to a particular time window, that resource is displayed on the screen between Enable and Disable, because it must occur in time between Enable and Disable.

If emulation is begun after the Pattern screen is programmed, then tracking of pattern matching is shown by an arrow that indicates which pattern is being looked for. The number of the pass count completed and traced cycle counts is also displayed so the user can know at any time how much of the pattern and/or pass count has been completed. Generally, the occurrences can complete and halt emulation before the Pattern screen is updated, even the first time, but this feature allows the user to know where in the pattern the events were completed if the completion is not accomplished.

```
-------------------------------------------------------------------------------
[P] [L] ADDRESS 1  [L] ADDRESS 2   [L] DATA  CYC CdRwBwSn [L] EXT 1  [L]EXT 2
-------------------------------------------------------------------------------
```

Pattern Descriptions

Each pattern consists of one line on the screen. The line consists of an Enable field for the whole line (when disabled, the remainder of the line is blanked), followed by Enable/Logical fields for each of the Address, Data, Status, and External Probe Trace fields. Once the pattern line has been enabled, each of the Address Data or Ext fields can be programmed by entering one of the logical characters (= equal, # not equal, > greater than, or < less than) and then entering the value to be recognized in the following fields. The cycle status (ST_3-ST_0 value can be specified by entering the CPU status mnemonic as shown at the bottom of the screen when in the field or by entering the binary value with the option of "X" being a "don't care." The remaining status is specified using the letter of the desired state. In all cases, leaving a field with a "-" results in "don't care".

4.4.1 Pattern Enable Field

The Pattern Enable field for each line of a resource is located in the left position in the pattern line. The Pattern Enable field is always visible as either a "-" (not used) or as a "*" (enabled). This field enables and disables the entire row for each pattern. This field allows the Pattern screen to be "clutter free" and also allows the EMS monitor to determine how many of the available patterns in each resource are actually being used. A pattern (row) is enabled by positioning the cursor to the left field and typing the character "*". If a pattern is enabled, modified, and later disabled (by positioning the cursor to the pattern enable field and typing "-" to disable), the information entered in this pattern "reappears" if this pattern is ever re-enabled. This feature allows the user to specify several patterns in the same resource and to enable only those that are needed. EMS monitor software automatically "packs" all visible pattern entries for a resource, even though some patterns in the sequence are not used. (Each enabled line will be examined in sequence and disabled lines will be ignored). For example, this allows the second, fourth, and seventh event of Resource C to respond as the first, second, and third occurrence of a match.

4.4.2 Logical Fields

The Logical fields (shown as "[L]" above) are used to control the relation of the Address, Data, and Ext Probe fields. The Logical fields are invoked by positioning the cursor on the desired logical field and selecting a logical operation from the menu line. If the logical operator field is blanked (by selecting "-"), then the corresponding entry field is also blanked and its value is interpreted by EMS as "don't care." Re-enabling the field by entering one of the logical characters redisplays the field with its previous value restored.

4.4.3 ADDRESS 1

To use a trigger resource to monitor the Address bus, the Address fields are used. Resource A, wherever allocated, has two address fields: ADDRESS 1 and ADDRESS 2. Resources B and C use only ADDRESS 2. Data is entered in both ADDRESS 1 and ADDRESS 2 in the same way.

Values for ADDRESS can be entered as a 6-digit hexadecimal number "sshhhh", which combines both the optional segment number and the address offset. "Don't care" hexadecimal digits can be entered as "X".

Examples: BC6CXD, 4D21, 23, XXXXXX

The first entry, BC6CXD, sets the segment number to BC and the offset to 6CXD; X is a "don't care" hex digit.

The second entry, 4D21, sets the offset to 4D21; the previous segment number is unchanged.

The third entry, 23, sets the offset to 0023; the previous segment number is unchanged.

The fourth entry clears the address field.

Entry for the Z8002 is the same as Z8001 except that only the four digit offset is used because there is no segment associated with the Z8002.

Values can also be entered as a hexadecimal number with one or more digits specified in binary and one or more digits specified as "don't care" bits. Only 13 characters are allowed in the address field, which means that no more than two hexadecimal characters can be expressed in binary.

Examples: (110X)D(10X0), (1X01)BFD07

The first entry, (110X)D(10X0), sets the offset to any of 0CD8, 0CDA, 0DD8, or 0DDA (due to the two "don't care" bits). This pattern is displayed as 000CD8&00FEFD (see mask examples below).

The second entry, (1X01)BFD07, sets the segment number to either 9B or DB and the offset to FD07.

Values can also be entered as a hexadecimal address with a hexadecimal mask field "sshhhh&mmmmmm" (0 bits in the Hex Mask field correspond to "don't care" and 1s correspond to "ON" or a "must match").

Example: 4C44&DFE5

This example sets the offset to 01X0 1100 010X X1X0. This value is calculated as follows: First set all "don't care" bits to "0". This yields 4C44 as the Hex portion of the entry. Then compute the mask field as "0" for "don't care" bits. This yields DFE5 as the mask field. The segment number is unchanged and this entry is displayed as SS4C44&FFDFE5 where SS is the previous segment number.

4.4.4 Using Two Addresses

ADDRESS 2 is similar to ADDRESS 1 and is available for Resource A, Resource B, and Resource C. When ADDRESS 2 is used with ADDRESS 1 for Resource A, an additional field appears between the two addresses. This field is initially set to "&" (logical AND) to allow address ranging (i.e. greater than ADDRESS 1 & less than ADDRESS 2). The field can be changed to "!" (logical OR) by positioning the cursor over the "&" and selecting "!" from the menu line or by hitting the space bar. The OR "!" function can be used for an "out-of-range" comparison (less than ADDRESS 1 OR greater than ADDRESS 2). It can also be used simply to OR two addresses (equal to 4016 OR equal to 4024, for example).

4.4.5 Data Fields

The rules for entering Data fields are similar to Address fields, except that Data fields are 16 bits wide instead of 23 for the address.

4.4.6 CYC

Cycle entries ("CYC" on the Pattern screen) can be any of the following:

- <SPACE> — Results in "don't care" entry
- Symbolic entry from menu line — Results in Z8000 status
- A binary (optionally masked) value — Results in one or more Z8000 status conditions.

4.4.7 Status Lines (Cd Rw Bw Sn)

A pattern entry may require a cycle to be a CPU cycle or a DMA cycle (C/D). Other status lines consist of Read/Write (R/W), Byte/Word (B/W), and Normal/System (N/S). The menu line prompts the user for the first character of the desired status. As an example, R in the R/W field selects "Read". Using the "-" for any of the status lines indicates "don't care" for that particular status line.

4.4.8 External Probes (EXT 1 and EXT 2)

The EXT 1 and EXT 2 fields are binary fields that specify the External Probe inputs for each pattern. The first option field at the far left of the pattern line must be enabled (indicated by the character "*") for the External Probe fields to be available. The External Probe fields consist of 16 individual bits that can be programmed to "1", "0", or "X" (don't care). For External Probes, each probe is treated as a separate field. The cursor automatically advances to the next bit after a new bit has been typed in. When the External Probe field is modified, new binary digits overwrite old values from left to right. The Ext 1 field always corresponds to the eight bits of Probe 1. The Ext 2 field can be configured on the Allocation screen to perform the functions shown in the following section.

4.4.8.1 Various Uses of EXT 1

Probe 2: EXT 1 and EXT 2 are two separate 8-bit External Probes. Each External Probe field has a logical field (=, #, <, >) associated with it. All bits (except "don't care" bits) must match for the External Probes portion of the pattern to be true. (Bit 0 is to the far right.) Probe 2 has a subfield associated with it ("Linked") which allows EXT 1 and EXT 2 to be linked together as one 16-bit External Probe. The logical field associated with EXT 2 on the Pattern screen disappears when this option is used. All bits (except "don't care" bits) must match for the External Probes portion of the pattern to be true.

Glitches: EXT 1 remains an 8-bit probe. EXT 2 performs glitch detection for EXT 1. A glitch is defined as more than one transition between sample times. If a 1 is entered in bit "n" of EXT 2 then a glitch on bit "n" of EXT 1 causes this bit to be true. All bits (except "don't care bits) must match for the external probes portion of the pattern to be true. If a "0" is entered in bit "n" of EXT 2, then "no glitch" on bit "n" of EXT 1 causes this bit to be true. If an "X" is entered in bit "n", then this bit is ignored. Bits are checked by EMS on the rising edge of Data Strobe, but only on those cycles that are traced. The cycles that are traced are determined by the particular default selections on the Configuration screen and by the trace qualifier (if allocated) on the Pattern screen. When viewing the EXT 1 and EXT 2 displays, the digit in the far-right position of each field represents bit 0.

CPU: EXT 1 is an 8-bit probe. EXT 2 is internally connected to the eight CPU signals ($\overline{NMI}$, $\overline{NVI}$, $\overline{VI}$, $\overline{SEGT}$, $\overline{ABORT}$[1], $\overline{STOP}$, $\overline{MI}$, $\overline{MO}$). If a "1" is programmed in bit "n" of EXT 2, then this bit will be true if the corresponding CPU signal is active. For example, if a "1" is programmed into bit 8 of EXT 2 (the far left bit of EXT 2) then this bit will be true if NMI is active (Low) on the rising edge of data strobe of any cycle. Cycles traced are determined by the particular defaults listed on the Configuration screen and the by the trace qualifier (if allocated) on the Pattern screen. If a "0" is entered in bit position "n", then this bit will be true if the signal is inactive on the rising edge of data strobe of a traceable cycle. Entering an "X" in bit "n" means "don't care" for this bit. All bits in EXT 2 must match for the EXT 2 portion of the External Probes to be true.

