Publication Number 87000063
Release 3.0
January 1982

9508 MICROSYSTEM EMULATOR

USERS MANUAL

Millennium Systems, Inc.
19050 Pruneridge Avenue
Cupertino, CA 95014
Telephone: (408) 996-9109
TWX/TELEX # 910-338-0256

WARNING:

This equipment generates, uses and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. As temporarily permitted by regulation, it has not been tested for devices pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

The following procedures may help to alleviate the Radio or Television Interference Problems:

1. Reorient the antenna of the receiver receiving the interference.

2. Relocate the equipment causing the interference with respect to the receiver (move or change relative position).

3. Reconnect the equipment causing the interference into a different outlet so the receiver and the equipment are connected to different branch circuits.

4. Remove the equipment from the power source.

NOTE:

The user may find the following booklet prepared by the FCC helpful: "How to Identify and Resolve Radio-TV Interference Problems". This booklet is available from the U.S. Printing Office, Washington, D.C. 20402. Stock No. 004-000-00345-4.

This manual contains descriptive, installation, and operational information about 9508 MicroSystem Emulator, manufactured by Millennium Systems, Inc. The manual is divided into six chapters, as follows: Chapter 1 contains an overall description of the emulator - its hardware, software, and operating features, along with performance specifications; Chapter 2 includes installation and startup procedures, along with information for designing interfaces; Chapter 3 describes all operating modes of the emulator and contains procedures, along with examples, of how the emulator is used for testing and debugging; Chapter 4 describes each emulator command and its syntax; Chapter 5 contains an overview of the system hardware; Chapter 6 describes system software in a general manner. In addition, there are four appendices (A through D) with reference material and one appendix (E) containing schematic diagrams of all circuit cards contained in the emulator chassis.

All components that are microprocessor type dependent (they constitute an emulator option) are described in a separate addendum for each microprocessor. The addendum(s) must be used with this manual.

PROPRIETARY RIGHTS LIMITATIONS

MAINTENANCE AND SERVICE POLICIES

Unless notified to the contrary, any claims for operations assistance and/or service will be provided by Millennium Systems, Inc., from its plant in Cupertino, California. If you require any assistance on this product, please call Millennium Systems Customer Service on the toll-free, hot-line numbers listed below:

National	(800) 538-9320/9321
California	(800) 662-9231

CONTENTS

CONTENTS

Chapter Page

Appendices

Figures

Figures (Cont'd)

INTRODUCTION AND OVERVIEW

GENERAL DESCRIPTION

Millennium 9508 MicroSystem Emulator (figure 1-1) is an in-circuit emulator for testing, debugging, and integrating software and hardware in 8-bit microprocessor systems under development. The 9508 MicroSystem Emulator is a stand-alone desk top unit with external cabling and probes for connecting to the microprocessor system under test (referred to as the unit under test). In a typical installation the emulator is also interconnected to a host system, where programs for the unit under test are developed, and to a console terminal, where the user enters interactive commands to set up the tests, input data, and view test results.

The host system typically is the Millennium 9520 Software Development System, but the 9508 emulator is compatible with most other computer or software development systems and thus can be used as an add-on in-circuit emulator. The console terminal normally is the Millennium 9501 Display Terminal, but any other compatible CRT/keyboard terminal may be used.

In a typical testing and integration situation, a program is written and assembled in the 9520 Software Development System and then downloaded over the emulator-host communications link into RAM (emulation memory) located in the 9508 MicroSystem Emulator or into RAM in the unit under test (UUT). The host system is then disconnected from the 9508 and returned to program generation or other functions. Meanwhile, the user can utilize the 9508 emulator in a stand-alone mode for executing and debugging the program, testing circuits in the UUT, or integrating the program and circuitry. The user interacts with the 9508 emulator through the keyboard and CRT of the console terminal to set up test conditions and analyze test results.

CONNECTING TO THE UUT; EMULATOR OPTIONS

To perform tests in the unit under test (UUT), the 9508 emulator is connected as shown in figure 1-2. The emulator is personalized to the particular microprocessor type in the UUT with an emulator option. An emulator option consists of the following components: an emulator card, an emulator pod, emulator probe, a PROM board, and cabling (all microprocessor type dependent). These components must be installed to set the equipment up for testing a UUT with a particular microprocessor type. Either the emulator card or the emulator pod has a microprocessor that is an exact duplicate of the type used in the UUT; the emulator probe plugs into the microprocessor socket on the UUT, thus replacing the UUT microprocessor.

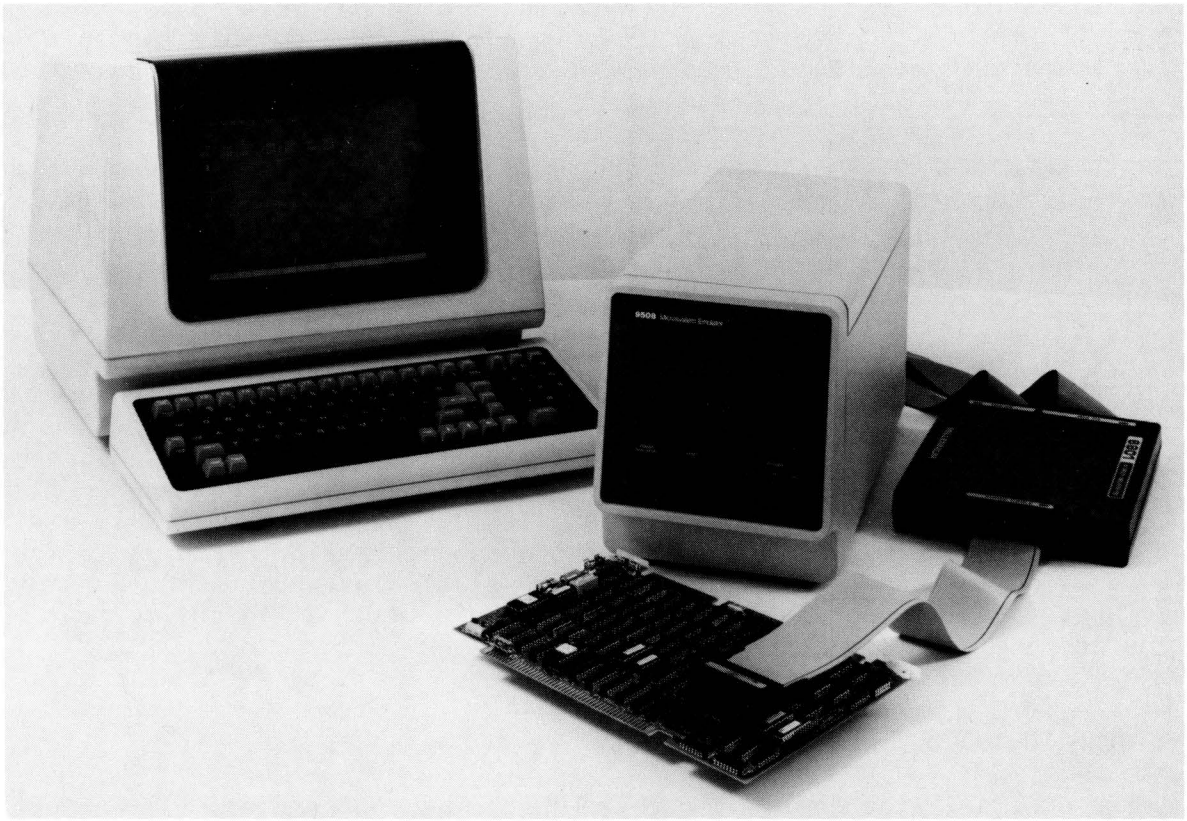


Figure 1-1. The 9508 Microsystem Emulator System

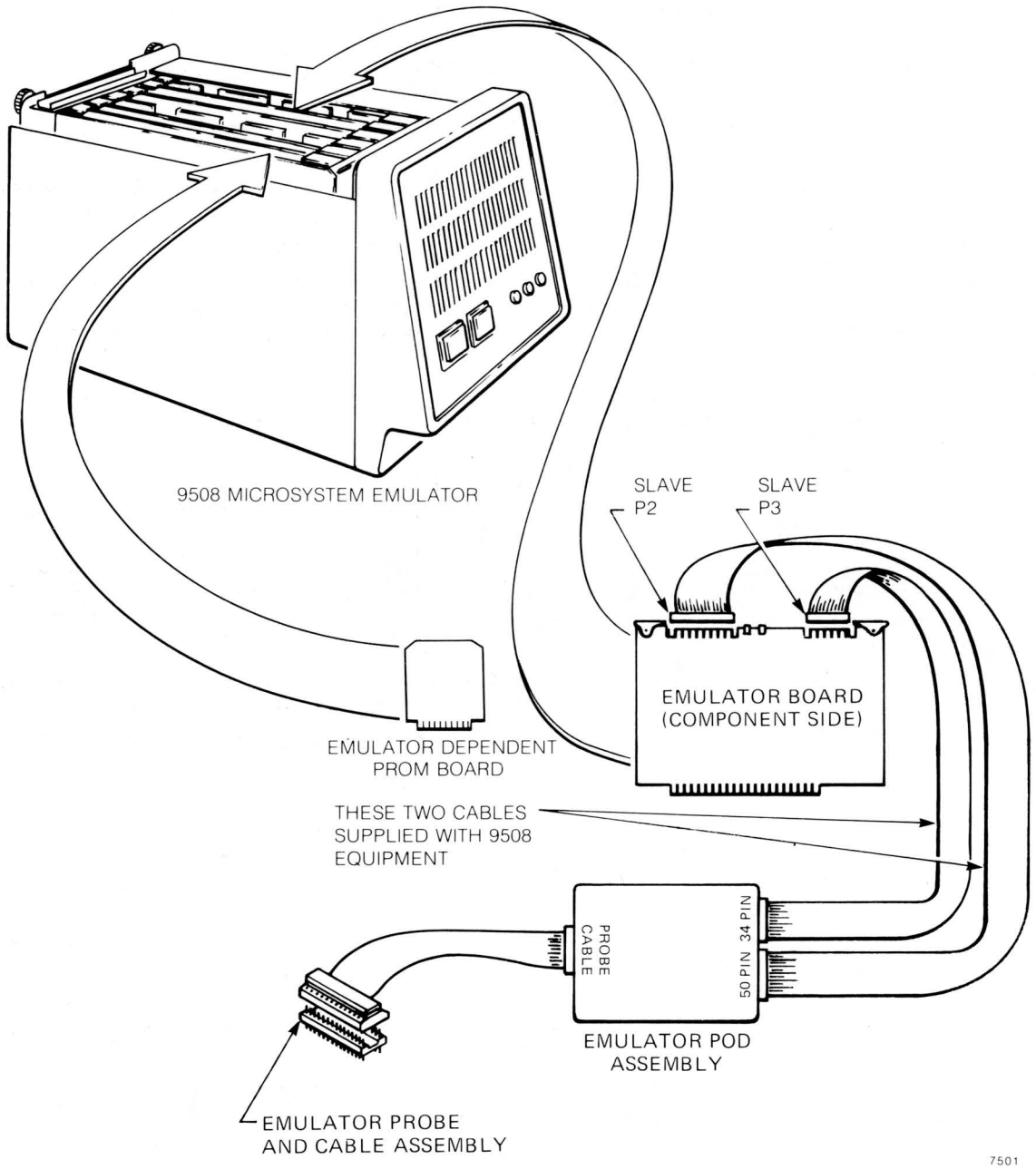


Figure 1-2. 9508 Microsystem Emulator, Typical Installation

The microprocessor types supported by the 9508 MicroSystem Emulator are as follows:

- o 6800/02
- o 6801/03
- o 6809
- o 6809E
- o 8048/49
- o 8021
- o 8041
- o 8080
- o 8085
- o Z80A

Each microprocessor (or microcomputer) listed as a line item above requires a different 9508 emulator option in order to personalize the equipment to that microprocessor (referred to as the target microprocessor). Each emulator option has a duplicate of the microprocessor chip being emulated, control circuitry for operating the duplicate microprocessor (referred to as the emulator processor), and the program code necessary for operating the chip within the 9508 emulator.

TESTING AND DEBUGGING WITH THE 9508 EMULATOR

To test, debug, or integrate software and hardware in a microprocessor system under test, the user always performs a variation of the following general procedure:

1. Download the program under test from the host system into the 9508 emulator memory or the UUT memory.

2. Set up test conditions for executing the program under test. This includes conditions imposed through the 9508 emulator software, as well as connecting the RTT test probe clips to any points in the UUT for signal monitoring during the execution.

3. Execute the program - either from the 9508 emulator or the UUT memory, either in continuous (real-time) or single-step mode, as set up in step 2 above.

4. Trace the execution as set up in step 2 above and halt on any breakpoint defined in step 2.

5. Display and evaluate the trace data accumulated during execution.

6. Read, display, evaluate, and alter memory locations, processor registers, and I/O ports in the UUT.

7. Disassemble any segment of the program under test and perform in-line editing in assembly language. Assemble patches back into the emulation memory. If testing UUT hardware, implement circuit changes. Repeat steps 2 through 7 as many times as necessary to achieve the testing and debugging objectives.

8. Upload the debugged and patched program to the host system.

The above general procedure is implemented in different ways during various steps of development of the microprocessor system. Initially, before the UUT hardware is ready, the program can be debugged using the 9508 system mode. In system mode, the clock and memory are furnished by the 9508.

As parts of the UUT circuitry become available, they are activated and tested one by one. For example, as the UUT system clock becomes available, it is used to run the emulator processor in the 9508. Likewise, UUT memory and I/O circuits can be introduced and tested one at a time. Testing is achieved by executing segments of the program under test out of the UUT memory, as well as signals in UUT circuits with the RTT data probe. Any testing done while connected through the emulator probe to the UUT and utilizing any UUT circuits is referred to as user mode. This is in-circuit emulation, in that the emulator processor substitutes for the microprocessor in the UUT. The trace, breakpoints, and other 9508 emulator debug features monitor the execution process allowing the operator to stop emulation, inspect and manually alter data accumulated at various locations during the execution.

As software and hardware development and integration nears completion, the program under test is executed entirely out of UUT memory. The 9508 can remain connected through the emulator probe and RTT data probe to monitor the process, until finally the 9508 emulator is disconnected, the UUT processor chip is installed in its socket, and the UUT is run independently.

OPERATING FEATURES

The key 9508 emulator features used for testing and debugging the program and hardware under test are listed in the following summary. All features (with some minor exceptions relating to the RTT data probe) are activated by the command set described in chapter 4.

Memory Mapping. The user can download the program under test directly into the UUT memory or into the 9508 emulation memory. To download into the 9508 emulation memory, segments of the UUT memory can be mapped into the emulation memory space. During early stages of development the program under test typically is loaded into emulation memory, progressively more and more of it is loaded into UUT memory as parts of UUT circuitry are activated. Mapping allows contiguous segments of the memory (and thus program execution) to be distributed between the emulator and UUT memories (MAP command in chapter 4).

Base Address Registers. Four base addresses can be specified by the user and then address memory locations can be specified relative to the base addresses. This is useful for working with relocatable program modules, which have been linked together (Bias command, chapter 4).

Emulator-Host Communications. The communications link between the 9508 emulator and most host systems can be set up so that the console terminal can communicate directly with the host (its operating system or communications interface). This allows both the 9508 emulator and host to be operated from a single console terminal (chapter 2 and chapter 3, section II).

Real-Time or Single Step Execution. When setting up test conditions for the test run, the operator can specify real-time or single step-execution (chapter 3, Introduction).

Real-Time/Single Step Trace. Different data generated during the test run can be monitored, saved, and displayed on the console terminal screen. If execution is in real-time, a history of address, data, and control signals can be selectively saved in a high-speed memory buffer and subsequently displayed. If execution is single step, the above data is saved. Also, the contents of the emulator processor registers are dumped and displayed after each instruction (chapter 3, section III).

RTT Data Probe. Up to eight test clips can be attached at circuit points in the UUT to sense signal status and record into the 128-entry buffer mentioned above (chapter 1, Hardware Overview; chapter 3, section III).

Breakpoints. One simple breakpoint and two mutually independent complex breakpoints can be set simultaneously. The complex breakpoints can be made dependent on the emulator bus information, as well as probe signals from the UUT circuitry (chapter 3, section III).

Memory, Processor Register, I/O Port Display and Modification. The user can display memory contents in hexadecimal or ASCII format and manually modify one location at a time, or fill large areas with a data pattern. In a similar manner, the user can access and load the emulator processor registers and I/O ports in the UUT (chapter 3, section 1).

In-line Disassembler and Assembler. The program under test can be disassembled, displayed in assembly language mnemonics, edited in mnemonic form, and assembled back into memory (chapter 3, section 1).

SYSTEM SOFTWARE

The 9508 emulator operating software architecture is based on an overall Executive Program, a Command Line Processor, which processes each command input from the console terminal, and a separate Command Routine for each command in the 9508 emulator command set. These software elements are described in detail in chapter 6.

The host system interfacing with the 9508 emulator must have a Convert utility module to format all data sent to the 9508 emulator. The host must also have Download and Upload utility modules, which perform the actual data transfer to and from the 9508 emulator. Each of the above utility program modules is an integral part of the 9520 Software Development System programs, but may need to be written and installed on other host systems (refer to appendix C).

SYSTEM HARDWARE

9508 Emulator Unit

The emulator is a fully enclosed chassis that contains up to seven circuit cards and a separate power supply module. All operator controls are located on the front panel of the unit, but cable connectors for the host, console terminal, and real-time trace signal probe are on the back panel. The back panel also mounts a cooling fan for the entire unit.

Circuit Card Chassis. All circuit cards, except the emulator dependent PROM board are inserted in a circuit card chassis. The connectors for the circuit cards are mounted on a motherboard, which extends across the bottom of the chassis. The motherboard provides data, address, control, and power bus lines interconnecting the cards. The following circuit cards are mounted in the card chassis:

1. The control processor card that uses a type 6800 central processor and functions as the master CPU in the 9508 emulator. Resident on this card are 14K bytes of PROM based operating system software, 2K bytes of protected system memory and the support circuits necessary to interface with other circuit cards.

2. The debug card supports breakpoints and controls interaction between the control processor and the target microprocessor.

3. The communications/RAM (Comm/RAM1) card contains 8K bytes of user available RAM, 4K bytes of operating system PROM, and additional circuits to support two RS-232 compatible ports (J1 and J2 on back panel). An additional 8K bytes of RAM are available with the inclusion of the optional Comm/RAM2 expansion card.

4. The real-time trace card performs two major functions: 1) gathers signal status information from the emulator processor bus and areas in UUT circuitry and stores this information in a 128-entry buffer memory for later recall and display; 2) implements the operator defined complex breakpoint equations to halt the emulator processor.

5. The emulator card contains control circuits for the emulator (target) microprocessor that replaces the unit under test (UUT) microprocessor during emulation.

The microprocessor dependent PROM board contains two 2732 EPROMs that provide 8K bytes of operating system memory. The PROM board is inserted in a separate connector, adjacent to the card chassis (refer to figure 1-2).

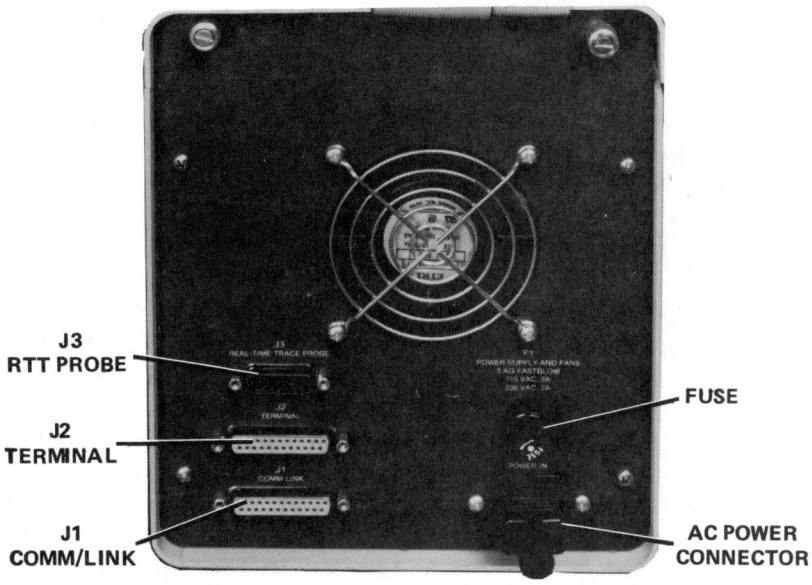
Power Supply. The 9508 emulator power supply provides four DC operating voltages to the printed circuit cards in the unit: ±5 and ±12. The power supply is regulated and has both overload and overvoltage protection circuits.

Cooling. Cooling is provided by an AC operated fan, mounted at the rear of the enclosure, that draws cool air through the power supply and across printed circuit cards. The enclosure cover must be replaced before AC power is applied. Operation without the top cover will defeat the cooling function of the fan and may cause damage to the power supply and/or the printed circuit cards.

Controls, Indicators, and Test Connectors. The following controls, indicators, and connectors are located on the front panel of the 9508 emulator (refer to figure 1-3):

POWER PUSH ON/OFF	A red backlighted alternate action pushbutton switch that illuminates when primary AC power to the emulator is turned on. When the switch is illuminated, this indicates not only the presence of AC voltage, but also the presence of +5 VDC output from the internal power supply.
RESET	A momentary contact pushbutton switch that reinitializes the internal master microprocessor and forces the system to jump to the self test program.
TRIGGER T1, T2, GND	Three test connectors that provide the emulator event trigger signals to an oscilloscope, logic analyzer, or other external equipment. The output from the T1, T2 connectors is a negative going logic level pulse. This negative pulse occurs whenever the corresponding trigger (T1 or T2) is asserted (refer to chapter 3, section III). The pulse begins 170 to 220 microseconds after the end of a bus cycle during which a specified trigger equation was satisfied. The pulse ends approximately 150 microseconds after the start of the next bus cycle.

After power is turned on and a testing/debugging session is begun, the operator has no need to use the above controls, except to reinitialize the emulator and restart the test/debug sequence. All other control is exercised from the console terminal keyboard.



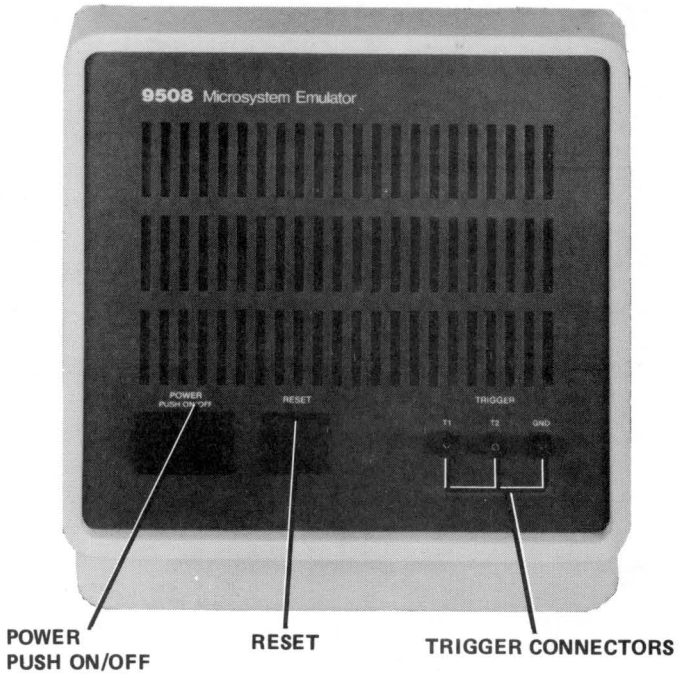


Figure 1-3. 9508 Front and Rear Panels

Rear Connector Panel. The following connectors are located on the rear panel of the 9508 emulator:

J1 COMM LINK	Standard RS-232-C, D-type, 25-pin (DTE configuration) female connector. Generally connected to the host system the 9520 Software Development System or user supplied host.
J2 TERMINAL	Standard RS-232-C, D-type, 25-pin (DCE configuration) female connector. Generally connected to the console terminal.
J3 REAL-TIME TRACE PROBE	15-pin, D-type female connector. Provides the connection for the real-time trace probe.
J4 POWER IN	Standard three-prong connector for application of AC power.
F1 POWER SUPPLY AND FANS	A fuse in the primary AC power input and cooling fan circuits.

Real-Time Trace Data Probe.

The real-time (RTT) data probe, figure 1-4, is a device that can sense signals at as many as eight-different locations in the UUT. The RTT data probe consists of eight-separate signal sensing test clips, a ground clip, and a clock sensing clip, all connected into a pod that buffers the signals and also allows the operator to switch-select different sensing (clocking) and data latching modes.

From the pod, the signals are applied to the 9508 emulator via the rear panel connector, J3. The signals are clocked and stored into the 128-entry RTT buffer memory once every emulator processor cycle, along with other signals monitored on the emulator bus by the RTT feature of the emulator. Thus, the RTT data probe signals become a part of the historical data gathered by the RTT feature during execution of the program under test.

Refer to section III of chapter 3 for a more detailed description of the RTT data probe and the RTT buffer memory.