4.4.9 No Time Limit

Under normal conditions, Resource A causes an action to occur after all patterns in Resource A have been recognized. In this case there is no time limit for the resource to go true. The Pattern screen provides a field (No time limit/Sequence must occur within n cycles) to require Resource A to be recognized within "n" cycles or the entire Resource is reset to look for the first pattern again. The value of "n" can be from 0-255 and is entered in decimal. The sequence restarts automatically if the "n" cycles occur before the sequence of patterns is completed. This feature is particularly useful to trace or trigger on a multiple cycle instruction with specific data transfers. A "0" entry for "n" also indicates that there is no time limit. The time limit feature is useful with two or more patterns and with "n" greater than or equal to the number of patterns in the sequence.

4.4.10 Programming the Snapshot Setup (Break/Trace mode only)

The last line of the Pattern screen is the menu line. The three lines above the menu line allow the user to program features associated with the trace snapshot. The top line in the trace snapshot area allows the user to position the trigger within the trace much like a logic analyzer that allows pre-, post- and center- triggering.

The default setting for these fields causes the trace trigger to act exactly like a breakpoint--execution breaks immediately at the end of the current instruction. "End SNAPSHOT 0 CYCLES after TRIGGER - BREAK after 1 snapshot."

[1]Applicable to Z8003; no connection for Z8001 and Z8002.

Possible examples include the following:

Examples:

1. **Start SNAPSHOT 10 QUALIFIED CYCLES Before TRIGGER**

   This is allowed if [decimal number] is less than the snapshot size.

2. **Start SNAPSHOT 10 QUALIFIED CYCLES After TRIGGER**

   This is allowed if [decimal number] is less than (64512 - snapshot size).

3. **End SNAPSHOT 10 QUALIFIED CYCLES Before TRIGGER**

   This is not allowed. This option selection produces an error message.

4. **End SNAPSHOT 10 QUALIFIED CYCLES After TRIGGER**

   This is allowed if [decimal number] is less than 64512.

This specification is a recommendation to the trace hardware. Complete instructions are always traced, and usually result in extra cycles overwriting the beginning. "Start SNAPSHOT 20 CYCLES Before TRIGGER" is safer for a post-trigger trace than "Start SNAPSHOT 0 CYCLES After TRIGGER", because it gives a "cushion" to the trace and prevents the trace activity at the end of the snapshot from overwriting the beginning. This preserves the trace information around the trigger.

To center the trigger in snapshots of size 512 qualified cycles:

1. Select option "b" in the field "# of snapshots:"

2. Select format number 1 above with [decimal number] = 256
   or select format number 4 above with [decimal number] = 256

The second line in the trace snapshot area allows the user to position the breakpoint in the selected number of snapshots (much as the trigger was postioned inside the snapshot). If a trigger and breakpoint are configured to resources, the second line in the trace snapshot area allows the following possible entries:

Examples:

1. **BREAK DISABLED**

   Always available.

2. **BREAK** **On breakpoint**

   Available if a breakpoint has been configured. Causes an immediate break, regardless of snapshots or cycles traced.

3. **BREAK AFTER 15** **Snapshots**

   Available if a trigger has been configured. Breaks after the 15th snapshot is complete.

4. **BREAK AFTER 15** **Snapshots or On breakpoint**

   Available if a breakpoint and a trigger have both been configured. Breaks after the 15th snapshot is complete, or immediately when a breakpoint occurs.

5. **BREAK** **15** **Snapshots** **After breakpoint**

   Available if a breakpoint and a trigger have both been configured, and instruction breakpoints are not used. When the breakpoint occurs, a count of 15 snapshots begins; execution breaks when the last snapshot is complete.

6. **BREAK DISABLED**

   Always available.

The last line in the trace snapshot area allows the user to program the number of snapshots (partitions) in the trace memory. The Trace memory is 1024 entries long and can be partitioned in any one of the following ways:

a. 1 partition of size 1024
b. 2 partitions of size 512
c. 4 partitions of size 256
d. 8 partitions of size 128
e. 16 partitions of size 64
f. 32 partitions of size 32
g. 64 partitions of size 16
h. 128 partitions of size 8
i. 256 partitions of size 4

(Partitioning is accomplished by entering the letter corresponding to the snapshot size.)

4.4.10.1 Timer Control Field (Timer mode only)

```
       Cycles        Earliest
COUNT  Usecs   FROM           START EVENT TO FINISH EVENT
       T-states      Latest
```

The first field on this line controls whether machine cycles (based on Address Strobe) are counted, or whether T-states (CPU clock) are counted. If "µsec" is selected, the number of T-states counted is converted to µsecs by using the measured clock frequency. If the clock frequency varies, this number will be wrong.

The second field allows the count to be reset to 0 every time a start event occurs. "COUNT FROM Earliest START" means that the count is reset only the first time; "COUNT FROM Latest START" causes the count to be reset on every Start Event.

Time Stamping/Tracing Option (Timer and Counter modes only)

TRACE ALL Counts
TRACE ALL Cycles

If "TRACE ALL Cycles" is selected, the finish event acts as a trace trigger. The trace will contain the last 1024 cycles of the last traced count interval. If "TRACE ALL Counts" (time stamping) is selected, the count for each interval will be traced along with each finish event.

Break option (Timer and Counter modes)

```
       Disabled
BREAK  if Count exceeds n
       after n Finish events
```

The first option, Disabled, requires a "<CTRL> C" to terminate emulation. The second option, "if Count exceeds n", allows breaking if no break occurs before n count. (n must be 65535 or less, and if specified in µsec, must correspond to less than 65535 T states.) The third option allows specifying exactly the number of count intervals before a break.

4.5 DEBUG SCREEN

The Debug screen allows the user to perform simple debugging operations. Debug commands are entered with a single capital letter from the menu line. Values entered in specific option fields "persist" (they are still there when the Debug screen is reentered). The display area associated with memory display or trace display (with the exception of the Watch area when active) is scrollable with the up and down arrow keys. Scrolling is only active after a command has been executed and by entering "<RETURN>"; scrolling remains active until the current command is exited by entering another "<RETURN>". When using the edit command, up and down arrows can be used to scroll through locations. "Q" is used to terminate the edit command.

Note

The scrollable display area is increased from the normal 13 lines to a total of 22 lines by setting the Watch command (area) to off.

When the cursor enters a field, that field is opened for modification. The user can then enter a number into the field. Values are entered into a field by typing the desired value. As characters are typed, the previously displayed characters are overwritten. While the user is modifying a field, the the previous value entered in this field can be restored by simply typing "<CTRL> U" (for "Undo"). However, once the cursor leaves the field, the old value is lost and cannot be recovered. The most-recently entered character can be deleted with an "ASCII DEL". To abort any of the Debug commands, the user can position the cursor at the first position in the line and enter another command letter, or enter "<TAB>" to exit debug. An upload/download can be aborted by typing "<BREAK>".

The following functions are performed on the "Debug screen":

1. Program up to 16 instruction breakpoints (software breakpoints)
2. Display target memory
3. Edit target resources (Memory, Registers, I/O, Special I/O)
4. Go (begin emulation) from current program PC or specified address.
5. Upload and download files from the host (including scripts)
6. Memory compare, fill, move
7. Step execution (single step or multiple step)
8. Display the Trace contents
9. Set up a Watch area to report on target resources after each break

The commands used to perform debug functions are explained and illustrated by examples in the following sections.

4.5.1 Breakpoint

Breakpoints work by substituting a special NOP instruction at the specified address. The Instruction Breakpoint (Breakpoint Set) must not be to ROM or other target enforced write-protected areas. If an Instruction Breakpoint is entered in a ROM address, the error message "Software Breakpoints Must be in RAM, Enter <RETURN> to Continue" appears when the user attempts to begin emulation. This message appears for each breakpoint set in ROM. After each is cleared, the emulation begins and such erroneous breakpoints are ignored. Instruction breakpoints are also operable in EMS mapped memory. The user must not set instruction breakpoints in the middle of multi-word instructions, as these words may never be accessed by the Z8000 as IF1. The setting or clearing of instruction breakpoints causes a display of the resulting condition.

Breakpoint [Display, Set, Clear]

Example: Breakpoint Display (displays instruction breakpoints which were previously set with the "Breakpoint Set" command)

Example: Breakpoint Set K at 017FBD (sets instruction breakpoint labeled "K" to 01 7FBD).

Example: Breakpoint Clear * (clears all instruction breakpoints)

4.5.2 Display

Display MEMORY FROM { **[hex address]** — **AS DIRECT** ; **RRA (hex add)** — **Base Add** ; **(hex add)(rn)** — **Indexed** ; **rrn (rn)** — **base index** ; **(hex add) (offset)** — **offset** }

{ **[Hex, ASCII, Octal, Binary, Decimal] [Word, Byte]** ; **[Instruction] [Segmented Nonsegmented]**[2] }

The display command offers various addressing modes as well as disassembled mnemonics. The mnemonics match the Z8000 PLZ/ASM standard except for some simplifying (for example, single brackets instead of double for segment numbers).

Example: Display MEMORY FROM 7811F7 DIRECT AS Binary Byte

Example: Display MEMORY FROM 3D15D5 DIRECT AS Decimal Word

Note

The memory displayed as instructions can be scrolled backwards as well as forward. However, the disassembler can not determine multi-word instructions, and errors are possible if the reverse scroll starts the new display in the middle of a multiple-word instruction. In general, to ensure correctness of instruction "SYNC"s, forward scrolling from a known first-word of an instruction is necessary. Scrolling into data areas also disassembles into incorrect instructions.

4.5.3 Addressing Modes

Direct is the most commonly used default mode. Memory display simply begins at the specified address.