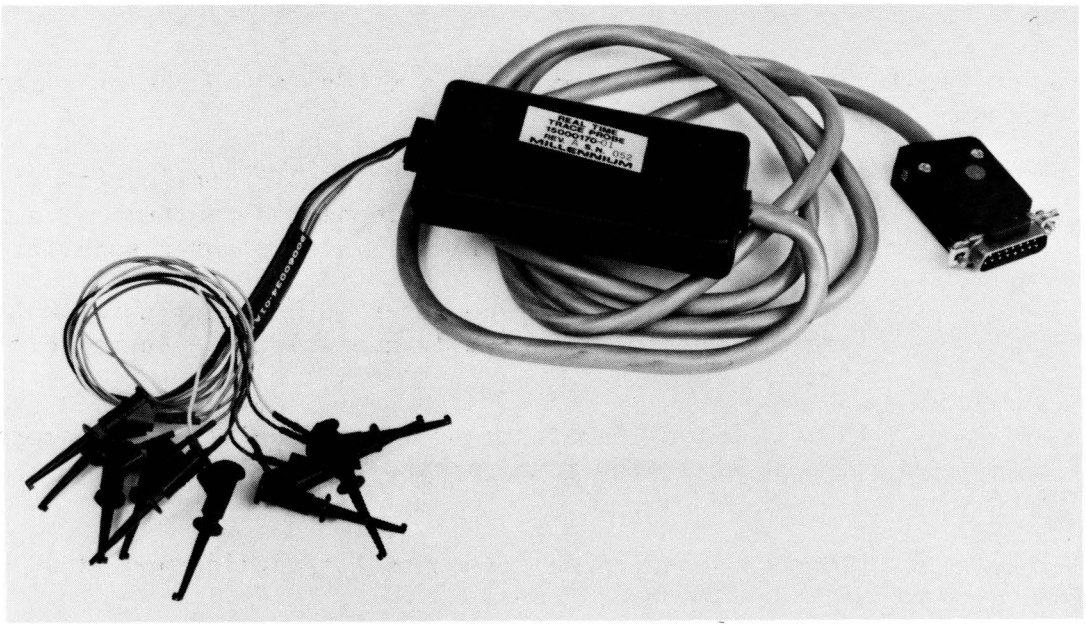


Figure 1-4. RTT Data Probe

Console Terminal

The console terminal enables the user to enter input commands and data to initiate and control the emulator test sequence. The console terminal may be the Millennium 9501 Display Terminal or another user furnished terminal. The only prerequisite for a user supplied console terminal is that it must be interactive, able to display 80-character lines, and be compatible with the RS-232C interface.

The 9501 Display Terminal, figure 1-5, consists of a microprocessor based intelligent CRT console and a separate ASCII keyboard unit that provides 96 alphanumeric and 32 control characters. The keyboard communicates with the CRT console via a cable connected to the connector panel at the rear of the CRT console (see figure 1-6). Also at the rear of the CRT console are other connectors and operator controls, as follows. (Refer to the manufacturer's Operator's Manual Model 950, TeleVideo No. B300002-001.)

Operator Controls and Cabling Connectors. Operator controls on the 9501 Display Terminal rear panel, figure 1-6, are the POWER switch, SEL SW. power source selection switch, the FUSE holder, CONTRAST control, the BAUD RATE switch bank and the FUNCTION switch bank.

Figure 1-5. 9501 Display Terminal

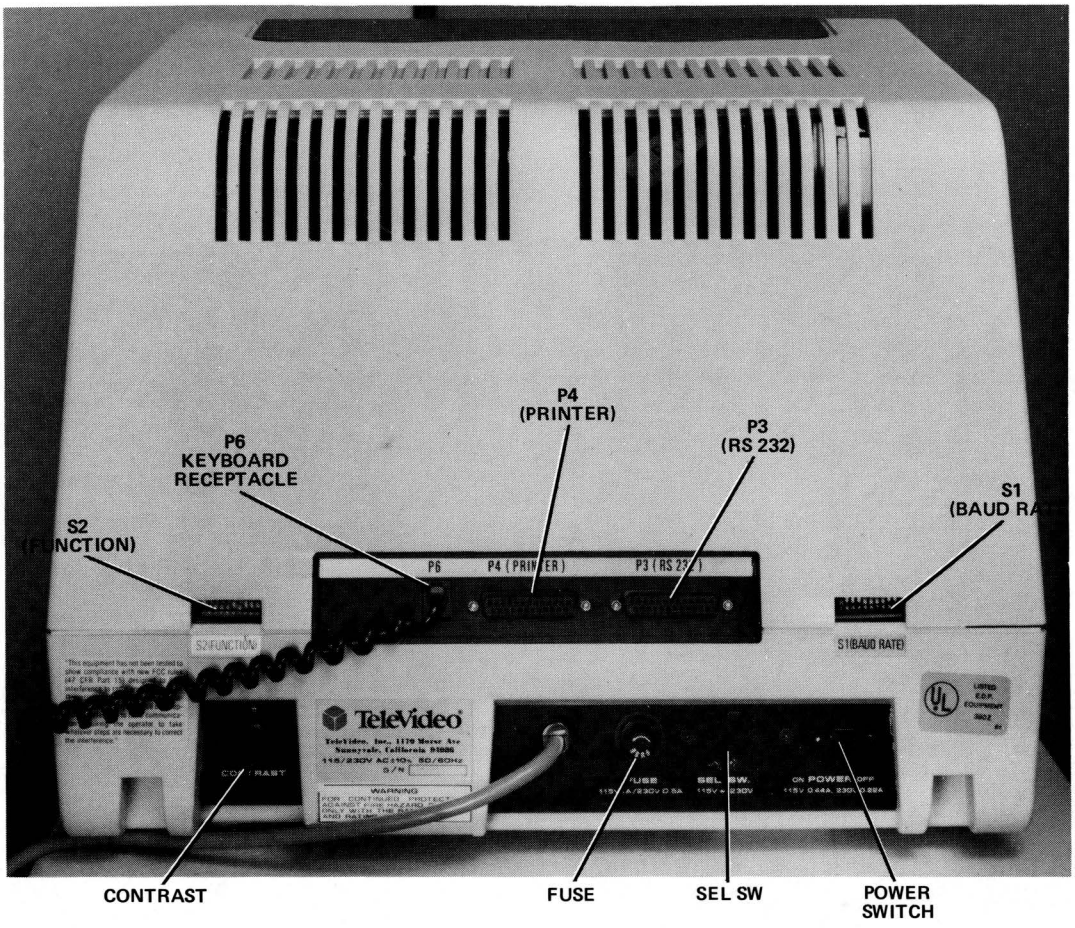


Figure 1-6. 9501 Display Terminal, Rear Panel Controls and Connectors

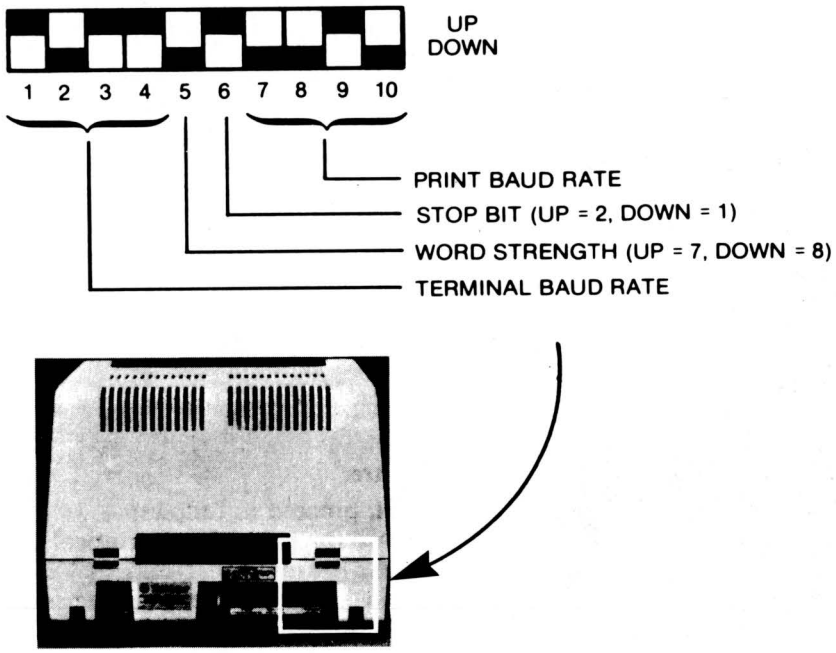
POWER Switch — A rocker switch that controls AC power to the console terminal. When the end of the switch with the white dot is depressed, AC power is applied to the unit; depressing the unmarked end of the switch removes power. One second after the power is applied, an internal beeper will sound, indicating the presence of AC power.

SEL SW 115/230 VAC — The position of this slide switch must coincide with the primary AC source voltage. The left position of the switch is for operation from a 115-VAC source; the right position is for a 230-VAC source. The switch blocking strip prevents inadvertent changing of the switch position.

CONTRAST Switch — A rotary potentiometer control that determines the white-to-black intensity of the visible display.

S1 BAUD RATE Switch Bank

A microswitch bank with 10 switches, that determine the input baud rate to the console terminal, the output baud rate from the display terminal, number of stop bits and word length. The following illustration (figure 1-7) shows the bit allocations and the baud rate selection bit combinations.

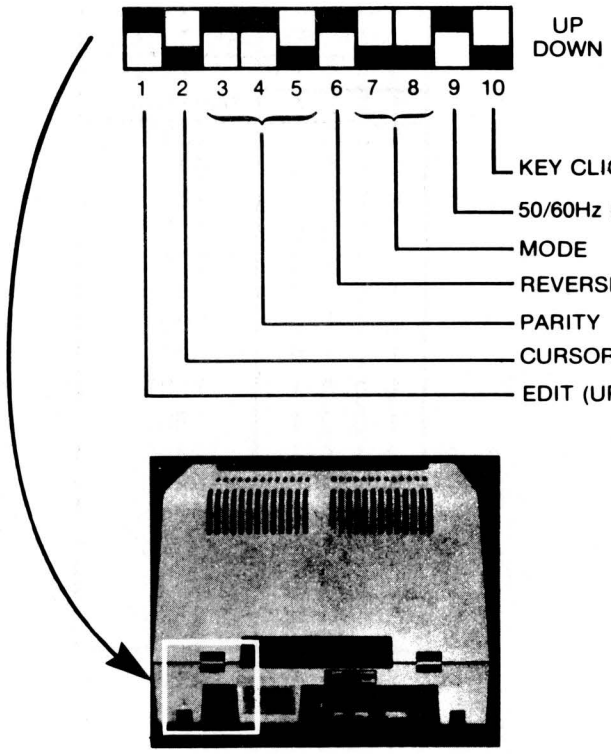


SWITCHES				BAUD RATE
7	8	9	10	Printer
1	2	3	4	Terminal
0	0	0	0	9600
1	0	0	0	50
0	1	0	0	75
1	1	0	0	110
0	0	1	0	135
1	0	1	0	150
0	1	1	0	300
1	1	1	0	600
0	0	0	1	1200
1	0	0	1	1800
0	1	0	1	2400
1	1	0	1	3600
0	0	1	1	4800
1	0	1	1	7200
0	1	1	1	9600
1	1	1	1	19200

Figure 1-7. Bit Assignments of the S1 BAUD RATE Switch Bank.

S2 FUNCTION Switch Bank

A microswitch bank with 10 switches that select operational functions of the display terminal. The following illustration (figure 1-8) shows the bit allocations, and bit combinations for transmission mode selection and parity selection.



SWITCH 3 4 5	PARITY
- - 0	NO
0 0 1	ODD (RECV and XMIT)
0 1 1	EVEN (RECV and XMIT)
1 0 1	MARK (Xmit parity disabled)
1 1 1	SPACE (Xmit parity disabled)

SWITCH 7 8	MODE
0 0	HALF DUPLEX
0 1	FULL DUPLEX
1 0	BLOCK
1 1	LOCAL

NOTE: Set 50/60Hz the same as the power line frequency to avoid display flicker.

Figure 1-8. Bit Assignments of the S2 FUNCTION Switch Bank.

FUSE 115 V1A/ 230 0.5 A — The fuse holder contains the primary power circuit fuse as specified in table 1-1. The intent of the fuse should never be ignored, nor should any attempt be made to defeat this protective device, as damage to the equipment may occur.

P3 (RS-232) — A 25-pin, female, RS-232C connector port for interfacing with the 9508 emulator. The internal configuration of the connector is described in the manufacturer's manual, and the signal lines utilized in connecting to the 9508 emulator are described in chapter 2.

SEL SW. 115V<-->230V — Switch position selects the primary input voltage for the console terminal. The switch position must match the primary local power source. A switch blocking strip is provided to prevent accidental switching.

P4 (PRINTER)	A 25-pin, female, RS-232C connector port that may be connected to an optional printer. This port is normally not used.
P6	A snap-type modular receptacle to which the cord from the keyboard unit is connected.

SPECIFICATIONS.

Tables 1-1 and 1-2 list selected operating characteristics and operating environment limitations for the 9508 emulator and the 9501 console terminal.

Table 1-1. Specifications - 9508 Emulator

CHARACTERISTIC	VALUE
AC Power Requirements	
Voltage	100, 120, 220, or 240 VAC +5%, -10%
Frequency	50-60Hz (±5%)
Main Fuse (F1)	3AG, 3Amp, Fast Blo (for 100/120 VAC); 3AG, 2Amp, Fast Blo (for 220/240 VAC)
Enclosure Dimensions and Weight	
Height	8.6 inches (21.844 cm)
Width	7.4 inches (18.796 cm)
Depth	16.4 inches (41.656 cm)
Weight	17 lbs (7.65 kilograms)
Environmental Limits	**Operating**
Ambient Temperatures	40° to 104°F (4.4° to 40°C)
Relative Humidity	20% to 80%
Maximum Wet Bulb	at 78°F (25°C)
	Storage
Ambient Temperatures	-8°F to 117°F
Relative Humidity	1% to 95% (No Condensation)

Table 1-2. Specifications - 9501 Display Terminal

CHARACTERISTIC	VALUE
AC Power Requirements	
Voltage	100-115 VAC (+10%) or 200-230 VAC (10%)
Frequency	50-60 Hz (± 5%)
Power Consumption	65 watts
Main Fuse	3AG, 1Amp, Fast Blo (for 100/110 VAC); 3AG, 0.5Amp, Slo Blo (for 200/230 VAC)
CRT Console Dimensions and Weight	
Height	14.0 inches (35.6 cm)
Width	16 1/2 inches (41.9 cm)
Depth	14 1/4 inches (36.2 cm)
Cabinet Weight	30 lbs (13.6 kg)
Keyboard Dimensions and Weight	
Height	3.0 inches (7.6 cm)
Width	16 1/2 inches (41.9 cm)
Depth	7 1/2 inches (19.0 cm)
Keyboard Weight	4.5 lbs (2.3 kg)
Environmental Limits	**Operating**
Ambient Temperatures	32°F to 122°F (0°C to 50°C)
Relative Humidity	10% to 95% (Non Condensing)
	Storage
Ambient Temperatures	-40°F to 149°F (-40°C to 65°C)
Relative Humidity	N/A (No Restriction)

INSTALLATION AND INTERFACE DESIGN

INTRODUCTION

This chapter describes the initial installation of the 9508 MicroSystem Emulator. If the 9501 Display Terminal and the 9520 Software Development System are used with the 9508 MicroSystem Emulator, installation is relatively simple. After connecting the 9508 emulator to its console terminal and the host, install the emulator dependent components, connect the real-time trace probe and any optional equipment. Then conduct a system power-on self test and the equipment is normally ready to operate. If the 9508 emulator is to be used with a console terminal or host system other than the 9501 and 9520, interface design will usually have to be performed prior to installation. Design parameters are provided for both the emulator-console terminal interface, as well as the emulator-host interface.

INSTALLATION PLANNING

The 9508 emulator may be located anywhere adjacent to other data processing or electronic test equipment, provided the temperature and humidity are within the limits specified in chapter 1. For optimum performance, the ambient temperature fluctuations should be kept as small as possible and a free flow of air should be allowed around the unit. The cooling air inlets and outlets should not be obstructed in any manner and the unit should not be operated with the top cover removed (it defeats the cooling function). The emulator is designed to be used as a tabletop unit.

AC Power and Grounding Requirements

The 9508 MicroSystem Emulator is wired to accept AC power input for 105V, 120V, 220V, or 240V at 50-60 Hz. Before installing the equipment, check the power specification label on the back panel to ensure that AC input requirements for the equipment coincide with the facility supply. If a discrepancy is noted, do not attempt to make adjustments and contact the Millennium Customer Service representative.

The emulator is equipped with a three-conductor power cable, which, when plugged into the appropriate receptacle, provides system chassis ground. The ground wire is the offset pin on the power cable three-prong connector. When operating the unit from a two-contact outlet, preserve the protection feature by using a three-prong-to-two-prong adapter and grounding the green pigtail of the adapter.

Cabling Requirements

The emulator is designed to interface to any console terminal that can send, receive, and display or print characters (e.g. an intelligent or non-intelligent CRT/keyboard terminal, teletypewriter, etc.) and to any general purpose computer programmed as a host software development system for the emulator. Both require at least a minimum 3-wire, RS-232C physical interface.

Signal connections to and from the 9508 emulator are made at the connector panel at the rear of the unit, see figure 2-1. The console terminal connection must be made at J2 and the host system connection at J1. If a real-time trace probe is used (for monitoring points in the circuits of a UUT), it must be connected to J3. In addition, test equipment that is to be activated by T1 and T2 trigger signals can be connected to the front panel TRIGGER connectors. While the host system is not connected to J1, provision can be made to connect a printer to J1, so that whatever data passes between the console terminal and the 9508 emulator through J2 is also echoed through J1 and a hard copy record of all interactive communications is printed.

INTERFACE DESIGN

If the 9508 emulator is to be installed and operated in conjunction with the Millennium 9501 Display Terminal and 9520 Software Development System, the physical interconnections, program interfaces, and operating protocol are all fully compatible and all units can be installed, checked out, and operated as shipped from the factory. If, however, either the console terminal or the host system is to be other than a Millennium standard product, its hardware and/or software interface with the 9508 emulator may not be compatible. The following paragraphs describe design elements of both the emulator-console terminal and the emulator-host interface.

Emulator-Console Terminal Interface

The emulator console terminal can be any 80 character-per-line CRT/keyboard or ASR-33 type send/receive terminal. The console terminal is connected to the J2 TERMINAL connector on back of the emulator by means of a cable - either the standard 10 ft. cable to the Model 9501 Display Terminal, or a user fabricated RS-232 cable.

The emulator-console terminal interface is RS-232C type, using a standard 25-pin D-type connector, as illustrated in figure 2-2 (on the emulator J2 is a female connector). A minimum of three wires must be used to implement the interface, with the other five maintained in the states shown in the figure. All remaining lines on the 25-pin connector can be left open or floating.

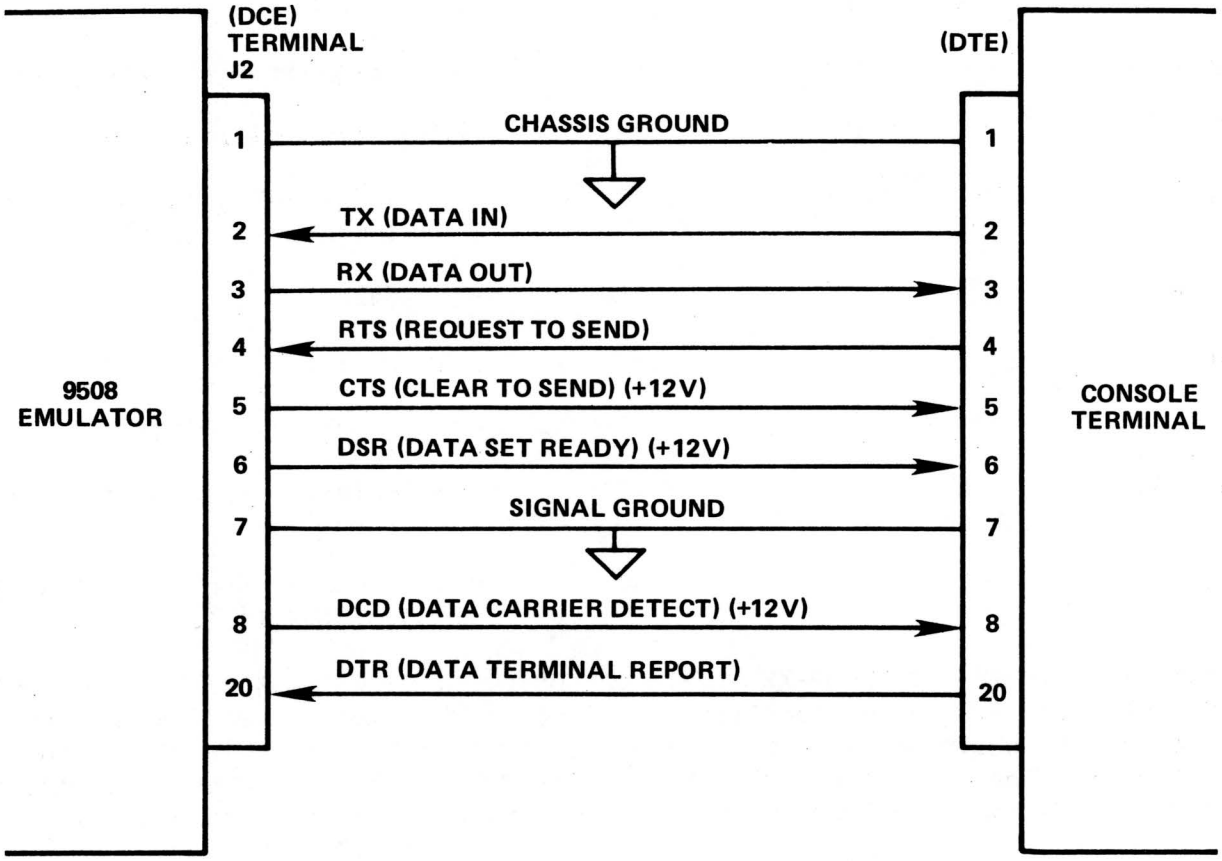
Figure 2-1. Rear Panel of the 9508 Emulator

Data transfer across this interface may be from 110 baud to a maximum of 9600 baud. Baud rate selection and parity are controlled by DIP switches at location A10 on the Comm/RAM circuit card in the 9508. Figure 2-3 shows how these switches are used to select baud rate and parity.

The data sent across the emulator-console terminal interface can be in 8-bit bytes. If either even or odd parity is selected, the eighth bit is used as the parity bit. In addition, each byte is also preceded by one start bit and followed by one or two stop bits. For baud rate of 110, two stop bits are used; for all higher baud rates only one stop bit is used.

There are certain special functions in the 9508 emulator that are activated by control characters from the console terminal keyboard (refer to chapter 4 for a complete listing of the special control characters). The codes for a selected portion of these control characters are stored in a type 2716 PROM on the control processor card of the emulator. These codes, listed in table 2-1, become effective when power is applied at initialization, or on reset (they are read and loaded into the master processor control memory). During operation, any code can be altered by using the TErmdef command (refer to chapter 4). The initial values can be changed by encoding a new PROM and replacing the existing PROM. Note, however, that the initial values may be read out of either of two locations in the same PROM, depending on the position of switch S8 on the DIP switch bank at location A10 on the Comm/RAM 1 card.

When power is applied to the system or the system is reset, the 9508 can be enabled to send out a string of up to 20 bytes to initialize the console terminal. To enable this transmission to the console terminal, place switch S8 in the OFF position (DIP switch bank at location A10 on the Comm/RAM 1 card). The character string to be sent out is stored in the same type 2716 PROM as is used for the special control characters. Table 2-2 lists the character string for the 9501 Display Terminal. If another type terminal is used, a different PROM may be encoded and installed in place of the existing one.



Minimum 3-wire Interface

PIN NO. (J2 on 9508)	FUNCTION
2	Transmitted data (TX) - data from console terminal to the emulator.
3	Received Data (RX) - data from emulator to console terminal.
7	Signal ground

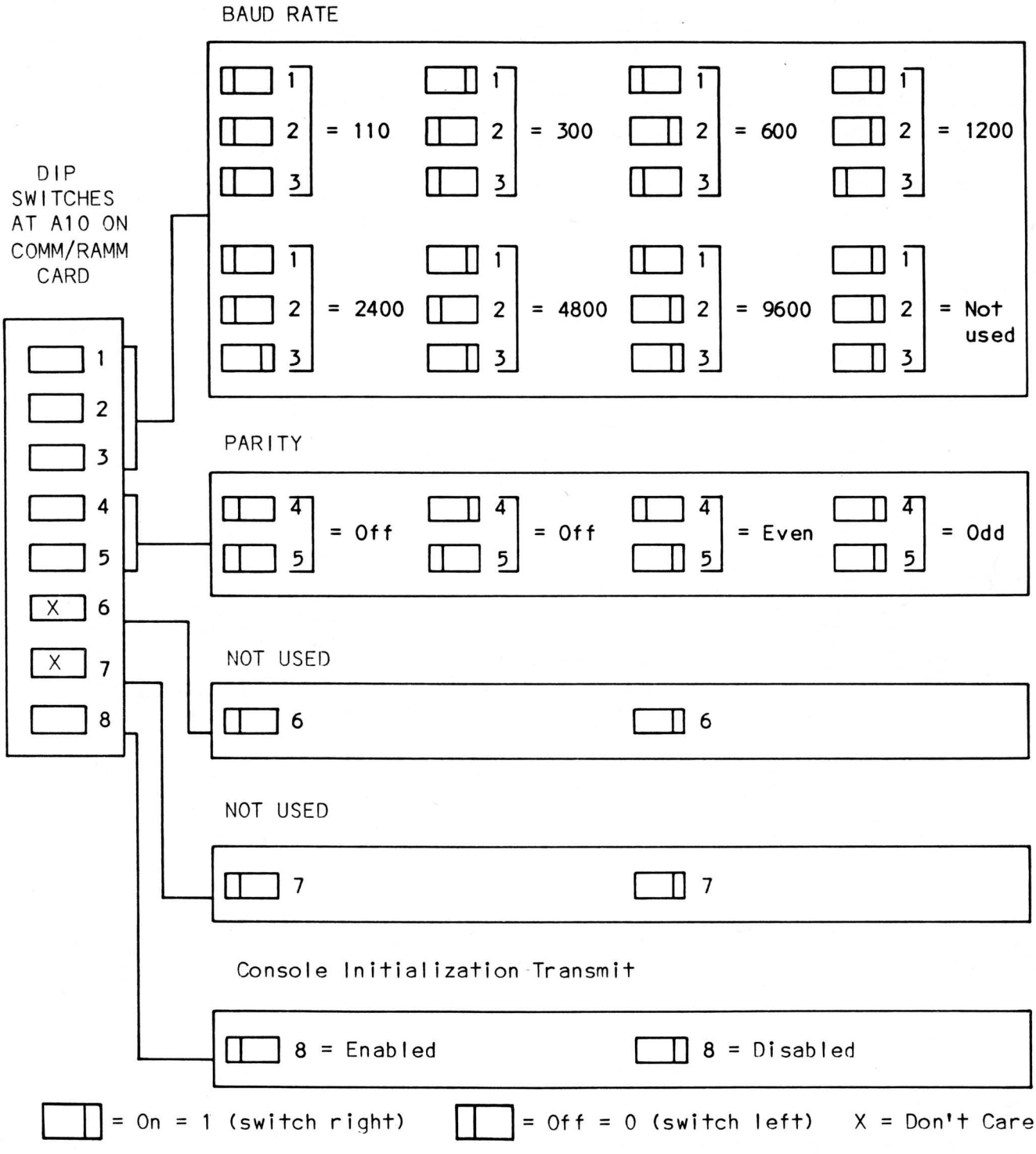
If connected, pins 4 and 20 must be maintained at +12 V or floating; pins 5, 6, and 8 must be allowed to remain at +12 V; connection to pin 1 is optional; all remaining pins on the 25-pin connector are not connected in the emulator.

Figure 2-2. Diagram of Emulator-Console Terminal RS-232 Interface

Table 2-1. List of Special Function Codes

SPECIAL FUNCTION KEY	PROM HEX ADDR* S8=1	S8=0	CODE IN PROM (HEX)	FUNCTION
CTRL A	7B0	7C0	01	Again (repeat last instruction)
ESC	7B1	7C1	1B	Escape (abort execution)
CTRL \	7B2	7C2	1C	Exit Interactive Communications (Terminal Passthrough) Mode.
CTRL H	7B3	7C3	08	Backspace
CTRL X	7B4	7C4	18	Delete line
RUB	7B5	7C5	7F	Rubout (delete character)
Space Bar	7B6	7C6	20	Alternately suspend and continue output to console.
CTRL P	7B7	7C7	10	Copy console terminal output to J1 COMM LINK port.

*Note: PROM is P/N 52080669-XX at A16 on control processor card; which set of PROM addresses is accessed for initial values of the special function codes depends on the position of switch S8 in the DIP switch bank at location A10 on the Comm/RAM 1 card. For use with 9501 terminal initial values are the same at either address location, but may not be for other type console terminals.



Note: Switches are factory set for 9600 baud, no parity, and intialization enabled.

Figure 2-3. DIP Switch Settings for J2 TERMINAL Port Console

Table 2-2. Console Terminal Initialization Characters

PROM HEX ADDR	CODE IN PROM (HEX)	MNEMONIC	FUNCTION
7E0	05	5	Count next 5 instructions (for transmission to console)
7E1	1B	ESC	
7E2	67	67H	Set 24 line display
7E3	1B	ESC	
7E4	66	66H	Buffer print off
7E5	0D	EOLCR	End of line, carriage return
7E6	-	-	-
|	|	|	|
|	|	|	|
7F4	-	-	-

Note: The character string listed in table 2-2 is stored in P/N 52080669-XX PROM at A16 on the control processor card and is used to initialize the 9501 Display Terminal. For other terminals different characters may be stored in PROM locations 7E1 through 7F4; the number stored at 7E0 indicates how many of the subsequent hexadecimal characters are to be sent out. The character string is transmitted only if switch S8 at location A10 on Comm/RAM 1 card is in the off position (0).

Emulator-Host Interface

The host system can be the Millennium 9520 Software Development System, or any other general purpose computer, programmed as a software development system for the 9508 emulator. The host system is connected to the J1 COMM LINK connector at the rear of the 9508 emulator by means of a standard RS-232 cable if the host is the 9520 Development System, or any other system.

If the host is a computer other than the 9520 Development System, numerous aspects of the communications link may need to be defined and certain program modules written for and installed on the host. The following is a summary of all major considerations that need to be made in order to define the communications link:

1. The physical connection (cabling). (See figure 2-4.)

2. How versatile or limited the communications link is and the procedure the operator will be using.

3. Format of the data transferred between the 9508 emulator and the host.

4. Encoding of the PROMs in which the initial selection of miscellaneous characteristics of the link are defined:

   a. Full or half duplex

   b. Synchronization between the host and emulator at the start of a data transfer.

   c. Link handshake protocol during data transfer.

   d. Baud rate.

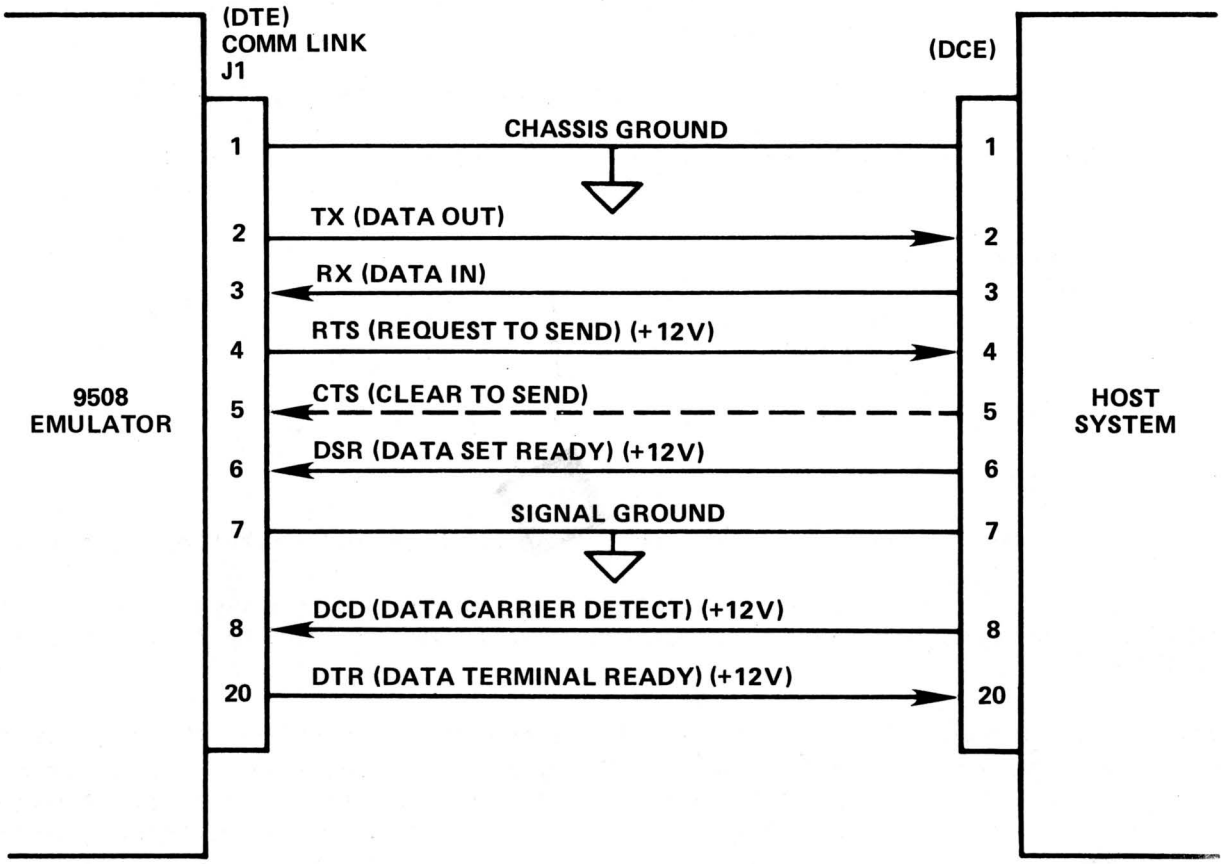
5. Interfacing programs that need to be written and installed on the host.

Together, the five preceeding requirements define all major physical and software characteristics of the link. Each is explained separately in the following paragraphs.

Physical Connection. The interface between the emulator (COMM LINK connector J1) and the host system is EIA Standard RS-232C, using a 25-pin D-type connector, as shown in figure 2-4 (on the 9508 emulator, J1 is a female connector). A minimum of three wires must be connected to the host to implement the interface, with up to five other optional lines available, as described in the figure. All remaining lines are unused.

The 9508 emulator can be cabled to the host in two basic ways, as shown in figures 2-5 and 2-6. The first is to connect the 9508 emulator to a console port of the host computer, where a console port is defined as one from which the host operating system will accept commands as well as data. Variations of the console connection are shown in figure 2-5.

The second basic way is to connect the 9508 emulator to a peripheral port of the host, where a peripheral port is defined as one through which only data can be passed in either direction, but no commands. The peripheral connection is shown in figure 2-6.

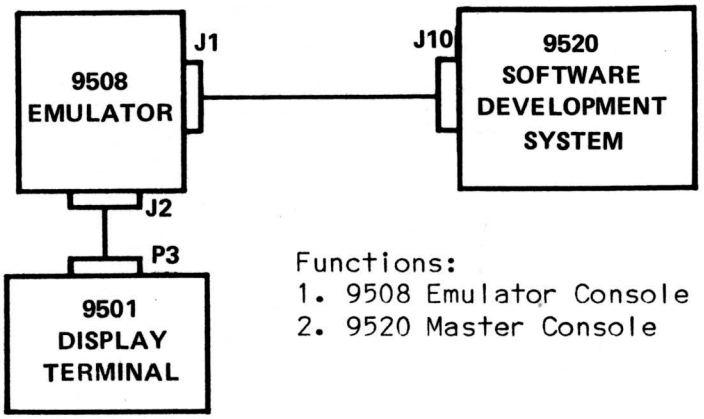


Minimum 3-wire Interface

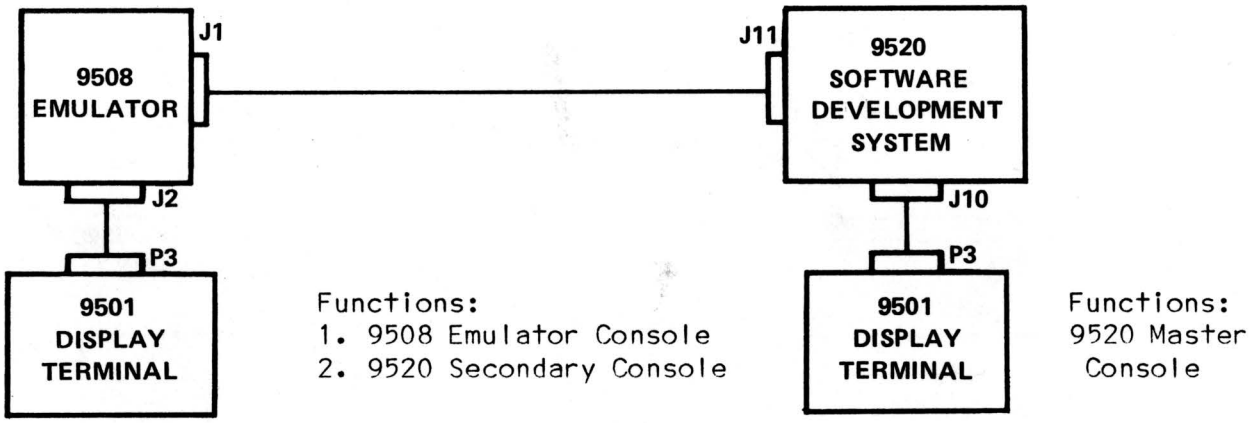
PIN NO. (J2 on 9508)	FUNCTION
2	Transmitted data (TX) - data from emulator to host.
3	Received Data (RX) - data from host to emulator.
7	Logic ground

If connected, pins 6 and 8 must be maintained at +12 V or floating; pins 4 and 20 must be allowed to remain at +12 V; connection to pin 1 is optional; all remaining pins on the 25-pin connector are not connected in the emulator.

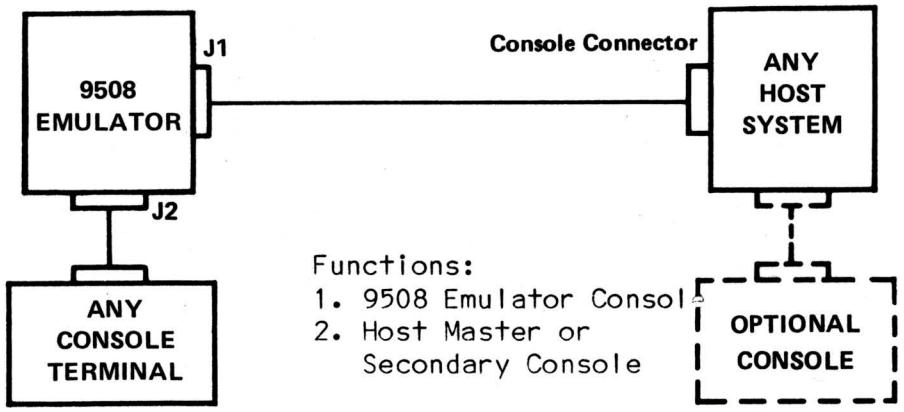
Figure 2-4. RS-232 Interface, 9508-Host System



a. Single Console Connection to 9520

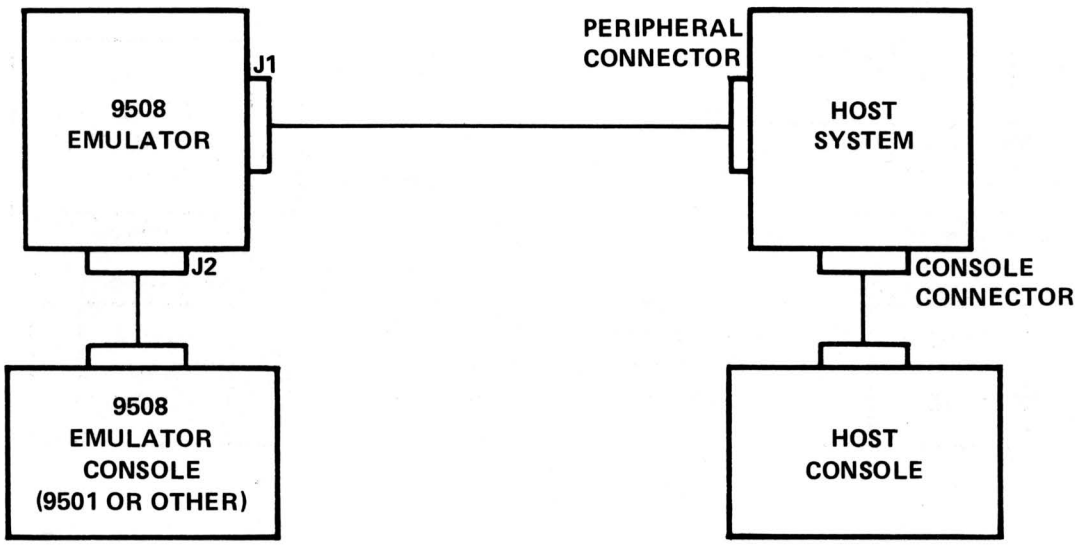


b. Multiple Console Connection to 9520



c. Connection to Host other than 9520

Figure 2-5. 9508 to Host System Interconnections



Note: On 9520 the peripheral connector J12 only outputs data and therefore is not suitable for connecting to 9508 (only downloading can be done, but no uploading).

Figure 2-6. Dual Console Interface, 9508-Host System

If the 9508 emulator is connected to the host in the console configuration, the emulator hardware/software architecture allows different levels of communication between the host and the emulator console terminal. In general, the console configuration is used in applications where the user must communicate with the host operating system, or at least with a specific utility in the host. To do this, the 9508 can be instructed to allow its console terminal to function as a console of the host - either a host master console (a. of figure 2-5), or a host secondary console (b. of figure 2-5). This console sharing feature is useful in installations where the host is in a remote location with respect to the 9508 emulator and the emulator user.

If the 9508 emulator is connected to the host in the peripheral configuration, no exchange of commands between the host and the emulator console is possible. The host must always be activated and commanded from its own console and the emulator-host interface is able to carry data only (during downloading or uploading).

The 9508 emulator hardware/software architecture includes software features that allow interfacing to various hosts, using the two cabling configurations and different operating procedures (or modes). The procedures are described in the following paragraphs.

In order to transfer a program to or from the host, it is necessary to perform the following:

1. Start the transfer program in the 9508, the RHEX command is used for downloading and the WHEX command for uploading.

2. Start the transfer program in the host. For the 9520, these programs are DOWNLOAD and UPLOAD, respectively.

3. Ensure that both programs are ready to begin the program transfer.

4. Transfer the program data.

Communication Procedures. There are three different procedures for downloading program material from the host to the 9508 emulator (or uploading to the host), for use with the two physical connections and various hosts. These are referred to as Case 1, 2, and 3. In addition, there is a general purpose interactive communications mode for performing other operations with the host programs. The applications of each are described in the following paragraphs to assist the interface designer in selecting the one most suitable. Refer also to chapter 3 for more details, including the step-by-step sequence of the procedures.

1. Case 1. The Case 1 procedure is used when the 9508 emulator is connected to the host in the peripheral configuration (figure 2-6). It can be used to download or upload, but does not allow any command exchange. That is, if the communication link is activated for uploading, the host must be ready to receive data, without any prompting. Likewise, if the link is activated for downloading, the 9508 emulator will interpret the first byte following synchronization to be data.

When the program data is being transferred, the 9508 emulator operates in the transfer mode. This mode is also activated in a subordinate capacity during the Case 2 and 3 transfers.

2. Case 2. The Case 2 procedure can be used when the 9508 emulator is connected to the host in the console configuration (figure 2-5). It is similar to Case 3, except only one command is sent to the host. When the user enters the RHex command together with a filename, the filename is appended to a DOWNLOAD command and sent to the host (refer to chapter 3 for the procedure). Thus, the single command to the host is predetermined by the 9508 emulator software. (For uploading, the WHex command is entered and causes an UPLOAD command to be constructed and sent to the host.) Thus, no program editing or other host operations can be carried out under Case 2.

3. Case 3. The Case 3 procedure can be used if the 9508 emulator is connected to the host in the console configuration (figure 2-5). Case 3 procedure is the most common and versatile, because it allows the emulator console terminal to communicate interactive commands directly to the host, as well as transfer program data. During interactive communications the emulator becomes entirely transparent and the 9508 emulator console terminal and host both function under the host operating system and/or communications interface protocol. The user can do program editing in the host, or any other host operations. However, the host operations must always be terminated with a download or upload data transfer.

4. Interactive Communications (Terminal Passthrough) Mode. This mode becomes available if the 9508 emulator is connected to the host in the console configuration. It allows the emulator console terminal to become a general purpose command entry terminal of the host and communicate directly with the host operating system and/or communications interface, just as in Case 3 above. However, it does not allow the downloading or uploading of data and thus can be only a feature supplementary to the Case 1, 2, or 3 procedures.

Data Formats (for Downloading and Uploading); Conversion Program Module.

Refer to chapter 6 for a description of the data formats required for downloading and uploading, and the Convert utility program that performs the necessary conversion.

The data sent across the 9508 emulator-host interface can be in 8-bit bytes. If either even or odd parity is selected, the eighth bit is used as the parity bit. In addition, each byte is also preceded by one start bit and followed by one or two stop bits. For the baud rate of 110, two stop bits are used. For all higher baud rates only one stop bit is used.

Download and Upload Programs. If a host other than the 9520 system is used, and a standard operating system utility is not used, Download and Upload program modules must be written and installed on the host. These modules become utilities of the host operating system and function to communicate with the Rhex and Whex command routines in the 9508 emulator, in order to synchronize the 9508 emulator and the host at the start of a download or upload, carry out the transfer of data blocks, and issue terminating commands. The specifications for writing the Download and Upload program modules are contained in appendix C. Refer to chapter 6 for a more detailed description of the interaction between the Download, Upload, Rhex, and Whex programs.

Miscellaneous Characteristics of the Emulator-Host Link. The 9508 emulator can support varying user selectable operating characteristics of the 9508 emulator-host communications link, in order to accommodate different host systems. Many of the characteristics are user selectable at all times during operation, but their initial values (following power on) can be defined during interface design. The following are the most significant link characteristics that must be considered when designing the 9508 emulator-host interface (for a complete list refer to Linkdef command in chapter 4):

1. Full or half duplex - the link can be either one.

2. Synchronization - The 9508 software provides for an exchange of a synchronization character with the host prior to starting a transfer of data blocks during downloading or uploading. If this capability is incorporated into the host Download and Upload utilities, it allows program synchronization between the emulator and host.

3. Link protocol - during the transfer of data blocks either of two handshake protocols may be used: 1) electrical protocol, utilizing the DSR and DTR lines of the RS-232 interface, or 2) an ACK character returned by the 9508 emulator to acknowledge each correct data block and a NAK character to indicate a fault. The electrical (DSR/DTR) protocol is always supported in the 9508 emulator RS-232 interface hardware and can be used to stop transmission at any time to protect against overflowing data buffers. The ACK/NAK protocol is implemented by the Rhex and Whex subroutines and its intended purpose is to allow the acknowledgement of the receipt of a correct data block, or request retransmission in case of an incorrect data block. The ACK/NAK provisions for the host must be specifically incorporated in the Download and Upload utilities. Refer to chapter 6 for details about either protocol and to appendix C for Upload/Download program specifications.

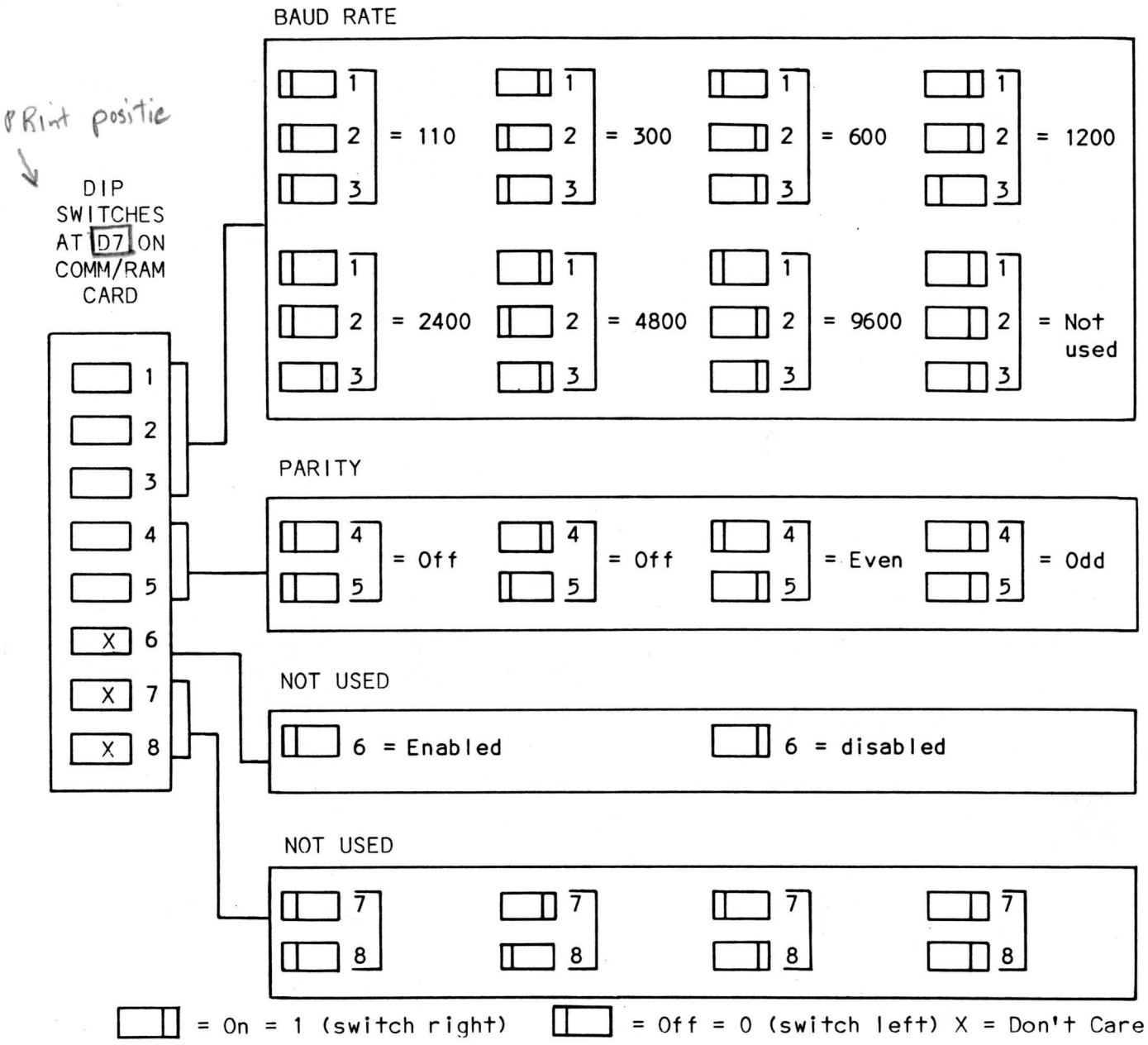
4. Baud rate - the 9508 emulator RS-232 interface provides for data rates up to 9600 baud. If the electrical (DSR/DTR) link protocol is used, the baud rate can be set at any value up to 9600 and the DTR line will halt data transfer when an overflow condition becomes imminent. If the DSR/DTR lines are not utilized, the baud rate must be limited to approximately 2400, to protect against overflowing the emulator data buffers.

Initial values (following a power on or reset) of the preceding link characteristics, except the baud rate and parity are stored in a type 2716 PROM at location A16 on the control processor card (refer to table 2-3). Baud rate and parity are set by DIP switches at location D7 on the Comm/RAM 1 card (see figure 2-7). To implement a different set of initial values, a new differently encoded PROM may be installed. Note that during operation, any of the initial values may be superseded by those entered with the Linkdef command (refer to chapter 4).