The other forms perform an addition of two values. These forms can be useful in scripts or debugging in which the stack, memory in relation to the PC, or some data structure are automatically displayed after a break.

[2]Segmented/nonsegmented option appears only for Z8001 and Z8003.

Base Addr uses a register (pair) to specify the segment and offset of the base address, and adds an optional constant.

Indexed adds an offset contained in a word register to the specified (segmented) hex address.

Base Addr and Indexed are equivalent for the Z8002.

Base Index adds the contents of a register (pair) used as a (segmented) address to the contents of a word register used as an offset.

Offset adds a specified (segmented) address to a specified offset.

4.5.4 Edit

The Edit command options allow the user to edit memory, edit I/O or special I/O, or edit registers. The fields associated with each of these options are as follows:

Edit [Memory] FROM [HexAddr] AS [Hex, ASCII, Binary, Decimal,Octal] [Byte,Word]

Edit [I/o, Special i/o] AS [Hex, ASCII, Binary, Decimal, Octal] [Byte, Word]

Edit [Registers] FROM [0-9, a-f, Pc, fcW, Nsp Refr, pSap] IN HEX

The following gives an example for each of the Edit command options:

Example: "Edit Memory FROM 008000 AS Hex Word

Causes editing to begin at address 8000 of Segment 00. Editing is performed on words that are displayed and entered in hexadecimal.

Example: "Edit I/o AS Hex Byte

Causes I/O Editing mode to be entered. Editing is performed on a byte basis with hexadecimal numbers.

Example: "Edit Register FROM fcW IN HEX"

Prompts the user with FCW old value. The user can then type in a new value in hexadecimal for the Z8000 FCW (Flag Control Word). Entering a new value gives a new prompt of the next register in sequence until "Q" (for quit) is typed.

Example: "Edit Special i/o AS Hex Byte"

Causes Special i/o Editing mode to be entered. Editing is performed on a byte basis with hexadecimal numbers.

Edit Register and Edit Memory begin at the address specified in the command line. The register name or memory address is displayed and the current data in that register or memory location is displayed. At this point, the user can enter a new value for the location. Whenever Byte is selected, an ASCII value can be entered by typing "<CHAR>" (i.e., a quotation mark followed by the desired ASCII character). The value entered is stored when any one of the following keys is depressed: "up arrow", "down arrow", "<SPACE>", "<RETURN>", ".", "@", "Q", "<TAB>", "<CTRL> G", "<CTRL> X", "<CTRL> T". To correct a character, type "<DEL>" or "<RUB>". To restore an entry to the value previously entered, type "<CTRL> U". Entering a "down arrow" or a "<RETURN>" advances to the next location; entering a "." re-edits the same location, allowing the user to check to see if target memory is responding. Entering an "up arrow" goes to the previous location. Entering an "@" allows the user to begin editing "at" a completely new location. The "@" key leaves the cursor at the beginning of the next line where the next location to be edited is typed in, followed by a space, "right arrow", or return. For Edit Register, the "@" is followed by the letter that identifies the register as prompted in the menu line.

Entering a "Q" returns to the Debug command line. "<TAB>" and "<CTRL> T" switch screens or enter Transparent mode as usual, ending the edit session. "<CTRL> G" and "<CTRL> X" start and stop emulation as usual, and also end the Edit session.

In Edit, a series of values can be entered on one line. These values must be separated by spaces, ending with a return or "down arrow." Each value separated by spaces is entered in the next sequential location.

In addition, a looping read can be started by entering "L" after the address is shown. A looping write can be started by entering a value, then an "L". This causes a tight, repetitive read or write of the same location to ease the examination by oscilloscope. The process is ended by a "<CTRL> C".

Edit I/O and Special I/O are like Edit Memory in most respects. There are five major differences:

- The initial I/O address is entered at the beginning of the scroll area of Edit, instead of on the command line.
- Ports are read by typing "R" in the data field instead of entering a value. Ports are not automatically read before writing. Reading a port must be explicitly requested.
- Writing to a port is accomplished by entering a value or series of values followed by <RETURN>. Thus the writing can be performed without a read.
- Typing "<RETURN>" is like typing "@" rather than "down arrow". Most I/O Editing requires random addresses.
- "<SPACE>" does not increment the port address when entering values in the data field. Entering a series of values causes them all to be written to the same port.

4.5.5 Go

Go FROM [hex address]

The Go command is used instead of <CTRL> G when emulation is to start from the same PC each time. <CTRL> G continues from the current PC; "Go" uses the address given. Like <CTRL> G, emulation continues until one of the following conditions occurs: breakpoint, improper memory access, or a <CTRL> C user break.

4.5.6 Host

The host commands are used to transfer loadable binary files on the host to and from target memory space. They can also be used to load and save scripts (see Section 4.7.1 for keystroke sequences) to and from the host.

Host [Load] [Memory,Script] FROM FILE [filename] [Seg No] ["d"]

Seg No and "d" are optional entries for the MCZ hosts to allow the filename to be loaded into a particular segment or data memory since this host does not directly contain segment or memory type information.

Host [Save] [Memory, Script] TO FILE [filename]

Example: Host Load Memory FROM FILE RAM.BIN (load binary file RAM.BIN)

Example: Host Load Script from file B.Script (download script B into EMS from host computer)

Example: Host Save Script to file All.Scripts (upload all scripts to host computer under filename All.Scripts)

Any host load or save can be manually aborted by typing "<BREAK>". If this is done or if a transmission error causes an abort, sync will be lost and the host program will probably have to be cleared by entering Transparent mode and then typing E05<RETURN> until the message "EMS-initiated abort" or "illegal request:E" appears. To be absolutely sure the host program is cleared, the user can type "X <RETURN>" to exit host and "HOST" to reenter.

The **[save] [memory] to file [filename]** prompts for the area to be saved as follows:

After <RETURN>, a *menu* is displayed in the display area that can be moved through with the cursor controls. (Right or left arrow for moving from beginning to end address, and up or down arrows for moving from blocks to block or to the entry point.

Number of blocks: 1 Default 1. Number of discontinuous memory blocks to be saved.
[Hex Add] to [Hex Add] Address range to be saved.
.
.
.
Entry Point [Hex Add]

Type <RETURN> to save 'Q' to Abort.

After <RETURN> is entered, EMS asks one more time "Are you sure?" Any response except "Y" aborts. "Y" starts the save process.

Caution

If the file named already exists, this process writes over the existing file. If no address range is specified, a zero length file results. The user must be careful to not enter extra carriage returns when executing the save (especially in script) to prevent destroying good files. The final query in a memory save will generally prevent problems, but accidental entry is possible when in a hurry or in a repetitive situation.

4.5.7 Memory

Memory [Compare] [HexAddr] WITH [HexAddr] [To,For] [HexWord] AS [Hex / Dec] [Byte / Word]

Memory Compare compares two blocks of memory each with the starting address specified with the length specified. Differences will be displayed on the screen. If the differences fill the screen, the compare will stop but typing the down arrow will scroll the screen and cause the compare to continue.

Memory [Fill] FROM [HexAddr] [For / To] **[HexWord] WITH [values]** [Asc / Hex,Dec] [Byte / Word]

Memory Fill fills a block of memory with a value or string of values. Multiple hex or decimal values are entered separated by spaces. The last four characters will be used in Hex or decimal Word mode and the last two will be used in Hex or decimal Byte mode (e.g., 5E4 will be treated as E4 in byte mode). ASCII strings are treated completely, including spaces. Decimal values will be converted to modulo 64K for word mode, and 256 for byte mode.

Memory [Move] source [HexAddr] destination [HexAddr] [For] [To] [HexWord] [Byte] AS[word]

Example: Memory Compare 001000 WITH 002000 For 0020 AS Hex Byte

Example: Memory Fill FROM 7F43BD to 5000 WITH ABCD EF12 Hex Word

Example: Memory Fill FROM 010200 to IFFF WITH 128 Decimal Byte

Example: Memory Move SOURCE 123456 DESTINATION 345678 For 1000 AS Byte

Memory Move moves a block of memory from a source address to a destination address for the length specified. If the blocks overlap, EMS will preserve the source.

The values in the [HexWord] option fields are inclusive (i.e., "To 1000" includes 1000).

Memory [Search] FROM [Hex Addr] [For] [To] [Hex Word] UNTIL = [= # <= >=] [Value]

Hex Word
Dec Byte

[ASCII]

Memory Search searches a block of memory for a pattern that can be up to 24 characters, including the end delimiters. For example, five words of four hex characters delimited by four spaces will search for the sequence of those five words, or 1234 8D07 5555 AAAA will search for a location in memory of 1234 followed by 8D07, etc. Word mode evaluates inputs on 16 bits and byte mode evaluates inputs on 8 bits for hex and decimal inputs. ASCII values are searched for in bytes with spaces treated as ASCII "20" instead of a delimiter.

4.5.8 Step

Step for [decimal word] instructions [Skip calls]

Example: Step FOR three INSTRUCTIONS

Example: Step FOR two INSTRUCTIONS Skip calls (count the Call itself, but do not count the called subroutine)

4.5.9 Trace

Trace DISPLAY FROM CYCLE [+,-] [decimal word] SNAP [+,-]
[decimal byte] AS {Hex / Instruct} [seg non-seg]

The specification of cycle and snapshot allow the user to choose where in the trace history the display should start. Once displayed, the history may be scrolled back and forth if it is larger than one screen. The selection of a cycle or snapshot earlier than traced will display the earliest cycle traced and likewise, the selection of a more positive cycle number than the last cycle number traced will display the last cycle traced. In the case of the latter, scrolling to earlier history can be accomplished with the up arrow cursor control.