Table 2-3. Initial Characteristics of the Emulator-Host Link

PROM HEX ADDR	STANDARD CODE IN PROM (HEX)	LINK CHARACTERISTIC	OPTIONAL CODE
7D0	00	Full Duplex Data Format	Half Duplex=01
7D1	00	Binary Data Format	TekHex=01
7D2	00	ACK/NAK Link Protocol	DSR=01; None=02
7D3	00	Abort on Error	Ignore=01
7D4	30	ACK Character is ASCII 0	
7D5	37	NAK Character is ASCII 7	
7D6	53	Start Synchronization Character is ASCII S	
7D7	03	Host Abort Character is CTRL C	

Note: PROM is P/N 52080669-XX at location A16 on control processor card and codes shown in table are those programmed in the factory; codes for alternative characteristics are given in optional code column.



Note: Switches are factory set for 9600 baud and no parity.

Figure 2-7. BAUD RATE and Parity DIP Switch Settings

UNPACKING AND INSPECTION

All of the hardware and software items that are required to install and operate the equipment at the user's site are shipped in packaged units. External cables are included for connecting the console terminal to the 9508 MicroSystem Emulator. All items shipped for the standard system configuration are listed in table 2-4.

The 9508 MicroSystem Emulator (and optional 9501 Display Terminal Console, if provided) was thoroughly inspected and checked out at the factory prior to packaging for shipment. After removing the equipment from it's box, inspect for scratches, dents or other damage that might have occurred during shipping. Refer to the shipping papers to verify that all components are present.

If physical damage is evident when received, do not operate the equipment. File a claim with the shipping firm immediately and notify Millennium Systems Customer Service department at once. Millennium will arrange for repair or replacement of the equipment without waiting for settlement of the claim against the carrier.

If the equipment must be returned to Millennium, attach a tag showing the owner, address, serial number, and a description of the failure. The original shipping carton and packing material should be reused with the RMA (Returned Material Authorization) number prominently displayed. An RMA number must be obtained by calling Customer Service on the toll-free, hot-line numbers listed in the preface. Millennium Systems Technical Support Representatives and Customer Engineers are available to provide consultation and assistance on request.

Table 2-4. 9508 Emulator Packaged Items List

QTY	ITEM DESCRIPTION
Standard Items	
1	9508 MicroSystem Emulator Unit
1	AC Power Cord
1	RS-232 Interface Signal Cable, 10 ft. long P/N 19910082-01 (for 9501 Display Terminal)
1 or 2	Emulator Dependent Control and Signal Cables to Emulation Pod (Quantity is Emulator dependent)
1	Microprocessor Dependent Emulator Module Consisting of the following: 1 Emulator Printed Circuit Board 1 Emulation Pod 1 PROM Board 1 Processor (40-Pin Connector) Probe Cable
1	Real-Time Trace Probe Assembly
1 Pkg	Document Package which consists of the following manuals: a. 9508 MicroSystem Emulator Users Manual b. Users Manual Addendum for Microprocessor Dependent Emulator Module
Optional Console Terminal	
1	9501 Display Terminal with Keyboard Unit (TeleVideo Model 950)
1	950 Display Terminal Operators Manual (provided with 9501 Display Terminal)

PREPARATION OF EQUIPMENT FOR USE

CAUTION

Do not perform the following procedures with the equipment power on, because damage may occur.

Use the following procedures to set up the equipment and complete internal and external connections.

Preparation and Setup of the 9508 MicroSystem Emulator

1. Remove contents of the microprocessor dependent emulator module from its packaging (refer to table 2-4).

2. Loosen the two thumbscrews at rear of the 9508 (figure 2-1) and remove the top cover.

3. Verify that all circuit cards are installed in the locations shown in figure 2-8. Check that they are firmly seated in their connectors.

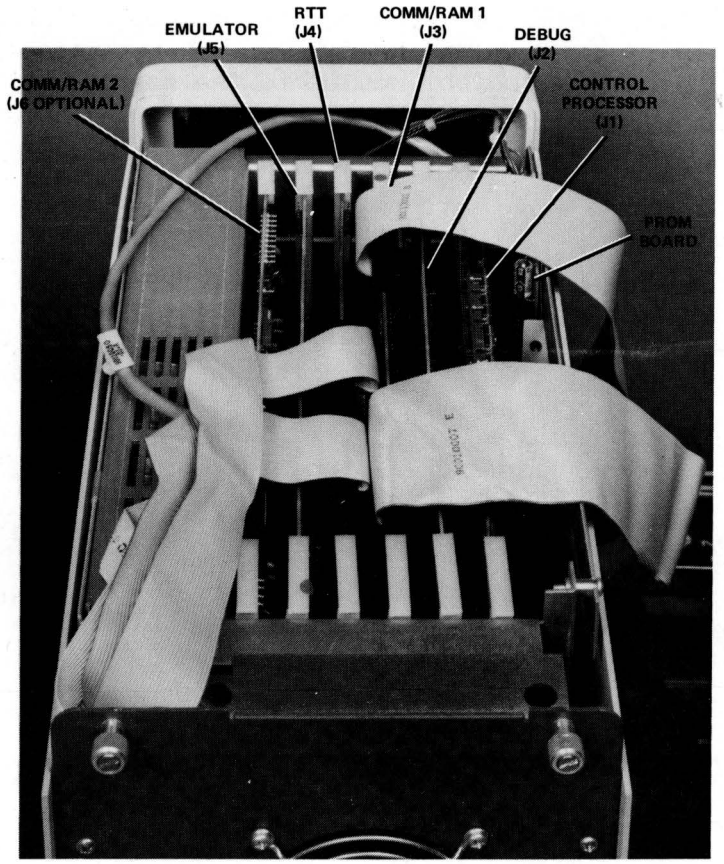
4. Insert the emulator card in the universal card cage in the fifth slot from the left side and push down on both card release tabs to seat the board firmly.

5. If the optional Comm/RAM 2 card is shipped with the emulator, install the card in the sixth slot. Note: Comm/RAM 1 and Comm/RAM 2 cards are the same, except Comm/RAM2 does not have the ACIA and PROM chips installed at locations 5B, 8B and 5C, 8C and does not have the same configuration jumper wires installed.

6. Remove the control and signal flat cables from their plastic bag. Connect the flat cable connector(s) on to the mating edge connector(s) at top of the emulator card. (The number of cables is directly related to the emulator card, but will never exceed two.)

7. Inside the chassis at the upper left-hand front corner there is a 40-pin connector (see figure 2-8). Insert the emulator dependent PROM card into this 40-pin connector with the component side facing the other circuit cards. Connect the edge cable onto the 4-pin connector at top of the PROM card. (The cable connector is keyed so that pin 5 is blank.)

8. Remove the tie-down bar by loosening the two screws. Guide the emulator card cables over the slot at the rear of the cabinet so the cables lie flat in the slot and under the tie down bar. Re-install the tie down bar and tighten the two retaining screws (clock-wise).



CIRCUIT CARD	PART NO.	LOCATION
Control Processor	13000244	J1
Debug	13000044	J2
Comm/RAM 1	13000015	J3
Realtime Trace (RTT)	13000170	J4
Emulator	(Processor Dependent)	J5
Comm/RAM 2	13000015	J6
PROM Card	(Processor Dependent)	

Figure 2-8. Circuit Card Locations in the 9508 Emulator.

9. Reinstall the top cover and tighten the thumbscrews.

CAUTION

The top cover must be replaced before AC power is applied. Operation without the top cover will defeat the cooling function of the fan and damage to the power supply and/or the printed circuit cards may result.

10. Connect the signal and control cables to the corresponding 50-pin and 34-pin connectors on the emulation pod.

Preparation and Setup of the 9501 Display Terminal

The preparation and setup requirements for the 9501 Display Terminal are described in the Operators Manual shipped with the equipment. Refer to the manual for various switch settings, configuring of interface connectors and setting the desired baud rates, word lengths, and stop bits.

The 9501 Display Terminal is equipped with an internal self-test diagnostic program, which can be conducted in a standalone mode, i.e., without the need for connecting the terminal to other equipment. The self test checks video attribute functions and also the communications path between the printer output port and terminal input port of the 9501 terminal. The RS-232 cable shipped with the 9508 MicroSystem Emulator is used for the test connections as described in the procedure.

1. Connect the power cable to the facility AC source.

2. Position the four terminal baud rate switches (7, 8, 9, 10 on S1 BAUD RATE switch bank to be identical to the print baud rate switches (positions 1, 2, 3, 4). This sets the baud rate for the printer output port (P4 PRINTER) the same as for the RS-232 input port (P3 RS-232), to which the 9508 emulator will be connected.

3. Switches 5 and 6 (of S1) are not to be changed for the self test.

4. Connect the RS-232 cable from connector P3 to P4.

5. Turn on the display terminal POWER switch. Allow 10 to 15 seconds for the equipment to warm up and display the cursor in the upper left hand corner of the screen.

6. Press and hold the SHIFT key while depressing the SET-UP/NO SCROLL key.

7. Press and release the 1 key.

   The screen should display all available characters and attributes for reverse video, grey shade, blinking line and underscore.

8. Press and release the 2 key.

   a. All but the bottom line of the screen should be cleared.

   b. After approximately 3 seconds, the word PASS should by displayed in the upper left hand corner of the screen to indicate the test was successful.

   c. If FAIL 2 appears, verify that baud rate switches are set the same for both ports (refer to step 2). Also verify that the RS-232 cable is securely connected between P3 and P4 and repeat the test.

   d. If the problem still persists, contact Millennium Customer Service.

9. Toggle the display terminal POWER switch off.

External Cable Connections

External cable connections for interfacing the 9520 Software Development System or any other host system) to the 9508 emulator and to the 9501 Display Terminal (or any compatible display terminal) are shown in Figure 2-5 and 2-6.

Initial Startup of Emulator

Use the following procedure to power up the emulator and console terminal from a cold start:

1. Toggle on the POWER switch of the console terminal. The physical location of this switch varies with different terminals, and may be located at the front, rear or side of the terminal chassis.

When power is applied, the response varies with different terminals. A typical response is as follows:

   a. The terminal bell sounds within 1 second to indicate power is on.

   b. After 10 to 15 seconds, the cursor appears in the upper left-hand corner of the screen.

At this time the operator can adjust the contrast (or intensity) control to obtain the desired brilliance of the screen.

2. Depress the front panel POWER switch on the 9508 emulator to turn it on. A power-on check is automatically performed by the 9508 emulator, whenever the system is powered up or reset, to verify the operational readiness of hardware components. The power-on check is implemented by a self-test program permanently stored in a type 2716 EPROM on the control processor card. The self-test program is automatically initiated when the POWER switch is turned on (illuminated), or when the RESET pushbutton is depressed to reset the emulator. The self-test performs an operational check of the modules in the 9508 and the results are presented on the display terminal. Assuming the emulator module is a Z80, and the 9508 is in good operating condition, the following message will be displayed on the display terminal:

```
9508 READY VER xx  (xx = operating system revision level.)

PROCESSOR=Z80 VY.Y
W=0000 X=0000 Y=0000 Z=0000
MAPPED  RAM 1 START=0000  END=1FFF
EVent Address  Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt   TMode  SEL TRIG
  1   = OFF   = OFF  All   XXXX XXXX  1      0 E1        0 MS  T1 IND       T1
  2   = OFF   = OFF  All   XXXX XXXX  2      0 E2        0 MS  T2 IND       T2

BReak T1=Dsbl  T2=Dsbl    Count=    0 MS    Qual=ALL
  LOC  MNEM OPRD/EADDR  A  SZHPNC  BC   DE   HL  IX/IY  SP  R  I  IM12 PCNEXT
  0000 XOR  A           00 000000 0000 0000 0000  0000 0000 00 00  0DD   0000
                        00 000000 0000 0000 0000  0000

REGBRK CONDITIONS:

SYSTEM MODE
D>
```

The system is now operational and ready to accept commands from the console terminal keyboard. Refer to chapter 3 for operating procedures and for an explanation of each line of the startup message above.

Resolving Start Up Problems

When the 9508 MicroSystem Emulator is initialized at power on or as a result of a system reset, the self test program checks the status of the memory components and the emulator circuits. Any error condition detected during the self test causes an error message to be displayed on the display terminal. The error message is displayed as follows:

```
** SELF TEST ERROR XX ** MODULE ZZZZ
   CONTINUE Y OR N?
```

where: XX is an error code number in the range 81 through 89, described in appendix A. (ZZZZ is an address of a program module, of no significance to the user.) The query Y or N? allows the user to specify whether to continue the self test program by overriding the error condition, or to stop the program. With some error conditions it is possible to complete the self test and then perform limited operations, in which case the operator must enter a capital Y. In response, the operating system will attempt to display the startup message and prompt the operator with D>. If N is entered, the system will make no response.

If the 9508 cannot be used due to the displayed error condition, the following limited attempt to correct the problem can be made:

1. Turn off AC power and disconnect the power cord.

2. Loosen the thumbscrews securing the 9508 top cover (see figure 2-1) and remove the top cover. Check that all printed circuit cards are firmly seated in the universal card cage and that all internal/external cable connections are secure.

3. Reinstall the 9508 top cover, reconnect the AC power cord and press the POWER switch. The system will automatically perform the self-test program. Verify that the test is completed without a malfunction and that the initialization message is displayed as described in the previous paragraph.

4. If the malfunction is still present, contact the Millennium Systems Customer Service Representative via the telephone numbers listed in the preface.

OPERATION

INTRODUCTION

This chapter describes how to operate the 9508 MicroSystem Emulator. The chapter is divided into three sections, as follows: section I describes a complete software or hardware debugging/testing sequence - beginning with startup of the equipment, downloading a program from the host, continuing with setup of test conditions, execution of the program under test, and ending with uploading a corrected and patched program to the host. During this sequence it is always necessary for the operator to: 1) communicate with the host and 2) define test conditions in terms of events and trigger equations that cause breakpoints to occur during program execution. These two subjects are too complex to be covered in context of the overall operating procedure in section I. They are described separately in sections II and III, respectively. The user should be familiar with sections II and III before performing the procedures in section I.

OPERATING MODES

User input commands entered at the console terminal keyboard are displayed on the terminal screen, along with the responses from the 9508 emulator. The 9508 emulator, at any time, is either prompting for an input from the operator (it is said to be at command level), or is executing. At the command level the Executive program in the 9508 emulator is prompting the operator with the D> prompt for the next command input. When the 9508 emulator receives a command (a command mnemonic followed by CR, or a control character) the program routine associated with that command is executed. During execution some command program routines interact with the operator by prompting for data (e.g. the ASM command) and most routines display results. On completion, control returns to the Executive, which again prompts with the D>.

If during execution the operator wishes to abort the command routine, he enters ESC (escape), control returns to Executive, and the emulator again is at command level. Similarily, during execution the space bar can be depressed once to stop any output to the console for display and then a second time to resume.

All command routines, except that of the Go command, execute some 9508 emulator resident utility. Such utilities display or modify emulator or UUT memory, communicate with the host system or the UUT, set up trace or breakpoint conditions for a test run, or in some other way prepare for execution of the program under test. When the Go command is entered, control passes directly to the program under development by the user (the program under test), and its execution is in progress.

During emulation, the program under test may be executed in the system mode or in user mode. In system mode the program is executed entirely out of memory located in the 9508 emulator, using the system clock generated in the emulator and there is no I/O available to the user. In user mode the program is executed with the UUT clock. Execution may be in real-time or in single step mode. In real-time the emulator processor executes continuously, at full clock speed. In single step mode the processor is paused after each program step and its registers are dumped into a buffer and displayed, before execution is resumed.

If execution is in real-time and complex breakpoints were previously set up, the operator had to specify one of four trigger modes. These modes have to do with different ways in which the preconditions for a breakpoint are defined and are described in section III of this chapter.

TEST INTERCONNECTIONS

To perform testing with the 9508 emulator, the emulator option corresponding to the microprocessor type used in the UUT must be installed. Cable interconnections must be made as shown in figure 1-2 (if testing is to be done entirely in the system mode, the emulator probe need not be connected to the UUT) and additional connections may be made as follows. The RTT data probe can be connected to J3 at the rear of the 9508 emulator and the probe clips can be attached at up to 8 different test points in the UUT; external test equipment, such as an oscilloscope or logic analyzer, can be attached to the TRIGGER connectors on the front panel of the 9508 emulator.

Refer also to chapter 2 for more detailed installation and startup procedures.

SECTION I
DEBUGGING/TESTING OPERATIONS

OVERVIEW

The 9508 emulator is used to test and debug users software, users hardware components and modules, as well as perform integrated testing of hardware and software. The users source file must be created at the host software development system and assembled into an executable object file that can be transferred from the host system to the 9508 emulator via the communications link. Thereafter, the 9508 emulator executes independently (internally in its slave processor), while helping the user debug and patch corrections into the object code. During the execution/debug operations the emulator receives commands from the operator and displays test results, status, or memory contents on the console terminal screen. At the conclusion of the debugging operation, the patched object code is uploaded to the host system.

The above sequence involves the following specific steps:

- o Inital startup of 9508 emulator.
- o Selection of the emulation mode - either system or user.
- o Mapping of 9508 emulation memory into UUT memory.
- o Downloading the users program from host computer to the emulator via the emulator-host communications link.
- o Setting up test conditions: defining events, triggers, breakpoint equations, activating either continuous or single step execution mode, activating trace.
- o Execute, interpret the displayed results, and patch in program corrections, or alter hardware.
- o Upload the debugged patched program from the emulator to the host computer via the communications link.

Each of the steps is described in detail in the following paragraphs, with references to sections II and III of this chapter for details on communicating with the host (download, upload, etc.) and breakpoint equation setup procedures.

INITIAL STARTUP OF EMULATOR

The 9508 MicroSystem Emulator can be initialized from a cold start by depressing the front panel POWER switch. Any time thereafter the 9508 can be reinitialized by depressing the RESET pushbutton. Refer to the Initial Startup of System procedure in chapter 2, which describes 9508 startup in more detail, including the self-test diagnostic, system initialization, and error conditions associated with the diagnostic. Use the following procedure to power up the system from a cold start during normal operating conditions:

1. Activate the display terminal power switch.

   a. On the 9501 terminal a bell tone will sound within 1 second to indicate power is on.

   b. After a few seconds, the cursor will appear in the upper left hand corner of the screen. At this time, the operator can adjust the contrast to obtain the desired brilliance of the screen.

2. Depress the front panel POWER switch of the 9508 emulator.

   a. The emulator will perform the self-test diagnostic and initialization process. If no errors are present during startup, a 13-line message is displayed ending with the prompt character, as shown in the example.

   b. If an error condition is encountered during the diagnostic, the following message is displayed:

```
** SELF TEST ERROR ** XX MODULE ZZZZ
   CONTINUE Y OR N?
```

   where: XX is an error code number in the range 81 through 89, described in appendix A. (ZZZZ is a reference to a program module number and is of no significance to the user.) The query Y or N allows the user to specify whether to continue the self-test program by overriding the error condition, or to stop.

   c. With some error conditions it is possible to complete the test and then perform limited operations, in which case the operator must enter a capital Y. In response, the operating system will attempt to display the startup message and prompt the operator with D>. If N is entered, the system will make no response.

```
PROCESSOR-Z80 V1.0
W=0000 X=0000 Y=0000 Z=0000
MAPPED  RAM 1 START=0000  END=1FFF
EVent Address  Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt   TMode  SEL TRIG
  1   = OFF   = OFF  All   XXXX XXXX  1      0 E1        0 MS  T1 IND       T1
  2   = OFF   = OFF  All   XXXX XXXX  2      0 E2        0 MS  T2 IND       T2

BReak T1=Dsbl  T2=Dsbl    Count=    0 MS    Qual=ALL
  LOC  MNEM OPRD/EADDR  A  SZHPNC  BC   DE   HL  IX/IY  SP  R  I  IM12 PCNEXT
  0000 XOR  A           00 000000 0000 0000 0000  0000 0000 00 00  0DD   0000
                        00 000000 0000 0000 0000  0000

REGBRK CONDITIONS:

SYSTEM MODE
D>
```

The system startup message contains 13 lines of information that identify the type of target microprocessor being emulated (in the example display a Z80 microprocessor), summarizes the test setup conditions (events, triggers, breakpoints), and displays the contents of the target microprocessor registers, as well as the first instruction at the beginning of the memory. The content of each message line, starting at the top of the display is as follows:

LINE NO.	DESCRIPTION

EMULATOR TEST SETUP CONDITIONS (Lines 1-8)

1 -- Indicates 9508 is initialized and ready for use. Software version level is specified.

2 -- Indicates microprocessor type that is being emulated (Z80 used in example) and the version of the microprocessor dependent software.

3 -- Indicates settings of the emulator bias registers (W, X, Y and Z).

4 -- Indicates starting and ending addresses for mapped RAM, into which the user can download the program under test from the host computer.

5,6,7-- Indicates current definition of events in terms of an address, data, bus activity, bus bits, and/or the status of trace clips; trigger setting (in terms of a pass count for E1 & E2, and delay count); breakpoint qualification mode (TMode) and the associated key trigger.

8 -- Indicates break and trace conditions as follows: breakpoints that are enabled; current count in general purpose counter; and the type of bus activity to be collected in real-time trace buffer.

CPU DEPENDENT CONDITIONS (LINES 9-13)

9,10,-- Identifies contents of the first location in memory and contents of
11 target microprocessor registers,

where: LOC = memory address (PC contents)
MNEM = mnemonic for XOR (or any other) logical instruction
OPRD/EADDR = operand/effective address
A = contents of accumulator
SZHPNC = flag register bits as follows: sign flag (S)
zero flag (Z); half carry (H); parity/overflow
(P); subtract flag (N); and carry flag (C)
BC = general purpose register pair
DE = general purpose register pair
HL = general purpose register pair
IX/IY = index registers
SP = stack pointer register
R = memory refresh register
I = interrupt vector register
IM12 = interrupt mode (0, 1 or 2) for maskable interrupts
PC NEXT = program counter address for next instruction to be
executed

12 -- Indicates any microprocessor register content conditions that will cause a break (set with REGBrk command).

13 -- Indicates emulator operating mode, set either to system or user mode by EM)ul command.

Note that parts of the above message are also displayed as a result of command entries, or as a result of program execution. For example, all lines, except 1 are displayed in response to the Status command, lines 5 thru 8 are displayed whenever the EVent command is entered, lines 9-11 are displayed whenever a program is executed with trace on, etc.

RESTARTING THE EMULATOR

Emulator operation can be restarted at any time by depressing the RESET pushbutton switch located on the 9508 front panel. This terminates the current processing operation and reinitializes the system. This causes the system to wait until the next command is entered at the console terminal keyboard.

SHUTDOWN OF THE SYSTEM

To shut the system down, first wait for the current processing operations to be completed, so that the system is waiting for an input command from the keyboard (as indicated by the prompt >D). Turn off power with the POWER switches on the 9508 emulator and on the display terminal.

DOWNLOADING USERS PROGRAM FROM HOST

Downloading means the copying of a program that is to be debugged or used for testing UUT hardware. This program is supplied by the host computer (either a Millennium 9520 Software Development System or another software generation system) to a memory which is mapped to either the 9508 emulation memory, or the UUT memory.

Any executable object program may be executed and debugged using the 9508 emulator when the system is appropriately configured with a target microprocessor emulator card and pod. The users program must be in the target microprocessor machine language.

After the download operation is completed, the communications link with the host can be broken and all debugging/testing activities are conducted by the 9508 and its console terminal in a stand-alone mode.

Details of communicating with the host system for transferring data between the host and emulator are described in section II of this chapter.

DEBUGGING/TESTING

In order to gain a sense of how debugging and testing of software and hardware is implemented with actual commands, it is useful to follow a generalized operational sequence, beginning at the point where the startup message is on the screen. This will be followed by an annotated example of a program debugging session, showing all operator entries and displays exactly as they appear on the terminal screen. These examples are presented in the paragraph titled SAMPLES OF TESTING AND DEBUGGING PROCEDURES.

Loading Users Software

The first task is to download the program to be debugged (or be used in debugging the UUT hardware) from the host computer into the 9508 emulator. By selecting the operating mode with the EMul command and activating specific segments of emulation memory with the MAp command, downloading can be either into 9508 system RAM, the UUT memory, or distributed in both. However, as indicated above, the equipment initially comes up in system mode, with 8K mapped to system RAM location 0 thru 1FFF. As long as the program resides within those boundaries, it is downloaded into the 9508 system RAM and the UUT hardware does not have to exist. Downloading is accomplished using the RHex command, as described in section II of this chapter. Following downloading, any portion of the program can also be transferred into the UUT memory with the MOve command and then executed in the environment of the UUT processor by changing to user mode with the EMul command.

Setting Up Test Conditions

Next, the test conditions for program execution - how it is to be traced and what will cause it to halt - are set up (prior to entering the GO command to run the program). This presents many different options associated with the various hardware and software based real-time and single step trace features of the 9508 emulator and breakpoint capabilities, described in section III of this chapter. However, it is useful to keep in mind that all are tools designed to give the programmer or hardware designer the ability to monitor program execution or signal status in hardware, stop execution, trap the surrounding conditions, and examine them in order to locate the source of a program or hardware bug.

In the most elementary sense, the Go command determines whether the program is executed continously (in real-time) or single stepped. The Until parameter by itself causes real-time execution, but either Step or Trace causes single stepping. (Single stepping is also caused by use of the REGBrk command.)

If execution is in real-time, the message EXECUTING IN REAL-TIME appears after the GO command is given. Emulator processor bus data and RTT data probe signals are collected into the 128-entry circular real-time trace buffer on the RTT card. Execution continues until the specified address is accessed, another kind of breakpoint is encountered, or the ESC key is depressed. Once execution is stopped, contents of the RTT buffer can be displayed by entering the DRT command.

If execution is single stepped and trace is on, the processor is paused after each instruction and its registers are displayed. A decision is made as to whether execution is resumed or breakpoint conditions are satisfied and execution is halted. The register contents of every execution step are displayed on the console terminal screen, along with a mnemonic of the instruction just executed. This execute/display sequence continues until the specified number of instructions are completed, until another kind of breakpoint is encountered, or ESC is entered. Note that during single stepping the RTT buffer also collects information for later viewing, just as during real-time execution. Regardless of whether execution is in real-time or single stepped, the Qual command may be used to qualify the type of information to be collected into the RTT buffer (e.g., fetch cycles only, memory accesses only, etc.).

Breakpoints can be set in many increasingly involved ways. In addition to the simple breakpoints already mentioned with the Go command the REGBrk command can be used to define the contents of a particular processor register as a precondition for breaking.

The complex breakpoints (refer to section III) can be set by specifying events E1 and E2, pass counts, and delays, which together define the trigger equations T1 and T2. Then, the TMode command can be used to select one of the trigger modes, which further stipulates how events and trigger equations relate to each other and combine to initiate a break. (Events, triggers, and trigger modes are described in section III of this chapter.)

As an example, a typical operation might consist of the following sequence that completely defines the information to be gathered during execution and sets a complex breakpoint:

If signals are to be monitored in the UUT circuitry - either for storage in the RTT buffer or for use as breakpoint preconditions - connect the RTT data probe clips to the UUT and set switches on the probe pod as explained in section III.

QUAL	Qualifies what information will be gathered into the RTT buffer
EVent	Defines event E1 and/or E2
TRig	Defines event pass count and delay count as preconditions for T1 and T2
Count	Instructs the general purpose counter what it is to count, thus specifying what the delay entered with TRig command is based on. This completes the definition of T1 and T2 equations.

The commands used to control the Real-Time Trace card are as follows:

TMode	Defines the trigger mode and completes setting the test conditions. The Status command may now be used to display the breakpoint conditions on the screen and reverify them. Note also that these test conditions remain set until they are either changed or erased by reinitializing the 9508 emulator with the RESET pushbutton, or when the power is removed. (Refer to section III for a detailed description of the trigger modes and their applications.)
BReak	Enables a break to occur on the T1 and/or T2 defined with above commands.

Executing and Correcting Hardware/Software Faults

Now, the user is ready to enter the Go command and start program execution. If the trace is on (Trace parameter of the Go command) after each instruction execution, the processor register contents are displayed. Updating the terminal screen (and thereby execution of the program) can be stopped and restarted at any time by depressing the space bar on the keyboard. Execution will stop when a breakpoint is encountered, or the terminating address (or required number of steps) is reached.

Assuming the program halts on a breakpoint, the user then proceeds to examine various conditions. In addition to the processor registers, the user may use the DRt command to view contents of the real-time trace buffer. The user can use the Dism command to see a sequence of opcodes, starting at a particular PC address, or the Dump or Exam command to view data at an address. If the system was executing in the user mode, he may want to inspect the contents of an I/O port in the UUT, by using the Port command.

Once the user decides on corrections to be made, the Asm command is used to patch in revised opcodes. The write feature of Exam command is used to alter data in memory, or the write feature of Port command to write to a port. Before entering the Go command to run the patched program segment, if the user needs to alter processor registers, the REG command can be used to access and write into any processor register.

This process of executing with the Go command, evaluating results, and making corrections will be repeated as many times as necessary. In between, breakpoint definitions may be changed in order to trap bugs that take on a new appearance as progressive corrections are accomplished. If the user finds that after every break, he is performing the same functions over and over again (for example, entering the DRt, Dism, Dump, or Exam command), the Onbrk command may be used to specify a list of commands that are automatically executed on program halt.

The above debugging process is equally applicable to software as well as hardware debugging, except that during hardware debugging the test clip status information will be used to set up trigger equations. The equation setup, as well as breakpoint qualification mode selection, may also differ in other ways.

Uploading The Users Program (refer to section II for expanded information)

Once debugging is completed the patched program can be uploaded to the host system by using the WHex command as described in section II of this chapter. The corrected program will then be stored on a disk.

SAMPLES OF TESTING AND DEBUGGING PROCEDURES

The remainder of this section contains sample procedures that demonstrate how specific software and hardware testing and debugging problems are solved with the 9508 emulator. Software is tested and debugged using real-time trace, the different breakpoints, and single-step trace capabilities of the 9508 emulator. A circuit in the UUT is tested by executing a program that stimulates the circuit and then observing whether the expected signals occur. This is done by connecting the RTT data probe to key points in the circuit and then collecting signals that occur during execution into the RTT buffer for later display and evaluation.

In all procedures it is assumed that all test conditions set up previously are cleared before starting. The following address or data parameter notations are used in the procedures:

addr - an effective or absolute hexadecimal address in the range 0-FFFF. If several different addresses are used, they are designated addr1, addr2, etc.

saddr - starting address, an effective or absolute hexadecimal address in the range 0-FFFF.

laddr and uaddr - an effective or absolute hexadecimal address pair, denoting the lower and upper address of a memory segment. Either can have any value between 0 and FFFF, but laddr must be less than or equal to uaddr.

nnn - decimal data. The number of characters indicates the range: n=0-9; nn=0-99; nnn=0-999, etc.

cccccccc - eight characters representing the states of the eight test clips of the RTT data probe. Each character can be 1 (logic high), 0 (logic low), or X (don't care).

Breaking when the program executes a designated instruction.

Description: This procedure is used to set a complex breakpoint and thus stop execution. The use of a complex breakpoint will not slow down the emulator processor, whereas single-step trace will. The program executes until the designated instruction is fetched, at which time control of the system is returned to the user (the emulator is at command level).

Procedure:

```
D>EVent 1 Clear A=addr B=F
D>BReak T1
D>Go saddr
```

Explanation: The EVent command line establishes the preconditions for event E1. The Clear parameter erases any previous conditions designated for event E1 before new conditions are applied. The A=addr and B=F set event E1 to trigger on any instruction fetch from addr. The BReak command line specifies that the break will occur when the conditions of event E1 or E2 are satisfied.

Breaking on the Specified Pass Through a Loop

Description: This procedure is used to stop execution of a program when the designated instruction is executed the specified number of times.

Procedure:
```
D>EVent 1 Clear A=addr B=F
D>TRig 1 P=nnnnn (maximum 65535)
D>BReak T1
D>Go saddr
```

Explanation: The EVent command line is similar to that given in the previous procedure. However, the TRIG command is used to specify a passcount. Specifying this parameter will prevent the event E1 trigger T1 from being generated until the conditions of the event E1 have been satisfied nnnnn times. If the instruction at addr is executed fewer than nnnnn times, no break will occur. In that case, either the program will run to termination, or the operator must press the ESC key to regain control of the system.

Breaking When Execution Proceeds Outside a Designated Range

Description: This procedure is used to stop execution of a program when the instruction to be executed lies outside the designated address range.

Procedure:
```
D>Qual Fetch
D>EVent 1 Clear A=<laddr B=F
D>Event 2 Clear A=>uaddr B=F
D>TMode Ind
D>BReak Both
D>Go Saddr
```

Explanation: Qual Fetch specifies that only instruction fetch cycles will be stored in RTT buffer. The two EVent command lines select boundaries of the memory range. The laddr address of the EVent command must be less than the uaddr address. TMode Ind specifies that a break will occur if either event E1 or E2 is generated.

Breaking When the Program Writes in a Designated Memory Area

Description: This procedure is used to stop execution when an attempt is made to alter the contents of a designated memory area.

Procedure:

```
D>Qual All
D>EVent 1 Clear A=>laddr B=MW
D>EVent 2 Clear A=<uaddr B=MW
D>TMode E12
D>BReak T1
D>Go saddr
```

Explanation: Qual All specifies that all types of bus cycles will be stored in the RTT buffer. The EVent 1 command line sets the conditions of event E1, which is being used to restrict the break to only those instructions where a memory write into memory locations higher than the designated address is attempted. The EVent 2 command line requires that the address be less than the selected upper bounds. The laddr must not exceed the uaddr. The TMode E12 command specifies that a break will be caused only if the conditions of event E1 (the address and bus cycle type) as well as conditions of event E2 are simultaneously satisfied.

Saving an Execution Trace Record Without Interrupting the Program

Description: This procedure is used to record and display up to 128 instructions prior to encountering EVent 1 (EV1). When EV1 is executed operation will not terminate until the breaking address.

Procedure:

```
D>Qual Fetch
D>EVent 1 Clear A=addr1 B=F
D>EVent 2 Clear A=addr2 B=F
D>TMode Frz
D>BReak T2
D>Go saddr
(wait for program to run to completion)
D>DRt
```

Explanation: Qual Fetch specifies that all instruction fetch cycles will be stored in the RTT buffer. The EVent command lines describe the conditions at which events E1 and E2 will be generated. The TMode Frz disables the RTT buffer from storing information at event E1, and causes a break at event E2. The DRt command initiates the display of console information from the RTT buffer on the console terminal.

Recording Instructions Executed Before and After a Designated Instruction

Description: This procedure is used to record and display 64 instructions executed before and 64 instructions after the execution of a designated instruction. Execution stops when all requested information has been obtained.

Procedure:

```
D>Qual Fetch
D>Count Rtt
D>TRig 1 D=64
D>EVent 1 Clear A=addr B=F
D>BReak T1
D>Go saddr
(wait for program to run to completion)
D>DRt
```

Explanation: Qual Fetch specifies that only instruction fetch cycles will be stored in the RTT buffer. Count RTT commands the general purpose counter to generate a signal each time the RTT buffer stores one word of information; in this case, a store will occur on each instruction fetch. The EVent 1 command line, together with TRig 1 D=64, sets the preconditions of event E1 to generate a trigger 64 counts after the instruction at the designated address has been executed; the rate of counting is determined by the general purpose counter's clock signal (RTT buffer store clocks). DRt displays the contents of the RTT buffer when all the selected information has been gathered.

Recording Instructions Executed After a Designated Instruction

Description: This procedure is used to record and display up to 128 instructions that were executed after the designated instruction. Execution stops after the requested information has been collected.

Procedure:

```
D>Qual Fetch
D>Count Rtt
D>Trig 1 D=128
D>EVent 1 Clear A=addr B=F
D>BReak T1
D>Go saddr
(wait for program to run to completion)
D>DRt
```

Explanation: This procedure is similar to the previous procedure. The D=64 parameter has been changed to D=128; this change causes the break to be generated 128 instruction fetches after the designated instruction, rather than 64 instruction fetches after.

Determining the Execution Time of a Program Segment

Description: This procedure is used to calculate the elapsed time between the execution of two designated instructions. Program execution terminates when the second instruction is executed. Time can be calculated in milliseconds or microseconds.

Procedure:

```
D>Count Ms (or Us)
D>EVent 1 Clear A=addr1 B=F
D>EVent 2 Clear A=addr2 B=F
D>TMode Arm
D>BReak T2
D>Go saddr
(wait for program segment to run to completion)
D>Count
```

Explanation: The Count command with its parameter specifies the measurement units. The two EVent command lines specify the beginning and ending of the program segment. The TMode Arm command enables the general purpose counter at event E1 and disables it at event E2, and causes a break at event E2. The final Count command displays the value stored in the general purpose counter.

Note: If the output, in response to the Count command, is 65,534, the counter has overflowed.

Measuring the Interval Between Probe Events

Description: This procedure is used to measure the time interval between two events in the UUT. The program terminates when the conditions of the second probe event are satisfied.

Procedure:

```
(Attach the RTT data probe clips to the desired signal lines
in the UUT.)
D>Count Ms (or Us)
D>EVent 1 Clear E=cccccccc
D>EVent 2 Clear E=cccccccc
D>TMode Arm
D>BReak T2
D>Go saddr
(wait for program execution to complete)
D>Count
```

Explanation: This procedure differs from the previous procedure only in one respect: the conditions for starting and stopping the counter are defined in terms of the RTT data probe test clip values instead of the designated instruction fetches.

Breaking on a Probe Event

Description: This procedure uses the RTT data probe inputs to stop program execution when the conditions of a probe event are satisfied.

Procedure: (Attach the RTT data probe clips to the desired signal lines in the UUT.)

```
D>EVent 1 Clear E=cccccccc
D>BReak T1
D>Go saddr
```

Explanation: The EVent command line defines event E1 in terms of the probe clip values. The BReak T1 command causes a break to occur when the specified probe clip values are satisfied.

SECTION II
COMMUNICATING WITH THE HOST

OVERVIEW OF THE COMMUNICATIONS LINK

During installation the communications link between the 9508 emulator and the host system can be connected in any one of several different ways. There are two possible physical interconnections (as shown in figures 2-5 and 2-6) and several different ways in which these interconnections can be utilized. Which physical interconnection and what communication procedure or mode is used depends on the hardware and software capabilities of the host system. The 9508 hardware and software is designed for maximum flexibility in implementing different ways of communicating, so that the 9508 emulator can interface to various hosts other than the Millennium 9520 Software Development System. The following paragraphs explain the link and the different ways of communicating.

Communicating with the host system is performed for two major reasons:

1) transfer of program material between the 9508 emulator and the host, and

2) perform miscellaneous operations in the host, using the 9508 emulator console terminal with the 9508 in a transparent or pass through mode.

Transfer of program material is done to download a program under development from the host system into the 9508 emulator, in preparation for debugging it, or to upload a corrected program from the 9508 emulator to the host (refer to Section I of this chapter for a description of the context in which the transfers are made).

The miscellaneous operations in the host can perform are, program editing, file manipulation (copying, moving or converting data formats, etc.), and any other activity that is usually carried out from a command terminal.

The following paragraphs describe three different cases of the download/upload operations, and a separate mode for the other interactive command operations. A particular communications link can support either one or several cases of the upload/download operations and/or the interactive communications mode.

HARDWARE AND SOFTWARE COMPONENTS OF THE LINK

Physically, the 9508 emulator is connected through the RS-232 interface to the host, either

1) as a console device (to a console port of the host, see figure 2-5), in which case both commands, as well as data can be transferred across the interface in an interactive manner, or

2) as an I/O device (to a peripheral or I/O port, see figure 2-6), in which case only data can be transferred, but no commands.

Which of the two physical interconnections is used on a particular installation depends on the available ports on the host and its software architecture.

Other characteristics of the link, such as the transmission mode (full or half duplex), parity checking, baud rate, handshake protocol (RS-232 DSR/DTR or ACK/NAK), and the data format (Binary or TekHex), each has an initial value defined in PROM, but can be changed by the Linkdef command at any time (refer to chapter 4).

In software, the 9508 emulator supports three different cases of download/upload operations and an interactive communications mode (also called the terminal pass through mode). A particular way of uploading/downloading and the interactive communications mode becomes available to the user depending on which of the two physical interconnections the interface uses. Thus, the complete communications link for any installation is defined by 1) the particular physical interconnection and 2) the ways in which the host has been programmed to communicate with the 9508 emulator.

In the 9508 emulator software architecture the three ways of download/upload are carried out with the WHex and RHex commands. For example, to download a program into the emulator or UUT the RHex command is entered, together with one of the following three parameters:

```
TRans
'String
TErm
```

Likewise, to upload a program to the host, the WHex command is entered, together with any one of the same three parameters.

To enter the interactive communications mode, the TERMINAL command is used. This command causes the emulator console terminal to be connected directly to the host, functionally switching the 9508 emulator entirely out of the channel.

The three cases of upload/download and the interactive communications mode are described separately. Refer to chapter 6 for a description of the interaction between program modules in the host and the emulator during dowloading and uploading.

THREE CASES OF DOWNLOAD/UPLOAD OPERATIONS

The following paragraphs describe each of three ways (cases) in which the 9508 emulator user can download or upload program files. The purpose of each case is explained first, followed by a typical procedure demonstrating how it is used for download/upload tasks in the host system. The procedure in each case is based on the host being a Millennium 9520 Software Development System with the MP/M operating system, but explanatory comments for expanding to other host systems are included. Depending on how it was initially set up, your installation may support one or more of these cases.

Each of Cases 1, 2, and 3 differ in the sequence of procedural steps and options available to the operator, but the following basic elements are required to perform any transfer and thus are common to all cases:

1. The transfer program (Rhex or Whex) is activated in the 9508.
2. The transfer program is activated in the host (for 9520 system the programs are DOWNLOAD and UPLOAD).
3. The 9508 and host programs are synchronized to make sure that both are ready to begin the transfer (this step is optional).
4. The program data is transferred.

Download/Upload - Case 1

Case 1 of downloading or uploading is the simplest and also the most limited. In Case 1 the 9508 emulator is connected to the host as an I/O device refer to (figure 2-6). That is, the emulator functions only as a peripheral I/O device of the host, rather than as a console. The emulator and host do not exchange any commands, instead, the emulator must always be ready to receive when the host begins to download data blocks and conversely, the host must be ready when the emulator begins to upload data. Case 1 is not intended for use with the 9520 Development System, but instead is intended for host systems that have a bidirectional I/O port available for connecting the 9508 emulator.

To initiate a download or upload, the user enters the RHex or WHex command, together with the TRans parameter. In the case of downloading, the host system must now be commanded from its own master console to activate a download utility. From there on, Download executes in the host, synchronizes with the Rhex command routine in the emulator, and sends data records to the emulator. The data records are interpreted by the Rhex routine and written to memory. A display, as described in the following example starting with the SYNC OK message, appears on the emulator console.

```
SYNC OK                          (appears only if Nosync was defaulted in
TRANSFER MODE                     entering the RHex command)
ADDR=XXXX
ADDR=YYYY
    |
    |                            (Where XXXX through ZZZZ are starting
    |                             addresses of each data record transferred)
    |
ADDR=ZZZZ

PC=WWWW                          (where WWWW is the address at which execution
                                  of the downloaded program is to begin; stored
                                  in emulator processor program counter)
```

In case of uploading, the Upload utility in the host must be activated first, so that the host is ready to receive data. The WHex command is entered with the optional Nosync parameter, addresses, and the TRans parameter at the emulator console terminal. This activates the Whex command routine, synchronizes the programs, and starts the transfer of data records. Again, any displays that may appear during the operation will be displayed at the emulator console and may also appear on the host master console.

Download/Upload - Case 2

Case 2 of downloading or uploading allows the user to enter a single command to the host, to activate the host and start the upload or download process. In Case 2 the 9508 emulator must be connected to the host in the console configuration (refer to figure 2-5).

Case 2 is for host systems whose architecture allows the 9508 emulator to invoke the Download or Upload utilities with one command, without any preliminary steps described under Case 3. Thus, the Case 2 procedure is a simple way to initiate a download/upload operation, but it does not allow the user to do any program editing or other operations in the host. A Case 2 interconnection can be set up with the 9520 Software Development System.

To do a download or upload in accordance with the Case 2 procedure, enter the RHex or WHEx command, followed by name of the file to be transferred, preceded by a single quote. For example:

```
RHex 'Filename
```

This causes the Rhex command routine to be activated and send a download command to the host. The command is similar to the command entered by the operator in the Case 3 procedure (e.g. DOWNLOAD Filename), except in this case the command is constructed and sent to the host automatically by the Rhex routine. The host system activates the Download utility module upon receipt of the command. The download process continues as outlined in the Case 3 procedure. The Download software responds by transmitting its version number and a Control \ (backslash) default character. The Rhex routine synchronizes with the Download utility and the 9508 emulator enters the transfer mode so the data records are downloaded. When the downloading is completed, the emulator returns to the command level. (Steps 1, 2, and 6 through 9 of the Case 3 procedure are similar in this procedure, except in step 2 the Terminal routine is not called up and a DOWNLOAD Filename command is sent out to the host instead.)

The procedure for uploading is similar to the download procedure, except the Whex routine is activated and the command UPLOAD Filename is sent to the host to invoke the Upload utility. The output looks the same as in the previous example.

Download/Upload - Case 3

To carry out downloading or uploading in accordance with the Case 3 procedure, the 9508 emulator is connected to the host in the console configuration (refer to figure 2-5). Case 3 can be used with the 9520 Software Development System, but can also be used with other host systems through which the 9508 emulator, with its console terminal, can communicate with the host as any other general purpose command terminal. For example, it is suited for systems that require any preliminary operations (e.g. data format conversion) before a download or upload command from the 9508 starts the actual data transfer. As another example, if the host is a timesharing system, the host protocol may require that the user first communicate with the network interface, log onto the operating system, and only then activate the Upload or Download program. Once logged on, the user can also perform program editing, file manipulation, or any other activity on the host prior to starting the download process. During this time the communication is directly between the emulator console terminal and the host system, just as during the interactive communications mode described below (i.e. the emulator is transparent).

To use the Case 3 procedure, the user enters either the RHex or WHex command, together with the TErm parameter. The following is a typical procedure for communicating with the host, starting at the point where the D> prompt is on the screen and the 9508 emulator is at the command level. The procedure ends with a download or upload operation.

1. Enter the Linkdef command to specify all characteristics of the communications link - default or define its parameters, as required (see chapter 4 for a definition of all parameters of the Linkdef command). In case the host is a 9520 development system, the initial values (after power on or system reset) are fully compatible and use of the Linkdef command is not necessary. For other host systems certain parameters may need to be redefined.

   Note: If the host is a 9520 and its DOWNLOAD program is being used, the command must be RHEX TERM. If the host is other than a 9520, the command parameters must be determined for each host system.

2. If you plan to terminate your activities with a download (see step 7 for upload), enter the RHex command, along with the Nosync parameter (optional), and the TErm parameter (required). This invokes the Rhex command routine in the emulator. The Rhex routine calls up the Terminal routine in the emulator, which then sends a CR to the host, thus setting up the communications link to operate in the same way as in the interactive communications mode.

3. The host now responds with its prompt and communication is opened directly between the emulator console terminal and the host system (its operating system, a data network interface, or any other part of its architecture that responded to the initial CR signal); the emulator passes all signals directly through and appears completely transparent to the user. If the host is a 9520, the operating system responds with the prompt 0A> (or 1A>).

4. Enter commands and/or data at the emulator console keyboard to perform all program editing, assembly, file manipulation, or any other operations in the host, in accordance with its operating protocol. (For the 9520 refer to the 9520 Software Development System Users Manual, Publication No. 87000064)

5. When all tasks in the host have been completed, and you are ready to download a program file, enter:

   DOWNLOAD Filename

   (a host other than 9520 may need a different command)

6. This activates the Download utility in the host and Download responds with:

Vers X.X (X.X = Download utility revision level.)

and sends a Control \ default character. The 9508 recognizes the Control \ and returns program control from the Terminal routine to Rhex routine. If synchronization was specified with the entry of the RHex command in step 2, the Rhex routine repeatedly sends a S character (and CR) to the host until it receives back a T character (and CR). At this time the 9508 emulator displays SYNC OK, enters the Transfer mode, and is ready to receive the first data block from the host. When a data block is received, the Rhex routine performs a series of error checks (refer to chapter 6 for a list of the errors monitored). If no errors are found, it may return an ACK character to the host (depending on the protocol in use, see chapter 6). The Download program continues executing and sends data blocks until the entire transfer is complete. During the above sequence the following message lines are constructed by the Rhex routine and displayed on the 9508 emulator console screen:

```
EXIT TERMINAL MODE
SYNC OK                          (appears only if Nosync was defaulted in
TRANSFER MODE                     entering the RHex command)
ADDR=XXXX
ADDR=YYYY                        (Where XXXX through ZZZZ are starting
   |                              addresses of each data record transferred)
   |
   |
ADDR=ZZZZ

PC=WWWW                          (where WWWW is the address at which execution
                                  of the downloaded program is to begin; stored
EXIT TRANSFER MODE                in emulator processor program counter)
D>
```

At this time the entire operation is complete and the 9508 emulator is back at command level.

Note: If the host is a 9520 and its UPLOAD program is being used, the command must be WHEX Start = execution Start-memory End-memory TErm. It the host is other than a 9520, the command parameters must be determined by each host system.

7. If the required task is an upload, the user would enter WHEX, the Nosync parameter (optional), the execution start address, the memory start address, the memory end address and the TErm parameters (see chapter 4 for definition of all parameters of the WHex command). This invokes the Whex routine, and sets up the interactive communications link as described in step 2.

8. When all tasks in the host have been completed and the user is ready to begin uploading, enter:

   UPLOAD Filename

   (a host other than 9520 may need a different message)

9. This causes the Upload utility in host to be invoked and it responds with:

   Vers X.X (X.X = Upload utility revision level)

   and sends a Control \ default character. From there on the interaction between the emulator and host is similar to the procedure in step 6 except data blocks are transferred from the 9508 emulator to the host; the Whex routine performs error checking and all messages displayed on the console screen are as described in step 6. When the upload is completed, the 9508 emulator switches back to command level.

Interactive Communications Mode

For the interactive communications mode (terminal pass through) the 9508 emulator is connected to the host in the console configuration (refer to figure 2-5). This mode does not have any direct use for downloading or uploading (it is not possible to download or upload using this mode), instead, the 9508 emulator console terminal becomes a general purpose command entry terminal for the host. The 9508 emulator itself is completely transparent, in that it simply passes all characters in both directions, without acting on them. The only exception occurs when the 9508 emulator detects a Control \ from the console or from the host. This causes the 9508 emulator to terminate the interactive communications mode and return to the command level.

To activate the interactive communications mode, enter the TERMINAL command on the 9508 emulator console keyboard. To terminate the mode, enter a Control \ either on the 9508 emulator console or from the host to the 9508 emulator. Note that the DOWNLOAD and UPLOAD commands must not be sent to the host in this mode, because either one will cause a Control \ to be returned to the emulator and force the 9508 emulator to return to the command level. (In addition, the Download or Upload utility will be activated in the host and it will attempt to communicate with the 9508 emulator.)