Example: Trace DISPLAY FROM CYCLE +0 SNAP +0 (display the trace memory starting from the trigger cycle of the first snapshot).

Example: Trace DISPLAY FROM CYCLE -32 SNAP +12 (display the trace memory starting from 32 before the trigger point in the 12th snapshot.

Example: Trace DISPLAY FROM CYCLE +7 SNAP -3 (display the trace memory starting with the 7th cycle after the trigger in the 3rd snapshot before the emulation breakpoint.

4.5.10 Watch

Watch [Set] [1, 2, 3, 4, 5, 6] AT [hex address]

Watch [Clear, On, oFf] [1,2,3,4,5,6,*]

Example: Watch Set 3 AT 011000 (monitor address 01 1000 of Z8000)

Example: Watch On (turn on watch area. Scrolling area decreases)

Example: Watch oFf (turn off watch area. Scrolling area increases)

Example: Watch Clear * (clear all watch addresses)

The Watch area is used to display specific registers and memory during debugging. The information in this area is non-scrollable and displays both old and new values for each watch item. The items displayed are: CPU registers, flags, and specified memory locations. The Watch area is shown in Figure 4-1. Old values are those at the beginning of the last emulation. New values are the current values if emulation is stopped, or the values at the beginning of the current emulation otherwise.

```
      R0   R1   R2   R3   R4   R5   R6   R7    Seg NS EP Vi Nvi CZSVDH FCW
New: 0000 0000 0000 0000 0000 0000 0000 0000    0  0  0  0  0   000000 0000
Old: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF    1  1  1  1  1   111111 FFFF
      R8   R9   R10  R11  R12  R13  R14  R15  N14  N15  PSAP    PC     REFR
New: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 000000 000000 W:9E00
Old: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFFFF FFFFFF W:FFFF

           0103FE                 004000
New: 1     2 FFFF 3      4      5 0001 0002 0003 0004 6
Old:         0000                  FFFF A9C7 6214 SDFA
```

Figure 4-1. Watch Display Area

4.6 MAP SCREEN

EMS mappable memory consists of 32 (optionally expandable to 63) 2K blocks that can be mapped to any segment of the Z8001. Up to two segments can be mapped at the same time. In addition, memory protection (over all segments) can be performed with this screen.

After any Map command is executed, the map affected will automatically be displayed. The Display command can be used to display a map when the screen is initially selected or when a different map is to be displayed.

The number of remaining blocks or segments is also shown at the bottom of the display.

MAP [Allocate] [Segment] [Hex Byte] TO [Hex Byte] AS [Target, Nonexistent]

This command restores a range of previously mapped Z8001 segments to Target, or declares them Nonexistent. No mappable memory is allocated with this command. Since only two segments can be mapped (a block at a time), this is a good way of clearing block-level mapping for segments. Map allocating should always be done before Map protecting because any protection previously set for the specified segments will be lost when these segments are reallocated.

Example: MAP Allocate Segment 00 TO 2A AS Target

This command restores segments 00 through 2A to target memory.

MAP [Allocate][Block] [HexAddr] [To,For] [HexWord] AS [EMS] [Target] [Nonexistent] [Copy]

This command allocates a Block of EMS mappable memory from the address specified in HexAddr To the address specified in HexWord (or For the extent specified in HexWord) AS either belonging to Ems (mapped), belonging to the Target (unmapped), or as Nonexistent. If the Copy option is selected, EMS will copy the contents of the target memory into the region of memory mapped to EMS (useful for copying PROM into mappable memory for patching). Only two segments may contain mapped memory, therefore if an attempt is made to map more than two segments an error message is displayed. Only 32 or 63 map blocks are available. When they are all used, an error message will be displayed.

If separate Normal/System or Code/Data is selected on the Configuration screen, there will be three space choices (N,S,-; C,D,-) in the Map Allocate command. The two commands

```
MAP Allocate Block C 000000 To IFFF AS EMS
MAP Allocate Block D 000000 To IFFF AS EMS
```

will map Code and Data to SEPARATE mappable memory. The command

```
MAP Allocate Block - 000000 To IFFF AS EMS
```

will map Code and Data to the SAME mappable memory.

MAP [Display] [All segments]

Displays mapping Blocks allocated previously in a two-dimensional grid. The vertical axis of the grid represents the most significant Hex digit of the Z8001 segment number, and the horizontal axis represents the least-significant hex digit. A "*" in the grid means that this segment has been allocated.

MAP [Display] [Segment] [Hex Byte] or MAP [Display] [S N] [C D] [Segment] [Hex Byte][3]

Displays the mapping of the blocks within Z8001 segment number HexByte. The display shows the address, state of mapping (i.e., USR or EMS), and any protection (Write protect, Normal/System, Code/Data).

[3]System or normal, code or data can be entered if the configuration screen is programmed for separate memory spaces. Segment applies only to Z8001 or Z8003.

Note

For EMS mappable memory, Stack and Data memory can not be separate.

An example of the form used for MAP Display All segments is shown in Figure 4-2.

```
Segment Map:
    <X0><X1><X2><X3>     <X4><X5><X6><X7><X8><X9><XA><XB><XC><XD><XE><XF>
<0X> *   *   T   Tc  Tc  Tc  Tc  Tc  Tc  Tc  Tdw Tdw Tdw Tdw Tdw T
<1X> Tn  Tn  Tn  Ts  Ts  Ts  Ts  T   T   T   T   T   T   T   T   T
<2X> T   T   T   T   T   T   T   T   T   T   T   T   T   T   T   T
<2X> T   T   T   T   T   T   T   T   T   T   T   T   T   T   T   T
<3X> T   T   T   T   T   T   T   T   T   T   T   T   T   T   T   T
<4X> T   T   T   T   T   T   T   T   T   T   T   T   T   T   T   T
<5X> T   T   T   T   T   T   T   T   T   T   T   T   T   T   T   T
<6X> N   N   N   N   N   N   N   N   N   N   N   N   N   N   N   N
<7X> N   N   N   N   N   N   N   N   N   N   N   N   N   N   N   N
```

Legend

Mapping	Protection
N means Nonexistent	c means Code segment
T means Target	d means Data segment
* means Mapped	w means write protect
n means Normal segment	
s means System segment	

Figure 4-2. Segment Map Display

MAP [Clear]

Clears all Block mappings, segment allocations, and protections. All memory is returned to the target system.

MAP [Protect] [Segment] [HexByte1] TO [HexByte2] AS [Write prot] [C,D,N,S]

Applies protection to Z8001 segments begining at segment HexByte1 and ending at segment HexByte2. Protection can apply to Code, Data, Normal, or System accesses. Seven options are available for segment-wide protection. These options are: Normal, System, Code, Data, Write-Prot, Write-Prot Data, and unprotected. Protection may apply to both Target Memory and EMS memory.

MAP [Protect] [Block] [HexAddr] [To,For] [HexWord] AS [Write prot]

[Normal / System / -] [Code / Data / -]

Applies protection for a previously allocated Block. Memory can be designated as Normal only, System only, Data only, Code only, or write protected. These protections can be combined (such as Write protect, Normal, and Code). Protection may apply both to Target memory and EMS memory.

4.7 HELP FACILITY

EMS provides a help facility for those commands that can be applied across many screens. The facility can be invoked by typing "?". In response EMS displays a Help screen listing the global commands as shown below.

```
COMMANDS[4]                                    EXECUTION
  <RETURN> to execute Debug or Map command       <ctl-G> to "go" from current PC
  <up/down arrows> to scroll Debug display       <ctl-C> to "cancel" emulation
                                                 <ctl-X> to single step
SCREENS                                  SCRIPTS
  <TAB><letter> to switch screens          <ctl-R><letter>: record script <letter>
  <ctl-T> to enter transparent mode        <ctl-S>:         stop record script
  <BREAK> to exit transparent mode         <ctl-P><letter>: play script <letter>
  <ctl-Z> to "zero" current screen         <ctl-O><letter>: loop on script <letter>
  <arrow keys> to move on screen           <ctl-W>: pause for input / resume (rec)
                                                    resume after input      (play)
OPTION FIELDS
   <Cap letter> to select option           <ctl-E><letter>: erase script <letter>
   <space>[5] to select next option        <ctl-Q>: kill all macro activity

DATA ENTRY FIELDS
  <ctl-U> to "undo" and restore previous contents while in field
  <RUB> to backspace over errors
  "s" or "h" on prompt line is a hex digit (0-9,A-F) or X for don't care
  "b" on prompt line is a binary digit (0,1) or X for don't care
  If segment number is omitted, previously entered segment remains

Type <RETURN> to continue.
```

Figure 4-3. Help Screen

---

[4] COMMANDS:

<Return> Refers to command lines in the debug and map screen. To execute the command after the option fields have been programmed, enter <RETURN>

<up, down arrows> For scrolling memory display or trace display.

[5] <space> can be used in an option field to scroll through the options in lieu of direct entry of the option letter. Use of the space bar for option scrolling leaves the cursor in the same option field for further scrolling while direct entry sometimes steps the cursor to the next option field (never on Allocation and Configuration, sometimes on Program, and always on Debug and Map.

4.7.1 Scripts (EMS monitor macros)

The following lists the procedures and requirements for the Script usage:

1. Start recording a script with "<CTRL> R [A-Z]". All keystrokes are recorded in the EMS monitor scratch memory. These user keystrokes usually represent a particular EMS setup for repetitive debugging sessions. Up to 26 scripts can be recorded. These scripts can be named with a single capital letter [A-Z] (starting a macro with the "<TAB>" character is a good idea because it always guarantees that the cursor will be placed in the first option field of the Change screen).