SECTION III
SETTING UP TEST CONDITIONS

OVERVIEW

When setting up test conditions for an emulation run, the user must first make a basic choice - whether to execute the program under test in real-time or single step mode. By making this choice, the user also decides which of the data generated during the execution will be monitored and available for viewing. If execution is to be in real-time, only transactions on the emulator bus and signal status from the RTT data probe are monitored and recorded. In single step mode, the contents of the emulator processor registers can also be monitored and viewed by the user. If the program under test is lengthy (the number of instructions is in the order of a thousand or more), another decision must be made - during which segment of the execution should the emulator bus data be saved for later viewing (since only 128 separate bus transactions can be saved).

The 9508 emulator hardware/software facilities for monitoring and gathering the data consist of:

- o Real-time Trace (RTT)
- o RTT Data Probe
- o Single Step Trace
- o General Purpose Counter

These facilities, with instructions on how to use them, are described at the beginning of this section.

The 9508 emulator has the capability to perform two different breakpoints:

- o Simple Breakpoints
- o Complex Breakpoints

Both are specified by the operator (i.e. their preconditions are defined) on the basis of data made available from the facilities listed above. The second part of this section describes breakpoints, how to set them, and how to utilize them in typical debugging and testing applications.

REAL-TIME TRACE

Regardless of whether the emulator processor is executing in real-time or single step, the real-time trace (RTT, also referred as hardware trace) always monitors the execution and gathers data. Real-time trace gathers data from two basic sources - the emulator processor bus and the RTT data probe clips. Whenever the emulator processor is executing, whether in real-time or single step, data from both sources is continuously monitored (i.e. once every processor cycle) and is available for storage in the 128-entry circular buffer.

The specific data monitored is as follows:

1. Status of the 16-address bus lines.

2. Status of the 8-data bus lines.

3. Type of processor cycle being executed: read, write, memory, I/O, or instruction fetch.

4. Status of the 8 RTT data probe clips.

Each reading of the data constitutes one word that can be stored in the 128-entry buffer.

Data can be read and stored either every processor cycle, or only during specific kinds of cycles. This is determined by the operator, using the Qual command. For example, the operator can qualify that data is stored only during I/O cycles, fetch cycles, or memory read cycles. (Refer to chapter 4 for a complete listing of the Qual command parameters.) Thus, the operator sets up conditions and the RTT takes signal status snapshots of the buffer during every processor cycle that has been qualified. The 128-entry buffer is filled and then writing resumes at the beginning of the buffer, so that the last 128-entries are always in storage.

To recall data out of the RTT buffer the operator uses the DRt command and the buffer contents are displayed entry-by-entry on the console terminal screen. A typical trace line display is as follows:

```
LOC    DATA   BUS   7 CLIPS 0   INSTRUCTION              DATA
0014    7A    MRF   0100 0110   LD   A,D
```

In addition to collecting data, a second major usage of the RTT card is to implement breakpoints - both simple and complex.

All hardware used to implement the data gathering are located on the RTT circuit card (and the RTT data probe). The QUAL command is used to define the types of data to be gathered. The DRt command is used to display the contents of the RTT buffer. Operator setup of the real-time trace data gathering is self explanatory from the descriptions of the Qual and DRt commands in chapter 4 and the description of the RTT data probe operation in the following paragraph.

RTT Data Probe

The active data acquisition probe with its ten test clips (described in chapter 1) provides the data gathering capability in prototype hardware (the UUT). Eight of the clips attach to test points, one is the ground clip and the tenth may be connected to an external clock source. The data is stored along with the emulator bus transaction data and displayed with the DRt command. The test clip information can also be used in the definition of the events used in constructing the breakpoint equations.

Three switches, that determine the exact manner in which data is sampled, are located on the external test clip pod (figure 3-1). They allow the user to select synchronous or asynchronous clocking of the test clip data and enable a transient (glitch) capturing latch on the test clip lead. Selecting the synchronous mode (switch 1 on the test clip pod) forces data on test clips 0-7 to be captured and stored on the trailing edge of the emulator cycle clock. Asynchronous clock selection (switch 2) allows the user to capture data on either edge of an externally supplied clock applied to the CLK clip of the RTT probe. The last externally clocked state of the test clips prior to the emulator clock is stored in the RTT buffer. The TCO latch (switch 3) can be used in either synchronous or asynchronous mode. When the switch is on (latch enabled), the clock edge used for sampling arms a latch so that any transitions occurring before the next clock edge are captured. If a transition does occur, the latch is set and sampled at the next clock edge. The latch mode is useful for capturing system pulses that happen at intervals shorter than the period of the clock being used for data collection.

SINGLE STEP TRACE

Single step trace (also referred to as software trace) monitors the execution and gathers data only when the emulator processor is executing in single step mode and when the trace is specifically enabled by the operator.

Whenever the emulator processor is executing in single step mode, the processor is paused after each program instruction and contents of the processor registers are displayed. Usually, when execution is in single step mode, single-step trace is also turned on (with the Trace parameter of the Go command, see chapter 4) and either every register dumps or selected register dumps are output to the console terminal and displayed on the screen. The opcode is automatically disassembled and also displayed. The following is a typical trace line display:

```
LOC  MNEM OPRD/EADDR  A   SZHPNC   BC    DE    HL   IX/IV  SP   R  I  IM12  PCNEXT
0000 LD   A,00        00  000010  E39B  E3E3 0000  0000 0208 4A 00  0DD    0002
                      EB  100001  0000  0000 0000  0000
```

(the above example of a trace line is the output of a Z80 target microprocessor)

In addition to being displayed, the register contents can be used to implement the simple breakpoints.

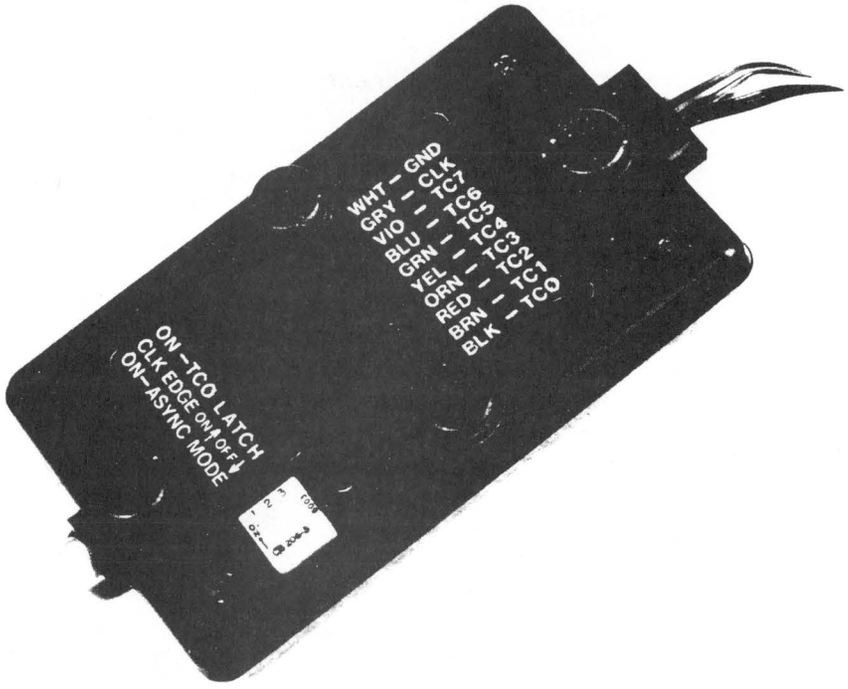


Figure 3-1. RTT Data Probe Pod

GENERAL PURPOSE COUNTER

The general purpose counter is active whenever the emulator processor is executing and, at the option of the operator, can be used for a variety of purposes. For example, during real-time execution, the counter can be used to measure the program execution time (in milliseconds or microseconds), or count the number of times a particular event occurs on the emulator bus. During either real-time or single step execution the counter can record all bus transactions, or only those qualified by the operator, count processor clock cycles, or record stores into the RTT buffer. The counter can count up to 65,534.

The contents of the counter can be displayed at any time by entering the Count or Status command. Contents of the counter can also be used as a precondition for a complex breakpoint, as described later in this section. Note: if the counter overflows its count remains at 65,534.

BREAKPOINTS

In the 9508 emulator four different kinds of breakpoints can be implemented. The processor can be instructed to:

1. Start executing at a memory address and continue until another address is accessed.

2. Start executing at a memory address and continue for a specific number of program steps.

3. Break when the contents of a one or more registers in the processor are at specific values.

4. Break on either of two triggers that represent specific user defined events. (A breakpoint can also be specified to occur on some combination of the triggers, as described below.)

In steps 2 or 3, the 9508 emulator will execute in the single-step mode. This means that the processor bus data and RTT probe data will be stored in the 128-entry buffer. In 1 and 2, the processor registers will be dumped following each program instruction and displayed.

The first three breakpoint types are set with the Go and REGBrk commands and their preconditions are few and straightforward. They are referred to as simple breakpoints. The last breakpoint type can be specified to occur as a result of many different preconditions and thus gives the operator more options - it is referred to as a complex breakpoint. A complex breakpoint can be set on either of the two triggers mentioned in 4.

It is important to note that whenever REGBRK conditions exist, or whenever the STEP or TRACE parameters are specified in the GO command, the UUT program will not be executed in real-time.

Up to four independent breakpoints can be set at any one time - two simple and two complex breakpoints. The procedures for setting both simple and complex breakpoints are explained in the following paragraphs. A reference list of commands used in setting up the breakpoints is as follows (the use of each command is explained in the remaining part of this section and also in chapter 4):

Go	Used to define memory address where emulation execution is to start, set simple breakpoints, single-step trace and cause execution to begin.
Count	Defines what the general purpose counter is to count.
Qual	Selects type of emulator bus data to be stored into the RTT buffer.
REGBrk	Used to define a simple breakpoint on basis of the contents of the emulator processor registers.
EVent	Defines events in complex breakpoint equations.
TRig	Sets pass count and delay count preconditions in the complex breakpoint equations.
TMode	Invokes one of the trigger modes, thus defining the relationship of trigger T1/trigger T2 and event E1/event E2.
BReak	Enables trigger T1 and/or T2 to activate a complex breakpoint.
ONbrk	Used to specify a list of commands to be executed automatically on any breakpoint halt.

Setting Simple Breakpoints.

To set any one of the simple breakpoints the operator needs to use only the Go command, or the REGBrk command if the break is to occur on the specific contents of one or more emulator processor registers. The following is a general outline for a procedure to set each of the three types of simple breakpoints listed in the previous paragraph.

1. If a break is to occur on the contents of one or more registers in the emulator processor (the third type of breakpoint) use the REGBrk command as described in chapter 4. Use the Go command to specify the program execution starting address (with saddr1 parameter), the tracing of execution steps (with Trace parameter, see chapter 4), and start the emulation. The program will be executed in single step mode.

2. To set the first type of simple breakpoint listed in the previous paragraph, set the starting address with saddr1 of the Go command and then specify the Until saddr2 parameter. Use the Trace parameter with its associated ALL, Jmp, and From saddr3 to saddr4 parameters to define what program instructions will be traced. Note that if no Trace or Step is specified, the program will be executed in real-time, but if Trace or Step is set execution will be single step.

3. Use either the Step [nnnnn] parameter or Count nnnn parameter of the Go command to set the second type of simple breakpoint. Use the Trace parameter with its associated ALL, Jmp, and From saddr3 to saddr4 (if Count nnnn was not used) parameters to define which program instructions will be traced.

NOTE

Breakpoints can be set, (see steps 2 and 3) even if a breakpoint was already set with REGBrk in step 1 (in which case two mutually independent breakpoints are now in effect).

Complex Breakpoints - Overview.

In the 9508 emulator there are two possible complex breakpoint triggers. They are designated T1 and T2. Either can be used to initiate a breakpoint and thus pause the emulator processor. For a breakpoint halt to occur, the preconditions for the selected trigger must be defined by the operator and the operator must also specifically enable the trigger. During execution, when the conditions for the selected trigger are satisfied it may also cause a breakpoint halt by itself (presuming the trigger is enabled). It can be set up to act in concert with the other trigger. The various ways in which the two triggers can be conditioned to act together are referred to as the trigger modes.

The preconditions that must be defined by the operator for each trigger are events, pass counts, and delay counts. These preconditions can be expressed as terms in an equation, which, when satisfied, causes the trigger to occur. Such an equation is referred to as a breakpoint equation and there is always a separate equation for trigger T1 and T2.

Events

An event is defined as a certain combination of an address, data, a particular type of bus operation, and a particular set of RTT data probe values, all occurring simultaneously. When all preconditions for an event are met, a signal pulse is generated on the RTT card and the event is said to have occurred. There are two independent events, designated E1 and E2 that relate to triggers T1 and T2, respectively. The conditions for both events are defined with the EVent command.

The circuits on the RTT card that determine when an event has occurred, consists of a set of comparators that compare operator defined requirements to the bus status. The operator defined requirements must be met simultaneously to assert an event and may include any of the following:

1. 16-bit address equal to, less than or equal to, or greater than or equal to.

2. 8-bit data equal to, less than or equal to, or greater than or equal to.

3. Eight RTT data probe bits each equal to 1, 0, or don't care (X).

4. A specific bus transaction type.

Any combination of these preconditions may be logically ANDed.

Pass Counts, Delay Counts, and Trigger Equations

An event may cause a trigger directly and thus initiate a breakpoint. In other cases, it may be desirable to have the event repeat itself several times before a trigger or break is generated. This repetition is referred to as the pass count and is useful where program loop debugging is involved. The pass count is set with the TRig command and can have any value from 0 to 65,535 (a pass count of 0 is equivalent to turning the pass counter off). There are two independent pass counts P1 and P2, related to triggers T1 and T2, respectively.

In some cases it also may be desirable to delay a trigger or a breakpoint after an event. For example, a delay of 64 real-time trace stores allows the trace line associated with an event to be centered in the trace display. The delay may be counted in events, instruction fetches, bus cycles, RTT stores, microseconds, or milliseconds, as selected by the operator with the Count command and recorded in the general purpose counter. This is referred to as the delay count and the number of counts is specified with the TRig command. There are two independent delay counts D1 and D2, related to triggers T1 and T2, respectively. Each delay count can be set to any value between 3 and 65,535; a delay count of 0, 1, or 2 is equivalent to turning the delay counter off.

Events, pass counts, and delay counts can be utilized by the user to set up preconditions for complex breakpoints and thus set up a trap for program bugs or signals in UUT hardware. The serial sequence of an event, followed by a pass count and a delay count, resulting in a trigger, is illustrated in figure 3-2. It is the simplest method for setting up a sequence terminating in a trigger and a breakpoint; there are more complex ways to combine the two events, E1 and E2, both triggers, T1 and T2, start and stop collection of data in the real-time trace buffer, and activate external test equipment through the front panel TRIGGER connectors. These ways of setting preconditions for a complex breakpoint represent four different trigger modes, each of which is discussed separately. For each trigger mode an equation is written in terms of events, pass count, and delay count, as shown in figure 3-2 and subsequent figures in this section.

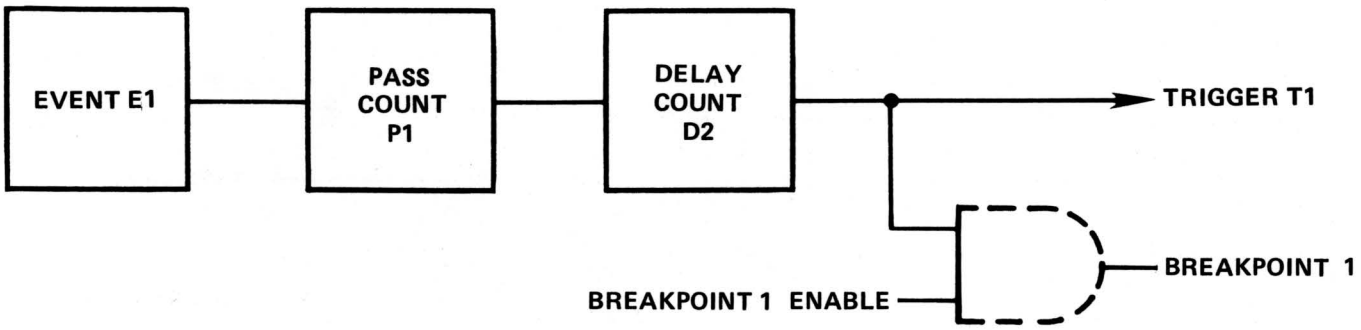
Trigger Modes

There are four trigger modes, each for a particular debugging application:

Independent Mode
Event 1 and Event 2 Mode (E12)
Arm Mode
Freeze Mode

The trigger modes and their trigger equations are described and illustrated below.

Independent Mode. The independent mode allows the user to define two complex breakpoints entirely independent of each other. The two events E1 and E2 are defined separately and, likewise, the other terms of the trigger equations for T1 and T2 do not interact. The logic flow diagram and the equations in figure 3-2 apply both to T1 and T2. Note that in the independent mode the real-time trace buffer always contains the last 128 qualified cycles prior to the breakpoint.



Note: Trigger T2 is generated in the same sequence as shown above for T1.

Trigger Equations (algebraic notation):

P1(E1)+D1 = T1
P2(E2)+D2 = T2

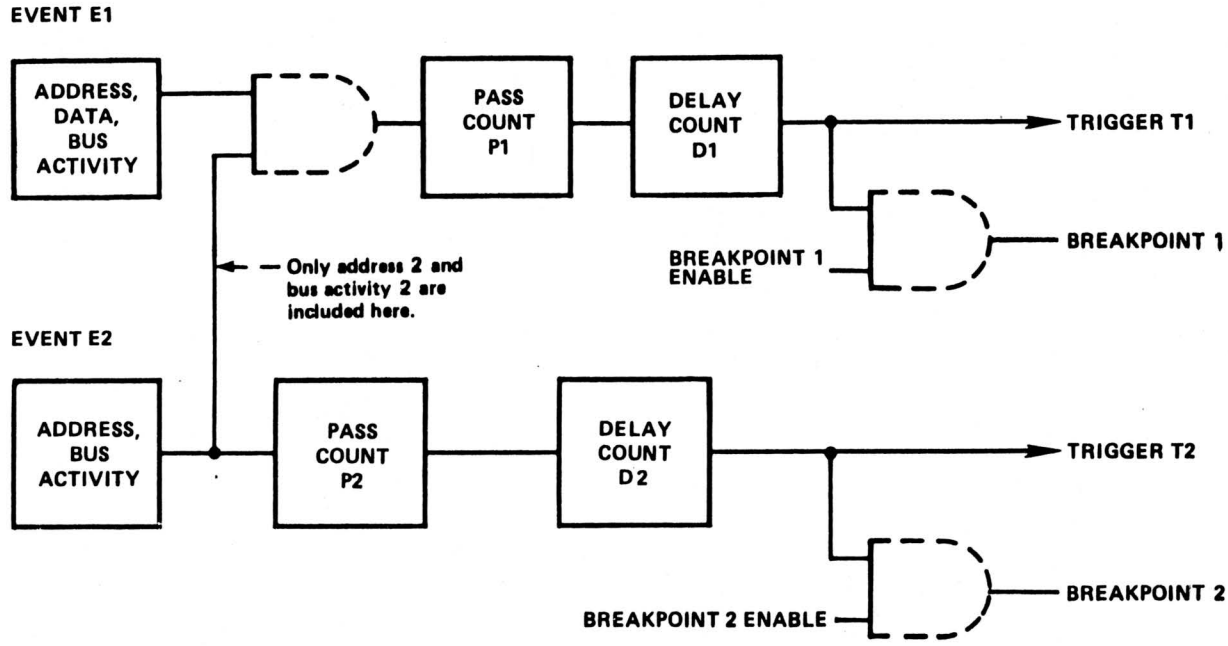
Where E1, E2 = Event 1, 2
P1, P2 = Pass Count 1, 2
D1, D2 = Delay Count 1, 2
T1, T2 = Trigger 1, 2

Figure 3-2. Trigger Generation Sequence Diagram - Independent Mode.

Event 1 and Event 2 Mode (E12). In the E12 mode there is a breakpoint when events E1 and E2 occur simultaneously. As shown in figure 3-3, the two event signals are ANDed and cause trigger T1 and breakpoint 1 to occur. Note that event E2 can be defined only in terms of an address bus value and the type of bus transaction, it cannot include a data bus value.

The E12 mode is typically for applications where it is necessary to break on any program access to a specific area of memory. In such a case event E1 defines any address equal to or greater than the lower address of the memory area, and event E2 defines any value equal to or less than the upper address. The real-time trace buffer contains the last 128 qualified cycles prior to the breakpoint.

Trigger T2 occurs independently, whenever its individual conditions are satisfied, but usually there is no need for it in the E12 mode.



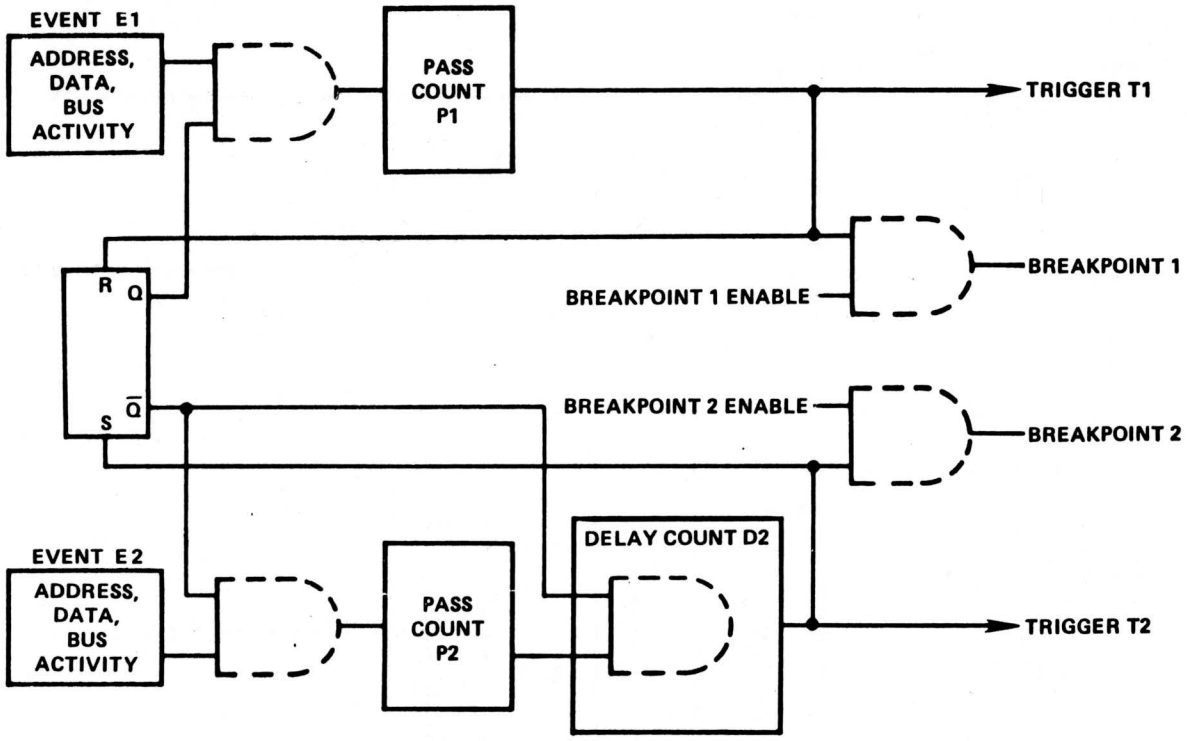
Trigger Equations (algebraic notation):

P1 (E1 ANDed with E2) + D1 = T1
P2 (E2) + D1 = T2

Figure 3-3. Trigger Generation Sequence Diagram - E12 Mode

Arm Mode. In the arm mode, the break occurs at T2, but only after T1 has been asserted (see figure 3-4). That is, if T2 occurs prior to T1, there is no break, regardless of how many times T2 is encountered. The event E1 input is initially enabled, but when trigger T1 is asserted, the event E2 input is enabled and event E1 input is disabled. When trigger T2, the primary trigger, occurs, event E1 input is again enabled and event E2 input is disabled, to preset the logic back to its initial state. This enabling/disabling sequence can occur indefinitely.

The best arm mode application example is having T2 in a subroutine called from a number of places, but not having a break occur until it is called from that part of the program that contains T1. The real-time trace buffer always contains the last 128 qualified cycles prior to the breakpoint. Note that the operator is not allowed to specify a delay count D1 for trigger T1, it must be disabled.