2. Script recording can be stopped by typing "<CTRL> S".

3. Play a previously recorded script by typing "<CTRL> P [A-Z]". The script named by a single capital letter [A-Z] is played back from EMS monitor memory.

Note

EMS must be in generally the same configuration (especially the location of the cursor) as when the script record process was initiated to reproduce the same results.

4. A script can be erased (cleared) by typing "<CTRL> E [A-Z]". For example, to erase script B, type "<CTRL> E B", or all can be cleared by typing "<CTRL> E".

5. Scripts can be aborted by typing "<CTRL> Q". The recording process should not be aborted by <CTRL> Q because this leaves the command sequence in an unclear state and can cause EMS to "crash."

6. Scripts can be "looped" automatically by typing "<CTRL> O [A-Z]".

7. If user input is required in the middle of a script playback, follow the following sequence during script recording:

   - Type "<CTRL> R B" to begin recording script "B".
   - Type "<CTRL> W" to wait for input.
   - Type INPUT--input at this time will not be recorded.
   - Type "<CTRL> W" to resume recording process and type rest of script.
   - Type "<CTRL> S" to stop recording.

To play back this script:

Position the cursor to the initial location where the recording process began. To prematurely abort the script playback, type "<CTRL> Q".

Type "<CTRL> P B" to play script "B".

Type the USER INPUT when the macro has paused for you; the terminal will prompt with a <CTRL> G or a beep.

Type "<CTRL> W" to continue with the macro.

A

Zilog

APPENDIX A

EMS 8000 Tutorial

A.1 - INITIALIZATION

This tutorial is intended to allow the first-time user to step through some of the beginning steps of using the EMS and thus to speed the familiarization process. Keystrokes are shown exactly as they are to be entered by the user, with the following exceptions:

- The control key (CTRL) that is used in conjunction with another key is shown as <CTRL> (letter).

  For example: <CTRL> G indicates that the control key <CTRL> is pressed simultaneously with the letter G.

- The tab key is shown as <TAB>.

- The space bar is shown as <SPACE>.

- The return key is shown as <RETURN>.

- The cursor control keys that are used to move between option fields are as follows:

  The right arrow is shown as <r arrow>.

  The left arrow is shown as <l arrow>.>

  The up arrow is shown as <u arrow>.

  The down arrow is shown as <d arrow>.

The EMS 8000 is used in the link between the user's host computer and terminal. When not in use, the EMS can be set to Transparent mode, which allows the host to be used normally without disconnecting the EMS. The exception to this is if the user's normal system requires the use of the BREAK key; note that EMS uses BREAK to terminate from the Transparent mode.

The user is assumed to have read through the remainder of this manual, and to be familiar with the terms used.

The EMS system should be assembled according to the installation procedure in this manual. The following items should be checked:

o Terminal connection to the terminal port on the back of the EMS via the RS-232 cable. The terminal always operates at 9600 baud.

o The host connection to the host RS-232 connector and an appropriate baud rate set on the EMS host baud rate switch. This does not have to be the same as the terminal. A table of settings is shown on the back panel of the EMS with switch position 5 being the most-significant bit and

- The Emulator pod connection to the EMS unit by the two emulator cables.

WARNING

Incorrection insertion of these cables will damage the unit.

- Make sure that the target cable is correctly inserted into a target system with pin 1 of the 40/48 pin connector corresponding to the pin 1 of the Z8002 and Z8001 socket. The EMS unit requires an external Z8000-type clock; if no target is available and the user wants to use the EMS for familiarization or software testing, the test socket on the front panel of the EMS can be used to supply the emulator pod with a clock. To use this socket to supply the clock signal, the test board inserted behind the front panel and to the left of the socket must be oriented for the correct CPU (40- or 48-pin) and the emulator cable must be inserted into the socket. When using the 48-pin Z8001, the connector is difficult to place into the ZIF socket and care should be exercised. The lever should be held slightly away from the connector as it is being inserted into the socket and then pressed fully vertical to lock the connector in place.

The EMS should be powered up and initialized. (See Section 2.6, Booting the System Up). In general, the order of powering up each component is not important, but in some cases if a full 25-pin cable is used, ground loops can cause a reset of the host or EMS or a failure to reset in either. In such cases, a three-wire cable using only signal ground, transmitting data, and receiver data lines should be used.

After completion of the session, the host link can be terminated by entering Transparent mode, typing "X" and then "<RETURN>". If the HOST program is in the middle of a load or download procedure, it responds with data lines and/or error messages. Several E05<RETURN>s clear the HOST and allow it to accept the X<RETURN>.

A.2 FAMILIARIZATION

After the EMS software is loaded, the EMS takes a few moments to initialize, then the terminal screen changes to the Change screen with the mode set to the Allocation mode. At this point, any of the EMS screens can be entered by entering the first letter of the screen name. The Change screen is shown below.

```
NEW SCREEN:  Allocation

                              Type "?" for help.


Allocation Configuration Debug Pattern Map
```

As the Change screen is shown, there is only one option field (the top position now showing Allocation) and the cursor is positioned at the beginning of that field. The options that can be entered in any option field are shown at the bottom line of the screen, as it is now showing each of the five action screens of Allocation, Configuration, Debug, Pattern, and Map. Thus to enter any of the screens, type the letter shown as a capital (i.e., A, C, D, P, or M).

A.3 TUTORIAL

The following tutorial provides a sequence of commands that guides one through a sample emulation using the EMS to debug a simple program. This tutorial is intended as a learning tool, particularly for first-time users of the EMS 8000.

In the following sequence of keyboard entries, a carriage return, (i.e., <RETURN>), is not entered unless specified, and blanks and new lines are used for clarity of reading only. Addresses containing segment numbers refer to the segmented CPU and can be shortened to just the offset portion for use with the nonsegmented CPU.) Incorrect entry of data can be corrected by entering "<CTRL U>" or by using <delete> and reentering the data. Where entries are different for users of Z8002, the Z8002 command are entered in parentheses after the Z8001, Z8003 commands.

Enter these keystrokes:	
A <TAB> C <TAB> D <TAB> P <TAB> M <TAB>	Each letter results in the screen changing to that particular screen, then exiting back to the Change screen. This is always the procedure for the selection of the desired menu.
?	Typing "?" enters the Help screen, which lists the control characters that are available for some key EMS functions.
<RETURN> <CTRL R> A	This initializes a recording process that records all the keystrokes entered as script until it is terminated. This script can be the host for retrieval and reuse later. The "<CTRL> R" starts a script called A. The bottom right corner reflects a message, "Rec", to indicate that recording is in progress.
A	Used to display the Allocation screen. The screen should now default to allocation setup. (This is a common setup that gives EMS a good basis for debugging average problems.) Changes to this screen are discussed later in this tutorial.
<CTRL Z> D	If the user has changed the Allocation setup and wishes to reset it to the default quickly or wishes to zero it totally and assign a new allocation, the <CTRL Z> performs this. A following D sets the default and 0 clears all allocations. Since the EMS has just been initialized, this setup is redundant.
<TAB> M	To reach Map screen. Cursor is at the first option screen with the options shown at the bottom of the screen. (Allocate Clear Display Protect)
A B 000000 (0000 for Z8002) T 0FFF E - <RETURN>	Allocate by Block address <00>0000 to 0FFF into EMS mappable memory without Copy. The copy option copies existing external memory into the mapped memory. This option is generally used when debugging read only memory (PROMS). As the cursor steps from field to field, be sure to look at the bottom line of the screen for the menu list. The <RETURN> causes execution of the command line just generated and displays the resulting map.

<right arrow> <right arrow> 1000<right arrow> F 1000 N <RETURN>	Alternately, if most of the command line is already correct, the cursor control keys can be used to step to a specific field for correction. The first 1000 resets the beginning address to 1000 with the segment (Z8001 only) unchanged. The F changes the allocation to a byte counting mode, and the second 1000 is the number of bytes. N sets the allocation to non-existent, which allows breaking by EMS if specified in the Configuration screen and prevents EMS from writing to it during Emulation mode. A <RETURN> anywhere in the command line causes execution of the command as it then exists.
<TAB> D	Goes to the Debug screen. The Debug screen consists of various command lines. When the Debug screen is entered, the cursor is placed in the command option field and the commands are listed at the bottom of the screen. These commands are used in the actual debugging process.
D <RETURN> <RETURN>	Display memory from 0. Return from display to command line.
M F 0 <r arrow> T OFFF 8D07 <r arrow> H W <RETURN>	Memory Fill from 0 to OFFF with 8D07 as Hex words.
D <RETURN> <RETURN>	Display memory filled with 8007.
E R P <RETURN> 000008 <RETURN> C000 (4000 for Z8002)	Edit Registers FROM Pc IN HEX. Load PC with a starting value of 8. Load FCW with System Segmented mode, (System Non-segmented for Z8002, although C000 could also be used with the Z8002 because it ignores the segmented/nonsegmented bit).
Q	Quit Edit.

	(For Z8002)	
E M 000000 H W	(E M 0000 H W)	Edit Memory FROM 0 AS Hex Words
<RETURN>		
[Z8001]	([Z8002])	(Entries not excepted with ( ) for Z8002 are common to both microprocessors.)
0 <RETURN>		Set Memory with initializing values of FCW and
0000 <RETURN>	(4000 <RETURN>)	PC in case of reset.
0008 <RETURN>		(Z8001 treats this as segment 00XX and the Z8002 treats it as location 0008.)
8 <RETURN>	(<RETURN>)	Z8002 does not use this location because the first instruction is at location 8.
1402	(<RETURN> (2102 <RETURN>)	Edit reads the last four entries on a line.
8000	(RETURN> (500 <RETURN>)	Try entering 2041402 <RETURN> instead of 1402
500 <RETURN>	(2104 <RETURN>)	<RETURN>.
1404 <RETURN>	(600 <RETURN>)	
8000 <RETURN>	(<RETURN>)	
600 <RETURN>	(<RETURN>)	
2922 <RETURN>	(2922 <RETURN>)	
2B40 <RETURN>	(2B40 <RETURN>)	
@		The "@" specifies: go to the next value as an address and continue editing there.
E8F0 .		The " . " (period) specifies: redisplay the previously edited location. In this case location 32 should be redisplayed with the contents changed to E8F0.
@		
60 <RETURN>		
E800<SPACE>E800<SPACE>E800		When editing memory locations, numbers entered in sequence and separated by a space are
<RETURN>		loaded into consecutive memory locations. This allows the user to enter a string of numbers instead of entering one line at a time. When editing I/O, all values on one line are written into the same address to allow multiple entries to the same port.
Q		Quits edit session.
D000008	(D 0008)	
D I S	(D I)	Display memory from location <00>0008 in
<RETURN>		Direct addressing as Instruction in Segmented mode (nonsegmented for Z8002). Execute the display command.

	At this point the disassembled version of the program should be displayed.
<d arrow> <d arrow>	Scroll the display down twice. The last line of the entry should now appear. The program consists of two immediate loads into register pair 2 and 4, INC @RR2, DEC @RR4, followed by a series of NOPs, and then a relative jump back to the INC step. (The Z8002 will have some extra NOPs).
<RETURN>	Terminate the display command.
<TAB> P	Go to pattern screen.
<d arrow> <RETURN> <r arrow>	These are alternate ways of traveling around the screen from field to field.
* = 000014 <r arrow>	Activate first line of trace trigger and set address equal to <00> 0014. Notice that the Z8002 ignores the first two 00s, which are the segment number for the Z8001.
<RETURN> <RETURN> <RETURN> <RETURN> <RETURN>	Step the cursor down to Snapshot Trigger set.
S <r arrow> 10 <r arrow> B <RETURN>	Set snapshot to start 10 cycles before the trigger.
<TAB> D	Return to Debug screen.
G 000008 (G 0008)	Set Go to start from <00> 0008. Note that the Data field is exited when it is full. Therefore, to correct errors, use the <RUB> key or exit the Data field and reenter. Only the maximum number of digits or less should be entered (e.g., G 8 <RETURN> could also be entered).
<RETURN>	Execute the Go command.
	At this point, the screen should flash a "Running" message momentarily in reverse video. A bell will then sound and the message will change to "Trig Break". To the right of this message will be another reverse video message, "Rec", indicating that you are still recording script.
T <RETURN>	Display trace history. The trace will be displayed from snapshot 0 cycle 0.

<u arrow> <u arrow>	Scroll the trace display up to show previous cycles. Keep entering up arrows until the scrolling stops.
<RETURN>	Terminate the trace display.
T - 1024	
<r arrow>	
<r arrow>	Display trace history again from the earliest history retained as Instructions Segmented (Nonsegmented for Z8002).
<r arrow>	
I S (I for Z8002)	
<RETURN>	
<d arrow> <d arrow>	Scroll the trace down to familiarize yourself with the display.
<RETURN>	Terminate the trace display.
	Now there is a trace showing the operation of a simple test program. The trace shows all instructions and steps in the test program including many NOPs. In debugging a program with unimportant steps, the EMS allows more efficient use of the trace memory size by qualifying the trace to skip the unimportant steps.
<CTRL S>	Stop recording of the script.
<TAB>	
<CTRL P> A	Now watch script A play, and repeat the process to here. Remeber that, in general, script must have the same starting conditions to guarantee the same results or care must be taken to have direct entry in option fields since the space bar causes a toggling to the next option.
<TAB> D	Returns to the Debug screen.
W S 5 500 <RETURN>	Set Watch area 5 at location <0>0500.
W S 6 600 <RETURN>	Set Watch area 6 at location <0>0600. Watch areas 5 and 6 give an automatic peek at four consecutive locations upon break from emulation, whereas Watch areas 1 through 4 let you observe one for each of these watch lines. The watch is useful for keeping track of important or troublesome locations.
<CTRL G>	Perform another emulation run which should also end in a trigger break. This method of filling snapshots and using the snapshot to break from emulation allows the user to adjust where in relation to the trigger, trace is recorded.
	Notice the Watch area and how the before and after values of memory locations are shown.

W F <RETURN>	Turn Watch area Off to allow more display room on the screen.
T <RETURN> <RETURN>	Display Trace history again as instructions.
W O <RETURN>	Turn Watch area back on so that register values can be monitored.
<TAB> P	Return to Pattern Screen.
* <r arrow> # 8D07 <r arrow> IF1	Enable the trace qualifier as data not equal to 8D07 on an IF1, which is the NOP instruction.
<CTRL G>	Start the emulation again; there should again be a trigger break.
<TAB> D	Return to the Debug screen.
T <RETURN>	Display the trace and notice that the NOP instructions were left out and that only IF1 cycles were trace. Without IFN cycles, most instructions cannot be disassembled.
<TAB>	Terminate the trace display. If exiting the Debug screen, it is not necessary to terminate the trace display first. <TAB> will exit the screen directly.
P <r arrow> <r arrow>	Enter the Pattern screen.
- (minus sign)	Disable the data field.
<r arrow> 1000	Set the cycle type to data. 1000 is the status number corresponding to a data cycle. (Alternately, D or DATA could have been entered.) When using cycle names, only enough of the name is required for unique identification. Thus D is sufficient for data but IF1 must be fully specified. The cycle status bit can also be set to "don't care". Thus, 1XXX would select any status with the first status bit a 1.
<TAB> D	Return to the Debug screen. Notice the command line is set to trace because that is what had been set up when the Debug screen was last exited.
<CTRL> G	Start the emulation again; there should again be a trigger break.

<RETURN>	Execute the trace command. Notice that the trace contains only the data cycles and trace triggers.
<TAB> P	Return to the Pattern screen.
- - <RETURN>	Turn off the trace qualifier.
4 <RETURN>	Set the pass counter to 4. This means that the trigger must be recognized four times before the event occurs.
<RETURN> <RETURN> <RETURN> <RETURN> <RETURN> <RETURN>	Move the cursor down to the Snapshot field.
<r arrow> 0 <RETURN>	Set the snapshot trigger to be at the trace trigger. The ten cycles were set to ensure that the trace would be terminated soon enough to preserve the beginning of the trace. Otherwise, the beginning of the trace would be lost due to being overwritten by multiple-cycle instructions occurring as the emulation was ending.
255 <RETURN> I	Set the break to occur after recording 255 snapshots and set the number of snapshots to 256. This will partition the trace into 256 separate trace memories.
<CTRL G>	Start emulation.
<TAB> D	Return to the Debug screen.
<RETURN>	Display trace. Notice that four cycles (the snapshot size) have been traced every fourth time through the loop (the pass count number). The dashed lines indicate that not all cycles are shown. If the start of a following snapshot is consecutive with the previous snapshot and there are no missing cycles, just a blank line is shown. Scan the trace by using the up and down cursor control arrows. The snapshot number is in relation to the first trigger, those occurring after are plus and those before minus, and the cycles are in relation to the last trigger in that snapshot. (Therefore, the last snapshot in a sequence which may have two or more triggers will have the cycles numbered from the last one.) As you become familiar with this display, try exiting the trace display command, modifying the starting snapshot and cycle numbers, and then reexecuting the trace display command (by hitting <RETURN>).

<TAB> P	Return to the Pattern screen.
<u arrow>	Go to the bottom of the screen by a "short cut."
H	Change number of snapshots to 128.
<u arrow> <u arrow>	
4 <r arrow> B	Start snapshot 4 cycles before trigger. Note that the number of cycles must not be greater than the number of cycles in the snapshot because the control is always after an event. Specifying "Start SNAPSHOT 4 CYCLE - BEFORE TRIGGER" allows the machine to perform arithmetic for you and then causes a break after the remaining portion of the snapshot is filled after the trigger. For example, if the snapshot is 8 cycles long, specifying that the snapshot should be started 4 cycles before the trigger actually causes a break 4 cycles after the trigger by subtracting the 4 cycles before from the length of 8.
	The break before is helpful because the trace continues until the completion of the last instruction. In the case of block move instructions, the EMS interrupts a few cycles into the intruction to preserve most of the trace.
	This configuration allows you to record the trigger instruction as a part of the recorded trace.
<CTRL G>	Start emulation.
<TAB> D <RETURN>	Return to Debug screen and display trace. Now the trigger is recorded and you can see instructions occurring around it.
<TAB> P	
<u arrow> D	Return to Pattern screen and set snapshot size to 8 snapshots of 128 cycles.
<CTRL G>	Start emulation.
<TAB> D <RETURN>	Display trace. Notice that there is now a full trace with the snapshots being the last 8 of 255 snapshots taken. Each of the snapshots terminates at the next trigger even though the snapshot was not yet full (the trigger is set to be every fourth execution of location 14). The last snapshot, however, continues to a full snapshot length and contains two triggers. Notice that the cycles are numbered from the last trigger in the last snapshot.