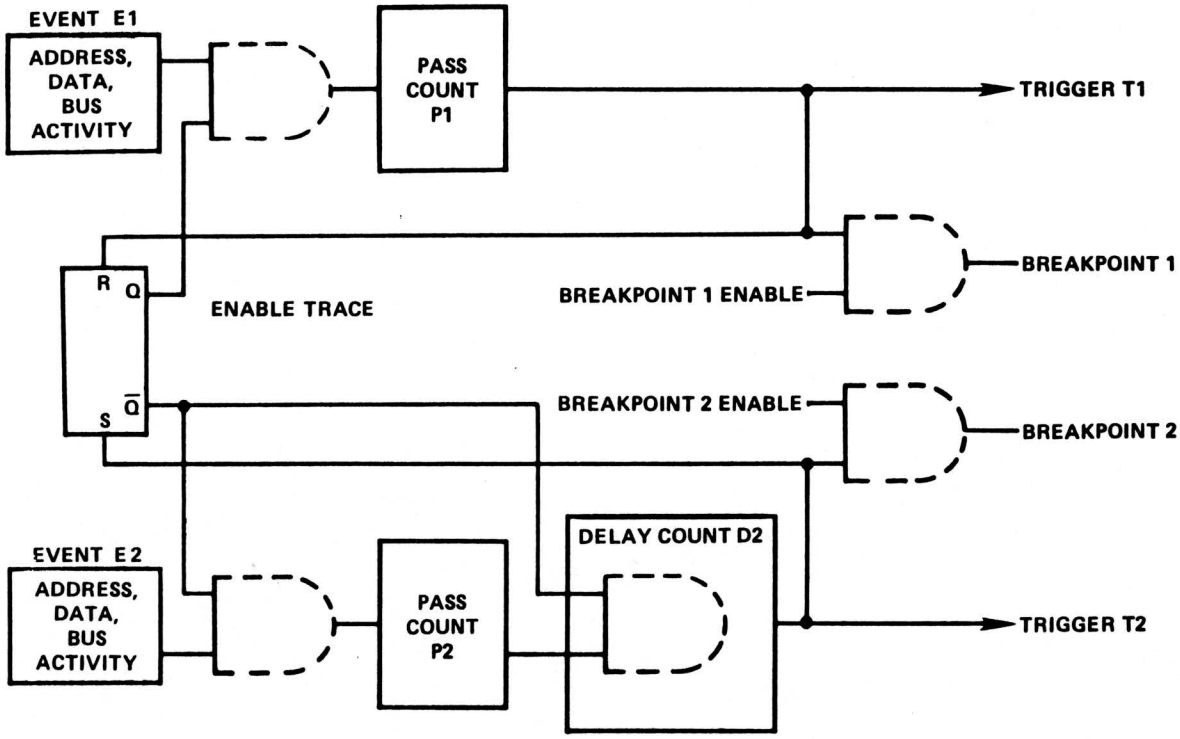
Trigger equations (algebraic notation):

E1 (P1) = T1 (delay D1 is not permissible as a precondition for T1)
T1 + E2 (P1) + D2 = T2

Figure 3-4. Trigger Generation Sequence Diagram - Arm Mode

Freeze Mode. The freeze mode is similar to the arm mode, except that trace buffer stores are prevented between the occurrence of trigger T1 and T2. Storage into the real-time trace buffer is frozen when T1 is encountered (see figure 3-5). This means that no new information is stored in the real-time trace buffer, but the program is allowed to continue until T2 is encountered.

The freeze mode is used when a program must be allowed to run to some conclusion (at T2), but it is necessary to retain a trace of a segment starting at the beginning.



Trigger Equations (algebraic notation):

E1 (P1) = T1 (delay D1 is not permissible as a precondition for T1)
T2 + E2 (P1) + D2 = T2

Figure 3-5. Trigger Generation Sequence Diagram - Freeze Mode

INTRODUCTION

This chapter describes the 9508 MicroSystem Emulator system commands. Each command description explains the overall function of the command, the syntax, all required and optional parameters, system output in response to the command, and includes examples of applications. The commands are described in this chapter in alphabetical order, but appendix B contains a summarized reference list of all commands, both in alphabetical and functional order.

All system commands are entered on the 9508 console terminal keyboard in response to the system prompt D>. A typical command consists of the command name, followed by one or more parameters and terminated by a carriage return. The command name and parameter fields are separated by one or more spaces or commas to fix the limits of each field. In addition, special function keys, the CTRL key used like a shift key in conjunction with selected alphabetic characters, allow editing of a command, repetition of the previous command, echoing of all console characters out the COMM LINK port (J1), as well as other functions. A description of the special function keys is presented later in this chapter.

COMMAND ELEMENTS

Command Name

A command name consists of one to eight alphanumeric characters. Out of these characters there is a minimum number that uniquely identifies the command. This minimum set is, in this chapter and elsewhere in the manual, typed in upper case characters and the nonessential characters are in lower case. Thus, if the command name is

BI)as

a user can invoke the command by typing any of the following:

BI
BIA
BIAS

When entering a command name on the console terminal keyboard, either upper or lower case characters can be used. They will appear on the screen as entered, but internally the system converts all characters to upper case. The documentation convention used in this manual employs the right parenthesis ,), to indicate a legal abbreviation (mnemonic) for the command name, as shown in the example; the right parenthesis,), is not entered on the keyboard. In the syntax description for each command in this chapter the command name also is shown in bold letters. Note that all commands are terminated with a carriage return (CR), but the CR is not explicitly called out anywhere in this manual.

Parameters

Parameters are alphanumeric or relational character strings separated by a space or a comma. Parameters can be further classified as either keywords or user supplied data. Keywords, like command names, must appear in specific syntactical positions, have a minimum character set (shown in capitals), and obey the same abbreviation and upper/lower case conventions as command names (they can be typed in upper or lower case, but are internally converted to upper case). In this manual the minimum character set is shown in capital letters.

User supplied data typically consists of memory addresses, data, or file names. Normally the number of character spaces is limited, and the significance of the data relates to a particular subset of ASCII characters (refer to ASCII Subsets description in this chapter). A more detailed description of the notation for user supplied parameters is in a separate section in this chapter.

CONVENTIONS FOR COMMAND SYNTAX

Various symbols and conventions are utilized in this manual to describe command syntax. Use of the right parenthesis ,), was described, other valid symbols and conventions are described in the following paragraphs.

Braces and Brackets

Braces{ } are used to enclose a required parameter, if the parameter is not entered the system will respond with an error message when trying to execute the command.

Brackets [ ] are used to enclose an optional parameter. If it is not entered the system will invoke to a fixed default value, use the last value entered by the operator, or simply fail to perform the function represented by that parameter.

Both, braces and brackets, are also used to indicate whether multiple parameters need to be position ordered or not. For example,

{parm1 parm2 parm3}

indicates that all three parameters are required, but can be entered in any order. Whereas,

{parm1} {parm2} {parm3}

denotes three required parameters that must be entered in the sequence shown.

Note: Generally, keywords (a specific character or minimum character set from which the program can key) are position independent, whereas user supplied data must be in sequence.

Braces and brackets may also appear nested within each other. The following example illustrates an optional parameter P that, if entered, must be accompanied by any one of the values N, O, or E:

$$\left[\ P = \left\{\begin{matrix} N \\ O \\ E \end{matrix}\right\}\ \right]$$

Stacked Parameters and Defaults

Where several choices exist for a parameter, one of which must be entered, it is indicated by stacking:

Choice 1
Choice 2
Choice 3

Braces or brackets are used around the stack. For example:

$$\text{BR)eak} \left\{\begin{matrix} T1 \\ T2 \\ Both \end{matrix}\right\}$$

indicates that a breakpoint enable must be selected to occur on T1 (trigger 1), T2, or T1 or T2.

If there are several choices of parameters and one of these is a default value, the default option is underlined.

For example, the command

$$\text{NA)me} \left[\begin{matrix} \underline{\text{Choice 1}} \\ \text{Choice 2} \end{matrix}\right]$$

can be entered either as: NA Choice 1
or NA

to select the Choice 1 parameter. If neither parameter is underlined three possibilities exist. If no entry is made for the parameter (it is defaulted):

1. The last previously entered value may be retained
2. The system may default to some fixed value
3. The system may simply bypass the parameter.

Which one of the three possibilities applies is identified in the detailed description of each command.

ADDRESSING MODES

Two ways of addressing are used:

- o Relative Addresses
- o Absolute Addresses

Relative Addresses

The relative address of any particular location is its position relative to the beginning of a relocatable program module. Whenever a user divides software into separate program modules that are to be subsequently linked together to form the executable program, the listing for each module contains relative addresses. Because the debugging tools in the 9508 emulator express addresses in absolute format, the user would constantly need to add the relative addresses on the listing to the absolute starting address of the module in order to follow traces, calculate breakpoints, and locate data variables. Instead, the 9508 emulator allows the user to specify a relative address and modify it with a bias offset value. This modified relative address is called an effective address and is the one actually used in the execution process. With this method of addressing a relative address is identified with, or related to an absolute address.

Relative Address Displays

Addresses appearing in displays generated by various command routines are also absolute or relative. In general, if the user inputs a relative address with a command, the display generated by that command routine will use relative addresses (and conversly, an input of absolute addresses will cause display of absolute addresses). Relative address displays are formatted as shown in the following example

If bias register settings for inputs are:

W = 100, X = 200, Y = 300, and Z = FF00

Absolute Address is displayed as Relative Address

then:	100	= W+0000
	237	= X+0037
	1300	= Y+1000
	FF30	= Z+0030

Absolute Addresses

An absolute address indicates the exact storage location with respect to the beginning of the 9508 emulation memory. Any address entry not preceded by a W, X, Y, or Z bias offset is treated as an absolute address. Any command that accepts a relative address also accepts an absolute address, but not vice versa. Therefore, any command that requires an absolute address parameter, identifies that parameter.

Absolute Addresses are input and displayed as hexadecimal numbers 0-FFFF, as shown in the following example:

```
0    = 0000
20   = 0020
F30  = 0F30
7123 = 7123
```

(leading 0's are always assumed with any inputs)

Addresses appearing in displays generated by various command routines are also absolute or relative, as determined during the input of the command (refer to Relative Address Displays).

USER SUPPLIED PARAMETERS

Parameters shown in all lower case characters represent data that is supplied by the user. The following user supplied parameters are standard throughout this manual.

Hexadecimal Address
hadrr: represents an absolute hexadecimal address. It represents an address in the range 0-FFFF. Leading zeros do not need to be typed. This cannot be a relative address; it must be absolute.

Hexadecimal Data
hh: represents hexadecimal data between 0-FF.

Starting Address
saddr: is a hexadecimal absolute or relative address.

Lower and Upper Address
laddr and uaddr: represent a hexadecimal lower address and upper address, respectively. Both are absolute or relative addresses, but laddr must be less than or equal to uaddr where both are treated as unsigned binary numbers in the range 0-FFFF.

Decimal Data
n: represents decimal data. Unless otherwise specified, the number of characters indicates the range. For example:

n = 0 to 9
nn = 0 to 99
nnn = 0 to 999

Binary Data
b: represents binary data. The number of characters indicates the maximum number expected. Note: the character c is used to denote the RTT data probe signals.

Disk Drive, Filename, File Type Descriptors
d: represents a disk drive, fn represents a filename, and .ft represents a file type extension. The filename consists of 1 to 8 alphanumeric characters, and the file type extension consists of 0 to 3 alphanumeric characters. If no file type is specified, the period (.) is ommitted. (This is a subset of the MP/M and CP/M filenaming convention used in the 9520 Software Development System.)

NOTE

Whenever a string of characters is provided for in a parameter, but only one character is entered, leading 0's are assumed.

Using Bias Offset to Define Effective Addresses

A bias offset is always used to modify a relative address in the 9508 emulator to define the effective address. There are four bias offset variables: W, X, Y, and Z, whose values are stored in bias registers. The values are set using the Bias command and then called up at any time to modify a relative address. For example, suppose the bias offset variable, W=0100, and the relative address=15. Then the command:

EXAM W15

displays the contents of

100 + 15 = 115

In this manner the bias offset value stored in any one of the four registers set by the Bias command can be followed by a hexadecimal relative address value of 0-FFFF, as shown in the following example:

If bias register settings are

W = 100, X = 200, Y = 300, and Z = FF00

	Relative Address	Effective Address	Absolute Address
then:	W0	100+0	= 0100
	X37	200+37	= 0237
	Y1000	300+1000	= 1300
	Z30	FF00+30	= FF30

Addresses may be entered in relative format and modified by the bias offset at any time, unless a command specifically requires an absolute address parameter.

NOTE: Throughout this manual, the term relative address is to be interpreted as an address that is modified by the bias offset value to provide the effective address.

ASCII SUBSETS

Within the full ASCII set (refer to appendix D) there are several groups of characters that are used in command names and parameters for specific purposes. Each of these groups, or subsets contain a limited number of characters or symbols. The table below identifies each subset and its characters or symbols.

CHARACTER SUBSET	CHARACTERS OR SYMBOLS
Alphabetic - Upper Case	A-Z
- Lower Case	a-z
Binary	0,1
Binary with Don't Care	0,1, X, x
Decimal	0-9
Hexadecimal	0-9, A-F, a-f
Relational Operators	=, >, <
Alphanumeric - Upper Case	A-Z
Lower Case	a-z
Numeric	0-9

ERROR REPORTS

Command entry and execution errors are reported in coded format to the console terminal. Command entry errors are indicated by a caret (^) pointing to the error location in the command line and an error code number is displayed. The error codes are described in appendix A. Examples of error reports, as they are displayed on the screen, are as follows:

```
D>BIES      W=20
  ^                                  **ERROR FF**

D>DUMP      30  10
              ^                      **ERROR 14**

D>EXAM      4P
            ^                        **ERROR 02**
```

Errors that occur during emulation may also supply additional error information to the user.

SPECIAL CONTROL CHARACTERS

Special function keys allow the user to enter selected commands by holding the CTRL key down while pressing an alphabetic character key. These are non-printing control characters for initiating commands that aid the user in the following operations:

- Repeating the previous command
- Editing input
- Echo all interactive communications between the console terminal and the emulator COMM-LINK connector J1 on back of the emulator (for use with a hardcopy printer)
- Halt/resume execution of output and command execution (applies only to commands with output)
- Terminate execution of a command and return to command level

Changes to Key Definitions

The TErmdef command can be used to change the special control characters recognized by the 9508 emulator. When the TErmdef command is entered, its parameters (if any) are scanned and the characters mentioned in the parameter list are updated. Then, the current settings are displayed.

Any control character can be entered while the emulator is at command level. Any control character that is not recognized is placed in the command line input buffer and echoed as a period (.) to the console.

The following control characters can be used for editing the command input:

CONTROL KEY	FUNCTION
Control A	Repeat the previous command. This function key is available only immediately after the termination of a command. If any other key is depressed first, the command buffer is erased.
Control H (or BACK-SPACE)	Delete the last character and backspace one character position.
RUBOUT, DELETE	Delete the last character. Note that the deleted character will be echoed. RUBOUT/DELETE is used with hardcopy terminals.
Control X	Delete the entire line entered on the display and backspace to the start of the current line.

There are two control characters that signal the end of input from the console (note that either is equivalent to a CR):

Control Z	End of input from console.
Null Line	A null line is a line that contains no characters - it is entered by depressing a CR at the beginning of a line.

After a command has been entered, there are special keys to control execution:

Control P	Copy all subsequent communications to the COMM-LINK port J1 until the next Control P is entered. This control character can be entered before a command is given, or during the execution of that command.
Space Bar	Stop output to the console terminal until the spacebar is again depressed. This results in stopping the execution of any routine that is in the process of outputting data to the console terminal. Some examples: display of RTT buffer with DRt command; Single step trace display with Go Trace All command; memory display with Dump command.

ESCAPE — Abort execution of the last command and return to command level. If the last command directed the emulator to execute UUT code, a trace line is displayed when the execution is aborted. See the Microprocessor Dependent Addendum for details concerning a specific processor type.

Control \ — Exit from interactive communications (terminal passthrough) mode.

SYSTEM COMMANDS

The following pages describe all 9508 MicroSystem Emulator commands. Commands are presented in alphabetical order for ease of locating. The description of each command is organized in the following format:

- Command Name
- Function
- Command Syntax
- Input Parameters
- Normal Output Response to the Command
- Examples

A summary of all 9508 emulator commands, with syntax, is presented in appendix B.

When the 9508 Microsystem Emulator is initialized by a power on or a reset the following startup message is displayed on the console terminal.

```
9508 READY VER X.X  (Where X.X = software revision level)

PROCESSOR=Z80 V1.0
W=0000 X=0000 Y=0000 Z=0000
MAPPED RAM 1 START=0000 END=1FFF
EVent Address  Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt  TMode  SEL TRIG
  1   = OFF   = OFF  All   XXXX XXXX  1      0 E1        0 MS T1 IND       T1
  2   = OFF   = OFF  All   XXXX XXXX  2      0 E2        0 MS T2 IND       T2

Break T1=Dsbl  T2=Dsbl    Count=     0 MS    Qual=ALL
  LOC  MNEM OPRD/EADDR A   SZHPNC   BC   DE   HL   IX/IY    SP  R  I  IM12 PCNEXT
  0000 XOR  A          00  000000  0000 0000 0000  0000   0000 00 00  0DD   0000
                       00  000000  0000 0000 0000  0000

REGBRK CONDITIONS:

SYSTEM MODE
D>
```

ASM

COMMAND NAME: ASM

FUNCTION: The Asm command is used to modify a program stored in memory. It allows the user to enter CPU instructions at specific memory locations in the form of assembler mnemonics instead of hexadecimal opcode.

(See Microprocessor Dependent Addendum for instruction mnemonics.)

COMMAND SYNTAX: A)sm [saddr]

INPUT PARAMETERS: saddr = The starting address at which the assembly is to begin. This address must be in the range of the address space of the program. If no errors are discovered in the command line, the address at which modification will take place is computed, and the user is prompted for input. The address can be relative or absolute.

Following command entry, the emulator responds with:

```
saddr
```

The operator then must enter the first input. The format for the input is:

```
Operation [operand1 operand2...]
```

where operation is:

```
Instruction mnemonic or
Pseudo operation
```

The instruction mnemonic and operand mnemonics are processor dependent. Every attempt is made to keep the manufacturer's mnemonics and syntax. See the Microprocessor Dependent Addendum for the details of a specific processor type.

The pseudo operations which are supported are:

```
ASCII "char-string"
BYTE hexdata [, hexdata, ...[hexdata]...]
BLOCK nnn
WORD hexdata [, hexdata, ... [hexdata]...]
```

ASM (Cont'd)

where: hexdata is a number from 0 to 0FF. Numbers starting with A-F must be preceded by a 0. The value nnn is a decimal number from 0 to 999. BYTE is used to allocate a byte of storage and to place data in the allocated storage area. BLOCK is used to advance the location counter. It does not alter the contents of memory. ASCII is used to allocate storage and initialize character data. WORD is used to allocate in units of two bytes and initialize them.

The end of input is signalled by a Control Z or a null line.

In response to each input by the operator, the ASM routine displays the input and prompts again by displaying the next sequential address.

NORMAL OUTPUT: D>

is displayed indicating end of job for the ASM command and prompting for the next command.

ERRORS: If the command line has been scanned and found to be correct, the ASM command will prompt the user for operations and operands. If errors are discovered in the operation, the message:

```
** INVALID OPERATION **
```

will be displayed. Similarly,:

```
** INVALID OPERAND **
```

indicates an error in the operands. In either case, the line will be discarded, the current address will be displayed, and the user will be prompted again.

EXAMPLES: Assume that the BI)as command has been used to set

```
W = 400
X = 100
Y = 800
Z = FF0
```

and a type 8048 microprocessor is being emulated.

Valid commands and responses by the ASM routine are:

```
D>A          D>A W        D>A X30
0000         0400         0130
```

Note that the 9508 has responded with the current address in each case.

ASM (Cont'd)

Assume the user has entered A W, and the 9508 emulator responded. The display now looks like:

```
D>A W
W+0000
```

The user enters MOV A,55 which is correct and the 9508 emulator responds with the new address. The display is:

```
D>A W
W+0000  MOV A,55
W+0002
```

If the user enters MVC A,7F, the 9508 emulator responds to this error:

```
D>A W
W+0000  MOV A,55
W+0002  MVC A,7F
** INVALID OPERATION **
W+0002
```

The return to the address enables the user to correct the invalid operation.

Valid Commands

```
D>a
  0000 ascii "1234"
  0004
D>a
  0000 byte 02
  0001
D>a
  0000 block 600
  0258
D>a
  0000 word 08
  0002
```

Invalid Command

```
D>a ascii "1234"   (Invalid parameter)
    Λ                                ERROR 02
```

BIAS

COMMAND NAME: BIAS

FUNCTION: The Bias command is used to define offset values of the bias variables W, X, Y, and Z. These offset values are used in the MOVE, ASM, DISM, DUMP, EXAM, EVENT, GO, WHEX and FILL commands in computing an effective address. The effective address is the value of the bias offset plus the relative address specified in the command.

COMMAND SYNTAX: Bi)as [parm1 parm2 parm3 parm4]

INPUT PARAMETERS: Parameters 1 through 4 are one of these:

```
W=haddr
X=haddr
Y=haddr
Z=haddr
```

where: haddr is a hexadecimal absolute address in the range between 0 and FFFF

The user can change more than one bias value in the same command line. Scanning will continue until an error is detected, or the end of line is found. All changes prior to the error will be made. If all parameters are defaulted, the current status of the bias variable is output.

NORMAL OUTPUT: In response to a command input the current bias offset values are altered, and the new values are displayed:

```
W=haddr X=haddr Y=haddr Z=haddr
D>
```

EXAMPLES:

Valid Commands

```
D>bi
W=0100 X=0F00 Y=0000 Z=0000
D>bi w=fff x=fff y=fff
W=0FFF X=0FFF Y=0FFF Z=0000
```

Invalid Commands

```
D>bi wi=  (Invalid parameter)
     ^                                   ERROR 02

D>bi w=55555  (Invalid hexadecimal character)
       ^                                 ERROR 10
```

BREAK

COMMAND NAME: **BREAK**

FUNCTION: The BReak command is used to enable or disable a breakpoint on the occurrence of trigger T1 or trigger T2.

COMMAND SYNTAX: BR)eak { T1 / T2 / Both } [Disable] [Cont]

INPUT

T1 = Trigger T1
T2 = Trigger T2
Both = Trigger T1 and Trigger T2

Initially (power on or system reset) all breakpoints are disabled. When the BReak command is entered:

Disable = Disable the breakpoint for the trigger(s) specified in parameter 1. If the Disable parameter is not specified (defaulted) the breakpoints are enabled.

Cont = Halt execution, display a trace line, and then resume execution. If this parameter is defaulted, the execution will stop and a trace line will be displayed.

NORMAL OUTPUT: There are two outputs. One occurs when a valid command is given. The other occurs when a breakpoint is encountered.

After the command line has been accepted, full current emulation status is displayed. On the last line of the display the breakpoint status appears as in the following example:

```
BReak: T1=enabled.stop  T2=Disabled
```

STOP or CONT are displayed only for breakpoints which are enabled.

When a breakpoint is encountered, a trace line is displayed (see chapter 3, section III for samples). Although some of the information is dependent upon the emulator, this line will also include:

PCLAST
PCNEXT
Cause(s) of the break in execution
Status of the emulator

(see appendix A for list of cause and status messages)

BREAK(Cont'd)

EXAMPLE:

Valid commands (Consider the commands as being given in sequence.)

```
D>br t1
EVent Address Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt   TMode  SEL TRIG
  1   = OFF   = OFF  All   XXXX XXXX  1      0 E1        0 MS  T1 IND       T1
  2   = OFF   = OFF  All   XXXX XXXX  2      0 E2        0 MS  T2 IND       T2

BReak T1=enbl.stop T2=Dsbl  Count=    0 MS    Qual=ALL

D>br b di
EVent Address Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt   TMode  SEL TRIG
  1   = OFF   = OFF  All   XXXX XXXX  1      0 E1        0 MS  T1 IND       T1
  2   = OFF   = OFF  All   XXXX XXXX  2      0 E2        0 MS  T2 IND       T2

BReak T1=Dsbl  T2=Dsbl  Count=    0 MS    Qual=ALL

D>br t2 c
EVent Address Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt   TMode  SEL TRIG
  1   = OFF   = OFF  All   XXXX XXXX  1      0 E1        0 MS  T1 IND       T1
  2   = OFF   = OFF  All   XXXX XXXX  2      0 E2        0 MS  T2 IND       T2

BReak T1=Dsbl  T2=enbl.Cont  Count=    0 MS    Qual=ALL
D>
```

Invalid commands

```
D>B T1      (The minimum subset is BR.)
  ^                                        ERROR FF

D>BR        (The first parameter is required)
    ^
                                           ERROR 03
```

COUNT

COMMAND NAME: **COUNT**

FUNCTION: The Count command is used to tell the general purpose counter what it should count, or to display the current setting.

COMMAND SYNTAX:

C)ount [Clear] [Ms | Us | Bus | Emclk | Fetch | Rtt | E1 | E2]

INPUT PARAMETERS: If the command is entered without any parameters, full emulation status is displayed. On the last line of the display the current contents of the counter and the units (one of the above parameters) is displayed, as in the following example:

CLEAR = Resets counter to 0.
Ms = Program execution time in milliseconds.
Us = Program execution time in microseconds.
Bus = Count all bus transactions.
Emclk = Count all emulator clocks cycles.
Fetch = Count all instruction fetches.
Rtt = Count all stores into the RTT Buffer.
E1 = Count all occurrences of event E1.
E2 = Count all occurrences of event E2.

After power on or reset, the initial value is Ms and the counter is at 0. The counter is incremented whenever the emulator processor executes the program under test (when Go command is entered). The counter is reset with system reset, when the Clear parameter is entered, or when the units (the second parameter) are changed.

NORMAL OUTPUT: If the command is entered without any parameters, full emulation status is displayed. On the last line of the display the current contents of the counter and the units (one of the above parameters) is displayed as in the following example:

```
Count    56MS
```

COUNT(Cont'd)

EXAMPLE:

Valid Commands

```
D>COUNT Ms      Count milliseconds

D>COU F         Count fetches

D>C E2          Count events E2

D>C             Display current status of parameters
```

```
D>c f
EVent Address Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt   TMode  SEL TRIG
  1   = OFF  = OFF  All   XXXX XXXX  1      0 E1        0 FET T1 IND       T1
  2   = OFF  = OFF  All   XXXX XXXX  2      0 E2        0 FET T2 IND       T2

BReak T1=enbl.stop T2=enbl.Cont  Count=    0 FET   Qual=ALL

D>c b
EVent Address Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt   TMode  SEL TRIG
  1   = OFF  = OFF  All   XXXX XXXX  1      0 E1        0 Bus T1 IND       T1
  2   = OFF  = OFF  All   XXXX XXXX  2      0 E2        0 Bus T2 IND       T2

BReak T1=enbl.stop T2=enbl.Cont  Count=    0 Bus   Qual=ALL
D>
```

Invalid Commands

```
D>COUNT X       ("X" is not a valid keyword)
        ^                                    ERROR 02

D>COU MI        ("MI" is not a valid keyword)
       ^                                     ERROR 02

D>C Ms Us   (Only one parameter can be specified)
       ^                                     ERROR 02
```

DISM

COMMAND NAME: **DISM**

FUNCTION: The Dism Command allows the user to display memory contents in assembly language mnemonic format. The format of the output is dependent upon the processor that is being emulated (refer to Microprocessor Dependent Addendum for particulars on a specific processor).