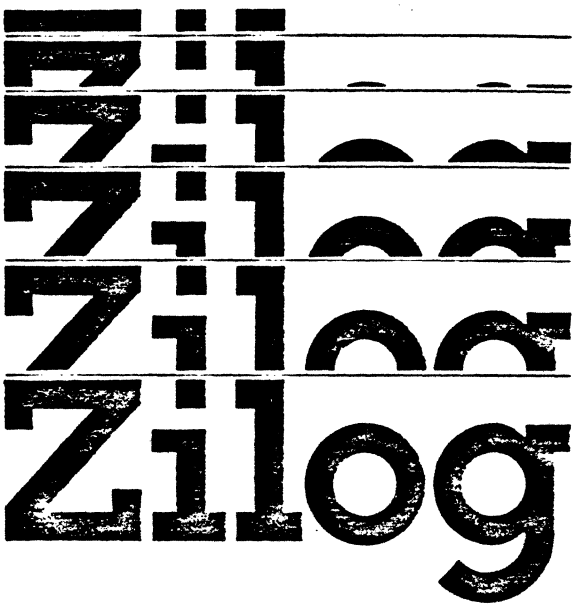
This concludes the use of this program, which has demonstrated some simple debugging methods. The rest of this section serves to familiarize you with the editing commands in the Debug screen.

A.4 EDITING COMMANDS IN DEBUG SCREEN

<RETURN>	Exit the trace display command from above if not already done.
E	Go to the edit command line. This command has already been used to load the simple tutorial program and to set the PC and FCW to the desired value.
R <RETURN>	Edit registers (the starting point is preserved from the last time the registers were edited).
60 .	Set PC to 60 and display the change.
@	Override the normal edit sequence so that instead of prompting for a change to the FCW, the EMS will prompt for a register name to be edited.
0	Zero for register 0.
1234 (u arrow>	Enter value 1234 and go the previous register in sequence.
0000 <SPACE>	Notice that in the case of registers, <SPACE> acts like a <RETURN>.
Q	Quit the edit mode.
E I	Edit I/o.
<RETURN>	Execute the command.
0021 <RETURN> R	Enter the address of the I/O port to be read and read it. Unless the target the EMS is presently executing in has I/O at location 0021, what is read will be nothing, generally reflecting the address due to the open bus.
<RETURN>	Set the mode to read the next port address.
ff12 <RETURN> R 44 <RETURN>	Read port ff12 and then write value 44 to it.
ff22 <RETURN> 12 <SPACE> 23 <SPACE> 33 <RETURN>	Write values 12, 23, 33 into port ff22 without a read cycle.
Q	Used to quit the Edit command.

In summary, the edit command has the subcommands "<RETURN>", ".", "<up arrow>", <down arrow>", "<space>", "Q", and "@" to allow the user to quickly enter data in an efficient manner. The user should remember small differences between each type of edit [whether it is memory, registers, or I/O (normal or special)]; commands apply to each.

B



APPENDIX B

LOAD/SEND PROTOCOL

B.1 INTRODUCTION

This appendix describes the protocol used for transfer of program (binary) text between EMS 8000 and its host computer. Transfers take place over an asynchronous serial line at a variety of speeds. Specific design goals for this protocol were:

- To be compatible with (at least) all development systems manufactured by Zilog for the Z8000.
- To be compatible with other systems (PDP-11, DEC-10, VAX, DG Nova, etc.)--highly desirable.
- To provide a form of error checking, which gives a reasonable assurance that the data sent is received correctly.
- To provide a method of error correction for data incorrectly received.
- To ensure that the download/upload process would not require an exorbitant amount of time for a reasonably-sized program.
- To provide the following features:
  - Support for the full addressing capability of the Z8000, including code and data separation.
  - Transfer abort can be initiated from either the sending or receiving station.
  - A method must be provided for transmission of error indications.

The Load/Send protocol that was developed to meet these objectives transmits the unencoded records in 8-bit binary. This procedure requires that the host computer be capable of transmitting or receiving 8-bit characters through the terminal interface. Also, the following software options must be disabled if present:

- Automatic carriage return at right margin of terminal.
- Conversion of tabs to spaces.
- Conversion of form feeds to line feeds.
- Printing of control characters as "char" or other control character translations.
- Non-transmission of certain characters (e.g., the DEC RT-11 and RSX-11M terminal drivers do not output null characters).

- Use of characters such as DC1 and DC3 (CTRL-S and CTRL-Q) to start and stop output.
- Use of the 8th bit for parity.
- Characters that mean "take the next character literally" (like backslash).

Both the MCZ and S8000 feature an "absolute" or "raw" mode that allows the binary protocol to be used with them. This protocol yields a line efficiency of 92%, which results in much faster downloads than the Tek Hex protocol.

B.1.1 Records

All transmitted data takes the form of single-line records that appear to the host computer as if they were entered by an (extremely fast) operator. The first character of each record is used to determine the type of the record:

Table B-1. Record Types and Associated Characters

Character	Record Type
A	Acknowledge Record
D	Data Record
I	Instruction Record
E	Error Message Record
L	Load (Host --> EMS) Request
S	Save (EMS --> Host) Request
T	Termination Record
*	Configuration Record

B.1.2 Load/Save Operation

The following sequence of events takes place for a load or send operation. First, EMS sends an "L" record (for download, host to EMS) or an "S" record (for upload, EMS to host) that specifies the filename of the file to be transferred. Then the host responds with a '*' (configuration) record specifying that 8-bit binary format is used for the data transfer. If a problem occurs opening the file, the host responds with an error record instead.

After this initial exchange, the host and EMS perform a dialogue of actually transferring data. The sender (EMS for upload, the host for download) sends a 'D' (data) or 'I' (instruction) record. After receiving it and checking the checksum, the receiver sends an 'A' (acknowledge) record if there is no error. If there is an error, the receiver sends an "E" (Error) record, then the sender "re-sends" the data record; after ten retries, the sender sends an abort error instead of the data or instruction record.

When all data is transferred successfully, the sender sends a "T" record, which contains an entry point and signals the end of the file. The receiver checks the checksums and issues an "A" record.

For downloading only, a second acknowledge record is sent by the host after EMS's acknowledge of the termination record. This prevents EMS from making another download request until the host has finished closing the file and is ready for a new command. This is particularly important for hosts without "typeahead" during the initial program load, which contains several files.

B.1.3 Load and Save Record Format

The Load and Save records have the following ASCII format:

```
Lfilename<RETURN>   (download a file)
Sfilename<RETURN>   (upload a program file)
Pfilename<RETURN>   (upload an ASCII file)
```

where "filename" is an ASCII string giving the name of the file to be read or written. This field is inherently system-dependent, so the EMS end of the link will do no error checking for file names. All information typed in the filename field in the Host Load or Save command is supplied. This may include extra parameters after the filename. For instance, the MCZ version of HOST uses a token after the filename (if supplied) as the Z8001 segment number, and to convey code/data information.

If a "P" request is used instead of an "S" request to save a file, that file should be opened as an ASCII file, and all address information sent with the file should be ignored. This feature will be used by the Host Save Script command to save the keystroke information so that the Host's editor may be used to edit the command script.

B.1.4 Configuration Record Format

The format of the configuration record is:

```
*BB<RETURN>
```

where * is the ASCII character '*' (hex 2A), and 'BB' specifies that both sending and receiving is done with an 8-bit binary format. Configuration records are issued in response to a 'L' or duration of the transfer. No formats other than 8-bit binary are currently supported.

B.1.5 Instruction/Data Record Format

Instruction and Data records contain the actual data transferred in a load or send operation. The record is in the following form:

```
<ID><text>
```

where <ID> is the ASCII character "D" (hex 44) or "I" (hex 49) which identifies the record as a data or instruction record (respectively); <text> contains address, count, and checksum information, and the data.

The <text> is always in this format:

Offset	Contents
0	Segment number
1	Offset low byte
2	Offset high byte
3	Byte count n
4	Checksum low byte
5	Checksum high byte
6	Data byte 0
	. . .
n+5	Data byte (n-1)

The segment number and address field define the Z8000 address at which "Data byte 0" will be loaded. The byte count is the number of bytes in the Data portion of the record. "Checksum" is the sum of the bytes in the binary record (not including the checksum field). Note that the bytes in the offset and checksum fields are low byte followed by high byte. Up to 128 data bytes may be sent in a single record. An acknowledge is sent immediately after the required number of characters are transmitted. No termination character, such as carriage return, is used in this type of record.

B.1.6 Acknowledge Record Format

The Acknowledge record is sent in response to a correctly received data or instruction record. Its form is simple:

A<text><RETURN>

where <text> may be zero or more arbitrary characters. The <text> field currently has no use, but may be used in the future for double buffered transmission. EMS will always send one byte of <text>.

After receiving an acknowledge record, the next data record or a termination record can be sent.

B.1.7 Termination Record Format

The Termination record indicates the end of transmission, supplies the program entry point, and contains a vertical checksum to ensure that entire data records were not missed. Its form is similar to a Data record:

T<text>

where T is the ASCII character "T" (hex 54) which identifies the record as a termination record; <text> contains entry point address and vertical checksum information.

The <text> is always in this format:

Offset	Contents	
0	Segment number	
1	Offset low byte	Entry Point
2	Offset high byte	
3	2 (length of v. cksum)	
4	Hor. Cksum low byte	
5	Hor. Cksum high byte	
6	Ver. Cksum low byte	
7	Ver. Cksum high byte	

The segment number and address field define the Z8000 address with which the PC will be loaded. The byte count is 2--the number of bytes of vertical checksum. "Hor. Cksum" is the sum of the entry point address bytes and the vertical checksum bytes. "Ver. Cksum" is the sum of the checksum fields of all of the 'D' and 'I' records sent during the current operation. Checksums in records re-transmitted due to error are not included.

B.1.8 Error Record Format

The Error record allows the Host or EMS to signal that some error has occurred. Any error sent to EMS by the Host (except checksum error) will abort the current operation, and will be printed on the terminal. EMS will only send four types of error records: horizontal checksum error, vertical checksum error, Host-initiated abort, and EMS-initiated abort. Upon receiving a horizontal checksum error, the Host should retry the most recent data, instruction, or termination record; upon receiving any of the others, it should abort the current operation.

The form of the error record is:

E<error number><text><RETURN>

where E is the ASCII character "E" (hex 45) which identifies the record as an error record; <error number> is a two-digit decimal ASCII number which identifies the type of error; and <text> (sent only from the Host to EMS) is the message to be printed on the terminal when the error occurs.

These <error number>s are defined in Table B-2.

Table B-2. Error Numbers and Meaning

Number	Meaning
01	Horizontal Checksum Error (initiates retry)
02	Vertical Checksum Error
04	Host-initiated abort
05	EMS-initiated abort

c

APPENDIX C

EMS 8000 EMULATOR TIMING ANALYSIS

While the EMS 8000 emulator is specified to emulate at a clock speed of 6 MHz, it uses a Z8000B CPU (10 MHz) in the user pod in order to compensate to some extent for the buffering required between the CPU and the target.

The 10 MHz CPU and the EMS control functions cause some differences in the expected timing specification. In general, these differences are slight and should not cause a problem unless the target is very close to the limits of the design specifications.

To aid the designer in handling such problems, the following table is a summary of worse-case timing. The table shows 6 MHz and 10 MHz specifications with specifications for a 10 MHz CPU running with a 6 MHz clock and the EMS 8000 running with a 6 MHz clock. The last column lists differences between a 6 MHz CPU and the EMS with a + signifying EMS is better and - signifying EMS is worse.

The definition of parameters is in the Z8000 CPU manual.

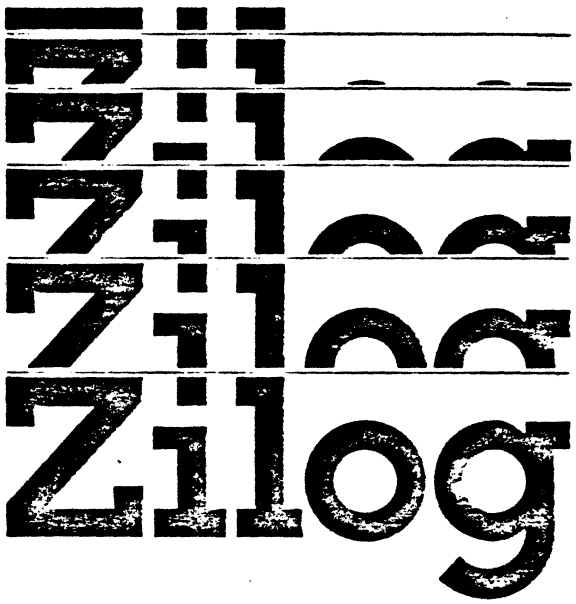
Table C-1. EMS 8000 Emulator Timing Analysis

	10 MHz Z8000		10 MHz Z8000 min	10 MHz Z8000 max	6 MHz EMS 8000 min	6 MHz EMS 8000 max	6 MHz Z8000 min	6 MHz Z8000 max	DELTA + = better; - = worse
1	Clock Cycle Time		100	2000			165	2000	
2	Width High		40				70		
3	Width Low		40				70		
4	Fall Time			10				10	
5	Rise Time			10				15	
6	Clock	SN +		70		88		110	+22
7	CK	SN min +	5		11		10		+1
8	CK	Bus Float		40		95		55	-40
9	CK	Addr Valid		50		80		75	-5
10	CK	Addr Float		40		81		55	-26
11	AD -	AD-		180	316		305		+11
12	AD +	CK	10		22		20		-2
13	DS	Addr Active	20		60		45		+15
14	CK	AD+		50		62		75	+13
15	AD min +	DS min + -	0		11		0		-11
16	AD -	DS in +	110		200		195		+5
17	AD -	MREQ min +	20		45		35		+10
18	CK	MREQ +		40		61		70	+9
19	MREQ max -	MREQ min +	80		131		135		-4
20	MREQ max -	Addr Float min +	20		33		35		-2
21	AD -	DS min +	15		40		35		+5
22	MREQ -	AD -	140		237		230		+7
23	CK	MREQ +		45		66		60	-6
24	CK	AS +		40		54		60	+6
25	AD -	AS min +	20		43		35		+8
26	CK	AS +		40		54		80	+26
27	AS -	AD -	140		244		220		+24
28	DS -	AS min +	15		29		35		-6
29	AS -	AS min +	30		51		55		-4
30	AS	Addr Float	20		-40		45		-5
31	Addr Float	DS	0		-40		0		-40
32	AS -	DS min +	30		53		55		-2
33	DS -	AD -	70		132		130		+2
34	CK	DS +		45		66		65	-1
35	DS -	AD min +	25		38		45		-7

Table C-1. EMS 8000 Emulator Timing Analysis (Continued)

	10 MHz Z8000		10 MHz Z8000 min	10 MHz Z8000 max	6 MHz EMS 8000 min	6 MHz EMS 8000 max	6 MHz Z8000 min	6 MHz Z8000 max	DELTA + = better; - = worse
36	AD -	DS min +	65		125		110		+15
37	CK	DS +		60		81		85	+4
38	DS -	DS min +	110		191		185		+6
39	CK	DS +		60		81		80	-1
40	DS -	DS min +	75		131		110		+21
41	DS -	AD -	120		217		210		+7
42	CK	DS +		60		81		90	+9
43	DS -	DS min +	160		276		255		+21
44	AS -	DS min +	410		693		690		+3
45	CK	DS +		65		86		85	-1
46	DS -	AD -	165		292		295		-3
47	CK	ST +		60		86		85	-1
48	ST -	AS min +	10		19		30		-11
49	Reset +	CK	50		80		70		-10
50	Reset min -	CK	0		-10		0		+10
51	NMI +	NMI min -	50		70		70		0
52	NMI +	CK	50		80		70		-10
53	(N)VI +	CK	40		62		50		-12
54	(N)VI min -	CK	10		3		20		+17
55	SEGT +	CK	40		62		55		-7
56	SEGT min -	CK	0		-7		0		+7
57	MI	CK	80		98		110		+12
58	MI	CK	0		-6		0		+6
59	CK	MO +		70		88		85	-3
60	STOP +	CK	50		72		80		+8
61	STOP min -	CK	0		-7		0		+7
62	WAIT +	CK	20		32		30		-2
63	WAIT min -	CK	5		1		10		+9
64	BUSRQ +	CK	60		105		80		-25
65	BUSRQ min -	CK	5		-10		10		+20
66	CK	BUSAK +		60		78		75	-3
67	CK	BUSAK +		60		78		75	-3
68	AD -	AD min +	50		107		95		+12
69	DS -	ST min +	30		48		55		-7

I



EMS 8000
INDEX

-F-

-G-

-H-

-U-

-V-

-W-

-Z-

Zilog Sales Offices

West

Sales & Technical Center
Zilog, Incorporated
1315 Dell Avenue
Campbell, CA 95008
Phone: (408) 370-8120
TWX: 910-338-7621

Sales & Technical Center
Zilog, Incorporated
18023 Sky Park Circle
Suite J
Irvine, CA 92714
Phone: (714) 549-2891
TWX: 910-595-2803

Sales & Technical Center
Zilog, Incorporated
15643 Sherman Way
Suite 430
Van Nuys, CA 91406
Phone: (213) 989-7485
TWX: 910-495-1765

Sales & Technical Center
Zilog, Incorporated
1750 112th Ave. N.E.
Suite D161
Bellevue, WA 98004
Phone: (206) 454-5597

Midwest

Sales & Technical Center
Zilog, Incorporated
951 North Plum Grove Road
Suite F
Schaumburg, IL 60195
Phone: (312) 885-8080
TWX: 910-291-1064

Sales & Technical Center
Zilog, Incorporated
28349 Chagrin Blvd.
Suite 109
Woodmere, OH 44122
Phone: (216) 831-7040
FAX: 216-831-2957

South

Sales & Technical Center
Zilog, Incorporated
4851 Keller Springs Road,
Suite 211
Dallas, TX 75248
Phone: (214) 931-9090
TWX: 910-860-5850

Zilog, Incorporated
7113 Burnet Rd.
Suite 207
Austin, TX 78757
Phone: (512) 453-3216

East

Sales & Technical Center
Zilog, Incorporated
Corporate Place
99 South Bedford St.
Burlington, MA 01803
Phone: (617) 273-4222
TWX: 710-332-1726

Sales & Technical Center
Zilog, Incorporated
240 Cedar Knolls Rd.
Cedar Knolls, NJ 07927
Phone: (201) 540-1671

Technical Center
Zilog, Incorporated
3300 Buckeye Rd.
Suite 401
Atlanta, GA 30341
Phone: (404) 451-8425

Sales & Technical Center
Zilog, Incorporated
1442 U.S. Hwy 19 South
Suite 135
Clearwater, FL 33516
Phone: (813) 535-5571

Zilog, Incorporated
613-B Pitt St.
Cornwall, Ontario
Canada K6J 3R8
Phone: (613) 938-1121

United Kingdom

Zilog (U.K.) Limited
Zilog House
43-53 Moorbridge Road
Maidenhead
Berkshire, SL6 8PL England
Phone: 0628-39200
Telex: 848609

France

Zilog, Incorporated
Tour Europe
Cedex 7
92080 Paris La Defense
France
Phone: (1) 778-14-33
Telex: 611445F

West Germany

Zilog GmbH
Eschenstrasse 8
D-8028 TAUFKIRCHEN
Munich, West Germany
Phone: 89-612-6046
Telex: 529110 Zilog d.

Japan

Zilog, Japan K.K.
Konparu Bldg. 5F
2-8 Akasaka 4-Chome
Minato-Ku, Tokyo 107
Japan
Phone: (81) (03) 587-0528
Telex: 2422024 A/B: Zilog J