COMMAND SYNTAX:

D!)sm [laddr] [uaddr | N=nnnnn]

INPUT PARAMETERS: The address can be relative or absolute.

laddr = Lower address in relative or absolute format. The range is 0-FFFF and must be less than or equal to the upper address uaddr. The effective address must be within the address space of the processor. If laddr is defaulted, the address of the next instruction to be executed (PCNEXT) is used.

uaddr = Upper address in relative or absolute format. The range is 0-FFFF and must be within the address space of the processor.

N=nnnnn = This parameter specifies the number of lines which will be output.

If the second parameter is not specified, the default is 10 lines.

NORMAL OUTPUT: Although the assembly language mnemonics and operand syntax are emulator dependent, the following general format is used in every case:

```
ADDR  OBJECT     INSTRUCTION
XXXX  XX XX XX   XXXX XX....
 .
 .
 .
```

A line feed is output every 22 lines and a new header is displayed.

If information is required that can only be known at execution time, the unknown data will be replaced by one or more question marks (?).

DISM (Cont'd)

EXAMPLES: Assume W=100, X=200, Y=800, and Z=1000

Valid Commands

```
D>DISM      Display address=PCNEXT. 10 lines

D>DIS       Display address=PCNEXT. 10 lines

D>DI 100    Display from 100. 10 lines

D>DI X5  X30  Display from 205 to 230
```

The command

```
    D>DI 100 n=12 (Z80 microprocessor)
ADDR  OBJECT            INSTRUCTION
 0100 21 00 03          LD   HL,0300
 0103 06 10             LD   B,10
 0105 11 00 04          LD   DE,0400
 0108 7E                LD   A,(HL)
 0109 3C                INC  A
 010A 12                LD   (DE),A
 010B 23                INC  HL
 010C 13                INC  DE
 010D 05                DEC  B
 010E 20 F8             JR   NZ,0108
 0110 C3 10 01          JP   0110
 0113 E0                RET
 D>
```

Invalid Commands

```
D>DSM   (Invalid command, DISM)
  ^                                    ERROR FF
D>

D>DI fffff (Invalid parameter)
     ^                                 ERROR 02
D>
```

DRT

COMMAND NAME: **DRT**

FUNCTION: The DRt command is used to display information in the RTT buffer. The RTT circuits collect information to store in the RTT buffer as follows:

- o Emulator Processor Address Bus Status
- o Emulator Processor Data Bus Status
- o Emulator Control Bus Status
- o RTT Data Probe Signals

according to the instructions specified with the Qual command. When the emulator has been stopped, the DRt command can be used to display information contained in the RTT buffer. (Refer to chapter 3, section III for a description of the RTT facilities.)

COMMAND SYNTAX:

DR)t [Go / nnn]

INPUT PARAMETERS:
Go = All entries that have been gathered since the last Go command was given (default).
nnn = Is a decimal number in the range 1 to 128.

NORMAL OUTPUT: Consists of a header, a blank line, and twenty entries, followed by a line feed. Then the process is repeated. (See Microprocessor Dependent Addendum for more information.)

```
LOC   DATA   BUS   7 CLIPS 0    INSTRUCTION    DATA

addr  hh     xyz   cccc cccc    mnemonic       operands
 .
 .
 .
 .
 .
```

addr — is the address (If bias registers are set non-zero a relative address is displayed for address equal to or greater than the bias setting.

hh — is hexadecimal data

x — represents the memory or I/O bus operation type indicator. M indicates a memory cycle and I indicates an I/O cycle.

y — represents the read/write indicator. R represents a read cycle and W represents a write cycle.

DRT (Cont'd)

z	represents the fetch/nofetch indicator. F indicates that this is a fetch cycle, while a space indicates that it is not a fetch cycle (i.e., it is not the first byte of an instruction; it is a cycle associated with the execution of the instruction).
cccc cccc	eight binary digits (bits 7-0) representing the logic status of each RTT data probe clip.
mnemonic	represents the operation mnemonic. This field is only present for fetch cycles; otherwise, it contains spaces. It is processor dependent.
operands	represent instruction operands.

EXAMPLES:

Valid Commands

```
D>drt
TRACE BUFFER EMPTY
D>

D>dr 9
TRACE BUFFER EMPTY
D>

D>drt 2
  LOC  DATA   BUS  7 CLIPS 0    INSTRUCTION       DATA

W+F7FB  FA    MR   0000 0000
W+F7FC  F8    MR   0000 0000
D>
```

Invalid Commands

```
D>DT          (Invalid command)
  ^                                    ERROR FF
D>

D>DRT 129     (Value too large)
      ^                                ERROR 02
D>

D>DRT 9A      (A is non-decimal character)
      ^                                ERROR 02
D>
```

DUMP

COMMAND NAME: DUMP

FUNCTION: The Dump command allows the user to display memory in hexadecimal and ASCII format. In the ASCII format, upper and lower case alphabetic, decimal, and special characters are displayed.

COMMAND SYNTAX: D)ump [laddr] [uaddr]

INPUT PARAMETERS: Either address can be relative or absolute.

laddr = is the starting address. If this parameter is not specified then the Dump command remembers the previous address. The initial address is 0 after power on or system reset.

uaddr = is the ending address. If this parameter is not specified, its value depends upon whether or not laddr was specified. If laddr was not specified, the previous value is remembered.

Both, beginning and ending addresses are rounded off, so that memory is always displayed in 16 byte increments. The laddr value is rounded down to the nearest address ending with a 0H, using the following algorithm:

Beginning Address = laddr ANDed with FFF0H

The uaddr value is rounded up to the nearest address ending with a FH, using the following algorithm:

Ending Address = uaddr ORed with 000FH

Both the beginning and ending addresses must fall within the address range of the processor.

DUMP (Cont'd)

NORMAL OUTPUT:

Normal output consists of a header and twenty (or less) lines of display, a line feed, and a repetition of this sequence until the ending address is reached, or the process is aborted (ESC key), or the Space bar is depressed.

```
D>DUMP 0 10

ADDR  0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F   ASCII

0000 41 61 42 62 43 63 44 64   45 65 46 66 47 67 48 68 AaBbCcDd EeFfGgH
0010 20 21 10 2E 30 31 32 33   34 35 36 37 38 39 00 24  !..0123 456789.$
```

Note that there is an extra space separating the contents of 0-7 from the contents of 8-F. The same is true for the ASCII portion of the display.

In the ASCII portion of the display, all upper case, lower case, decimal, and special printable characters are displayed. Non-printing special characters (SOH, DEL, CONTROL-A, etc.) are represented by a period (.); the code for a period (2E) is also represented by a period.

EXAMPLES:

Valid Commands

```
D>d
  ADDR  0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F    ASCII
  0000=04 00 33 34 30 70 E0 78   2B 95 17 0B 0A 06 0E 01    ..340p.x +.......
D>d 0001 0010
  ADDR  0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F    ASCII
  0000=04 00 33 34 30 70 E0 78   2B 95 17 0B 0A 06 0E 01    ..340p.x +.......
  0010=22 D8 50 E8 72 E8 C2 E8   01 01 82 29 02 0A 0C A8    ".P.r... ...)....
D>
```

Invalid Command

```
D>DU 1234 123 (laddr > uaddr)
     ^                                        ERROR 02
D>
```

EMUL

COMMAND NAME: **EMUL**

FUNCTION: The EMul command is used to select the emulator operating mode. The emulaior is capable of operating in either of two modes - system or user - which establishes the following conditions:

Mode	Processor Clock	Mapped Memory	I/O
S	System	Mapped	None
U	UUT	UUT or Mapped	User

In system mode, the processor clock is supplied by the 9508 emulator and memory is emulation memory, as mapped by the user and no I/O.

In user mode, the processor clock is supplied by the UUT, memory is UUT or mapped with the I/O capability. The initial value after power on or reset is system mode.

COMMAND SYNTAX: EM)ul [S / U]

INPUT PARAMETERS:
S = Select system mode
U = Select user mode

If no parameter is specified, the current mode is displayed. The mode is also displayed after power on or reset and reverts to system mode.

NORMAL OUTPUT:
EMULATION MODE = M

where M = SYSTEM or USER

EMUL (Cont'd)

EXAMPLES:

Valid Commands

```
D>EM U
USER MODE

D>EM S
SYSTEM MODE

D>EM        (will display status user or system mode)
```

Invalid Commands

```
D>EL        (Invalid command)
  ^                                          ERROR FF

D>EM 3      (3 is not a valid parameter)
     ^                                       ERROR 02
D>
```

EVENT

COMMAND NAME: EVENT

FUNCTION: The EVent command is used to define events E1 and E2 in terms of:

- o An Address Bus Value
- o A Data Bus Value
- o Type of Bus Transaction
- o State of Each RTT Data Probe Clip

COMMAND SYNTAX: EV)ent {1 / 2} [Clear] [option1 option2 option3 option4]

INPUT PARAMETERS:

1 = Define event E1
2 = Define event E2

Clear — Set everything to don't care or default values. Otherwise, retain current settings of all options which are not changed.

option1 through option4 — These options are used to specify the status of the address bus, data bus, control bus, and RTT data probe signals which constitute the event. Each option can be chosen from the four groups (identified by keywords A,D,B, and E) listed in the table below:

Keyword	Operators	Value	Function
A	=	0-FFFF	Bus Address (hexadecimal)
	=>		
	=<		
D	=	0-FF	Data Bus (hexadecimal)
	=>		
	=<		
B	=	All	All bus activity. (Default value)
		F	Fetch cycles only
		I	I/O address only
		IR	I/O reads only
		IW	I/O writes only
		M	Memory access only
		MR	Memory read only
		MW	Memory write only
		R	Read operations only
		W	Write operations only
E	=	ccccccccc	RTT data probe logic signal status, where c=0, 1, or X

EVENT (Cont'd)

NORMAL OUTPUT: If there are no errors in the command line, the current event definitions are displayed following each entry of the EVent command.

EXAMPLES:

Valid Command

```
D> EVent 1 A=1234 D=<3F B=F E=XX110X00
EVent Address  Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt  TMode  SEL TRIG
  1   = 1234  =<3F   F     XX11 0X00  1      0 E1        0 MS T1 IND       T1
  2   = OFF   = OFF  All   XXXX XXXX  2      0 E2        0 MS T2 IND       T2

Break T1=Dsbl  T2=Dsbl    Count= 1203 MS    Qual=ALL
D>
```

Invalid Command

```
D>EV CLEAR  (no event defined)
     ^                                        ERROR 02
D>
```

EXAM

COMMAND NAME: EXAM

FUNCTION: The Exam command is used to display the contents of a specified memory address on the console screen and then:

1. Modify the contents, increment the address, and display the contents of the next address.

2. Leave the contents unchanged, increment the address, and examine the contents of the next address.

3. Modify the contents of the current address and terminate the command.

4. Leave the contents of the current address unchanged and terminate the command.

Modification is accomplished by keyboard entry and incrementing is done by depressing the space bar.

COMMAND SYNTAX: E)xam [saddr]

INPUT PARAMETERS: saddr = Relative or absolute starting address. The address must be in the range of the processor address space.

If the address parameter is defaulted, address location 0000 is displayed or the last reference.

INPUT/ OUTPUT Exam is an example of an interactive command which alternates between input and output modes. After the command line is entered the address is calculated, the contents of the location are read and displayed, followed by a dash (-). At that time, the user can take any of the four actions listed above. For entering changes to a memory location the valid character set is:

A - F
a - f
0 - 9

EXAM (Cont'd)

EXAMPLES: The four cases are illustrated by the following examples:

Case 1: As an example, suppose the user types:

```
D>EXAM 10
```

The system will display:

```
0010=AB-
```

The user now types two hexadecimal characters (for example F3), which replace the AB at memory location 10. The display is as follows:

```
0010=AB-F3 D8
```

If the user types an invalid character, it will be ignored and a bell (Control - G) character will be echoed. If two valid digits are typed, the system alters the memory location, increments the location counter and displays the contents of the new address.

Case 2: The user does not wish to alter the contents but wants to look at the contents of the next address. The user depresses the space bar, the system displays the existing contents of the current location, increments the address, and displays contents of the next location. Using the example above, the user would see:

```
0010=AB-F3  AC-AC
```

Case 3: Modify the contents of the current address and terminate the command by typing the new data (e.g., 09) followed by a carriage return:

```
0010=AB-F3  AC-09
D>
```

EXAM (Cont'd)

Case 4: The user can also choose to leave the contents of the current address unchanged and terminate the command by a carriage return. For example, instead of entering 09 <CR>, after the contents of address 11 were displayed, the user could have entered a <CR>. The display would look like this:

```
0010=AB-F3  AC-
D>
```

Whenever the address is divisible by eight (the least significant digit is a 0 or an 8), a new line is begun and the corresponding address is displayed, for example:

```
0018=3C
```

The user can abort the Exam command at any time with the ESC key. An address which was in the process of being modified, when the ESC is entered, is not modified and the partial data is discarded.

If a relative address is entered, the address is also displayed in relative form.

```
D>EX  W15
W+0015=AC
```

In case of a memory write error, the system begins a new line, and displays

```
WRITE ERROR ADDRESS=haddr
```

where haddr is the address where the error occurred, and the Exam command is aborted.

FILL

COMMAND NAME: FILL

FUNCTION: The Fill command is used to fill memory from a beginning address to an ending address with a user specified byte or up to 16 8-bit bytes of data. The system will verify the data by performing a write, read, and compare at each address.

COMMAND SYNTAX:

```
F)ill     laddr     uaddr     hex-string
                              'char-string
```

INPUT PARAMETERS: Either relative or absolute addressing can be used for laddr of haddr

laddr	= Lower address, hexadecimal.
uaddr	= Upper address, hexadecimal, must be greater than, or equal to, laddr.
hex-string	= Hexadecimal string up to 16 digits long (requires 2 characters per digit).
'char-string	= ASCII string of up to 16 characters, preceded by a single quote ('):

NORMAL OUTPUT: If there are no errors in the command line or in the execution of the command, it will terminate with the message:

```
FILL COMPLETE
```

After 256 bytes of memory are filled, an X is displayed on the terminal, indicating fill in progress.

FILL (Cont'd)

EXAMPLES:

Valid Commands (assume the bias register W=0050)

```
D>FILL  0  10  87

FILL COMPLETE

D>FILL  W  100  87
X
FILL COMPLETE

D>FILL 0 W 'ABCD

FILL COMPLETE
```

Invalid Commands

```
D>FILL  0  10  (required third parameter missing)
          ^                              ERROR 13
D>
D>FILL  10  0  F3 (laddr > haddr)
           ^                             ERROR 14
D>
```

GO

COMMAND NAME: GO

FUNCTION:

The Go command causes the emulator processor to begin executing the program under test and proceed until a breakpoint is reached, or the ESC key is depressed. The Go command, together with the REGBrk command, determines whether the command execution is continuous (real-time) or single step and if the single step trace is activated. Refer to chapter 3, section III for a description of single step trace.

COMMAND SYNTAX:

```
G)o [saddr1] [Until saddr2 [R]]  [        All    [Count nnnnn      ]]
             [             [W]]  [Trace   Jmp    [From saddr3 To   ]]
             [                ]  [        Clear  [          saddr4 ]]
             [Step [nnnnn]    ]  [                                 ]
```

INPUT PARAMETERS:

If all parameters are defaulted, execution takes place in real-time, starting at PC next. (Note: in this case and all those described below, real-time execution may be preempted by single step execution by activating the REGBrk command). On completion of command execution, the emulator always returns to the command level.

saddr1 = The starting address (relative or absolute) at which execution begins; default is PC next.

Until saddr2 = Continue execution until the relative or absolute address saddr2 is accessed. The access can be qualified to be either a memory read (R) or a memory write (W). The saddr2 must be within the address space of the processor.

Step [nnnnn] = Single step nnnn times. The decimal nnnn can be any value from 1 to 65535; default is 1. When execution of the required number of steps is completed, a trace line of the last processor cycle is displayed.

The third parameter Trace activates single step trace. It is an optional parameter, but if not entered, will default to the previous selection. After power on or reset, the initial value is no trace. If Trace is entered, one of the first three parameters listed below must also be entered:

All = Trace all instructions

Jmp = Trace only branch instructions (i.e. any program discontinuity)

Clear = Turn single step trace off

GO (Cont'd)

Count nnnnn = Specifies the number of single steps to be traced (1 to 65,535). Note that if this count is satisfied before the conditions of the second parameter Until or Step are satisfied, execution is still terminated.

From saddr3 To saddr4 = Specifies where single step trace is to start (saddr 3) and where it is to end (saddr4). Note that execution may start before saddr3 is accessed and continue after saddr4.

NORMAL

After the Go command is entered, the Go program routine determines whether execution is to be in real-time or in single step mode. Whenever execution is in real-time, the following message is displayed:

EXECUTING IN REAL-TIME

Whenever execution is in single step mode, a trace line is displayed for each instruction that satisfies the trace criteria, starting at the top of the screen. A typical trace line appears as follows:

```
LOC  MNEM OPRD/EADDR  A   SZHPNC   BC    DE    HL   IX/IV  SP   R  I  IM12  PCNEXT
0000 LD   A,00        00  000010   E39B  E3E3  0000 0000   0208 4A 00 0DD     0002
                      EB  100001   0000  0000  0000 0000
```

(The above trace line represents the output of a Z80 processor, see other Microprocessor Dependent Addendums, for corresponding trace output information.

ERRORS:

During execution the emulator processor monitors the UUT power and clock. Some emulators also detect illegal opcodes. When a fault condition is detected, the emulator is halted, a break line is displayed, and the cause of the interruption is identified in the same display. See appendix A for a list of messages.

GO (Cont'd)

EXAMPLES: Assume that a Z80 type microprocessor is being emulated. The address space is 0-FFFF.

Valid Commands

```
D>g
EXECUTING IN REAL TIME
  LOC  MNEM OPRD/EADDR  A  SZHPNC  BC   DE   HL   IX/IY  SP  R  I  IM12 PCNEXT
  1945 JR   NZ,191B     A1 000000 0087 2399 9E00  0000 010E 3B 00  0DD   191B
                        F5 100000 0000 0000 0000

EMULATION STOPPED
D>g u 1234  (Start execution at PC next and continue until 1234 is accessed.)
EXECUTING IN REAL TIME
  LOC  MNEM OPRD/EADDR  A  SZHPNC  BC   DE   HL   IX/IY  SP  R  I  IM12 PCNEXT
  1928 INC  B           4F 100001 FD57 BF3F FC00  0000 010E 6A 00  0DD   1929
                        F5 100000 0000 0000 0000  0000
EMULATION STOPPED
D>g s 10 (Start execution at PC next and execute 10 instructions single step.)
  LOC  MNEM OPRD/EADDR  A  SZHPNC  BC   DE   HL   IX/IY  SP  R  I  IM12 PCNEXT
  1934 SUB  B           7A 000011 C408 BF3F FC00  0000 010E 12 00  0DD   1935
                        F5 100000 0000 0000 0000
STEP COUNT COMPLETE
EMULATION STOPPED
D>
```

Invalid Commands

```
D>GO 12345  (starting address outside range)
     ^                                    ERROR 02
D>
D>GO PC  (PC is not a valid parameter)
     ^                                    ERROR 02
D>
```

LINKDEF

COMMAND NAME: LINKDEF

FUNCTION: The Linkdef command is used to define the characteristics of the emulator-host communication link for the uploading or downloading of program files, to display current settings, or to reset to initial values. The characteristics under control of Linkdef are:

- o Baud Rate
- o Half/Full Duplex
- o Format for Data Transfer (Hex, Bin)
- o Parity
- o Protocol (None, Ack/Nak, DSR/DTR)
- o Ack CHARACTER
- o Nak CHARACTER
- o Error Timeout Limit
- o Action on Error
- o Start Synchronization Character
- o Host Abort Character
- o Turnaround time

How the above characteristics are involved in the downloading and uploading process is described in chapter 2 (under Miscellaneous Characteristics of the Emulator-Host Link) and in chapter 6 (under Emulator-Host Communications). Initial (power on or reset) values of these characteristics are stored in a PROM on the control processor card, or selected by DIP switches on the Comm/RAM 1 card.

COMMAND SYNTAX:

L)inkdef [Clear] [B= 110 | 300 | 600 | 1200 | 2400 | 4800 | 9600] [D= H | F]

[Fmt= Hex | Bin] [P= N | O | E] [PR = N | Ack | Dsr]

[Ack=hh] [Nak=hh] [Time=nnnnn]

[Err= I | A] [Sync=hh] [Habt=hh]

[TUrn=nnnnn]

LINKDEF (Cont'd)

INPUT PARAMETERS	Clear	Resets all link characteristics to initial values. If this parameter is not specified, current values remain unless altered in the command line.
	B=	Selects baud rate in the range between 110 and 9600. The initial value corresponds to the DIP switch selection on the Comm/RAM 1 board.
	D=	Selects full or half duplex data transfer. Choices are F or H. The initial value is full duplex. In half duplex mode, a carriage return is simulated by a carriage return and linefeed.
	Fmt=	Selects either TekHex or Binary data format for downloading or uploading. Hex selects the TekHex format. Bin selects the Binary format, and is also the initial value. Note that selection of Binary format also forces parity setting to N (none).
	P=	Selects Parity. choices are N, O, E for none, odd, or even, respectively. Initial value corresponds to the DIP switch selection. Selection of the Bin data format will override the setting and force parity to none.
	PR=	Selects Protocol. Choices are ACK/NAK, DSR/DTR, or no protocol. Initial value is ACK/NAK. (Refer to chapter 6.)
	Ack=hh	Defines ACK character. Two hex characters representing printable ASCII characters can be used. This character, followed by a CR, is used to acknowledge the receipt of a correct block of data during downloading or uploading. The initial value is 30 (ASCII 0).
	Nak=hh	Defines NAK character. Similar to ACK, but indicates that a fault has been detected and requests a retransmission. The initial value is 37H. (ASCII 7).
	Time=nnnnn	Defines timeout limit. Decimal number from 0 to 65535; units are in 100 millisecond increments. If no data is received within the timeout limit, a timeout error occurs. Initial value is 65535.
	Err=	Defines action to take when an error is detected. Choices are to ignore (I), or to abort (A). Initial value is A.

LINKDEF (Cont'd)

Sync=hh — Defines start synchronization character. When the user issues the RHex or WHex command and then exits the terminal mode, the Rhex or Whex program routine sends this character, followed by a CR, to the host program to indicate that it is ready to begin receiving or transmitting data.

Habt=hh — Defines host abort character. Whenever Whex or Rhex is aborted or aborts itself because of an error, it transmits this character to the host processor to abort the communication process in the host. The initial value is Control C which is hex 03.

TUrn — Defines turnaround time. This is the time lapse before the host accepts characters after it has transferred data to the emulator. It is a decimal value between 0 and 65535; units are 100 milliseconds each; default is 0.

*** CAUTION ***

When using Rhex and Whex commands with the Upload and Download programs of the 9520 Development System, do not change the default settings for Fmt, P, Ack, Nak, Sync, and Habt parameters, in order to preserve compatibility between the emulator and host programs.

NORMAL OUTPUT: If there are no errors in the command line, the current values of all parameters are displayed after each entry of Linkdef (it can be entered without any parameters to view the status). The format is as follows:

```
B=9600   TUrn=nnnnn   P=None   D=F     Fmt=Bin   PR=Ack
Err=A    Time=nnnnn   Ack=0    Nak=7   Sync=3    Habt=03
```

Note: If an invalid setting has been entered (on the BAUD RATE switches) for the baud rate, B=INV will be displayed.

LINKDEF (Cont'd)

EXAMPLES:

Valid Commands

```
D>L

B=9600  TUrn=    0  P=None  D=Full  Fmt=Bin  PR=Ack
Err=A   Time=65535  Ack=0   Nak=7   Sync=S   Habt= C

D>L b=2400

B=2400  TUrn=    0  P=None  D=Full  Fmt=Bin  PR=Ack
Err=A   Time=65535  Ack=0   Nak=7   Sync=S   Habt= C
D>
```

Invalid Command

```
D>L b=19200  (Invalid keyword)
      ^                                  ERROR 03
D>
```

MAP

COMMAND NAME: MAP

FUNCTION: The Map command is used to map segments of the UUT memory to the 9508 memory located on the Comm/RAM card. It is also used to disable (unmap) or to display the current map. The 9508 emulator always has a standard 8K block of emulation memory RAM, referred to as block 1. There is also an optional 8K block, referred to as block 2 (the Comm/RAM 2 card). The user must specify the block number, the starting address, and the size of the memory segment to be mapped. The size is dependent upon where in the processor memory space the starting address is located; the following table summarizes the available choices:

If the Starting Address (Haddr) is located at any Boundary of:	Size of Memory Block Available:
1K (0, 400, 800, C00, etc.)	1K
2K (0, 800, 1000, etc.)	1 or 2K
4K (0, 1000, 2000, etc.)	1, 2, or 4K
8K (0, 2000, 4000, etc.)	1, 2, 4, or 8K

COMMAND SYNTAX:

MA)p [[1 / 2 / Clear [1 / 2 / Both]] Haddr [1 / 2 / 4 / 8]]

INPUT PARAMETERS:

1 = Block 1 (RAM 1)

2 = Block 2 (RAM 2)

Both = Blocks 1 and 2; this is the default value for Clear.

Clear = Clear the specified block(s)

Haddr = Lower (starting) address of the map. Hexadecimal absolute address which must lie on a 1, 2, 4, or 8K boundary anywhere within the absolute address space of the emulator processor.

1, 2, 4, 8 = The last parameter is the segment size (units are K bytes) and must be 1, 2, 4, or 8, depending upon the user's needs and the boundary; default value is 1. Figure 4-1 illustrates the boundary addresses for each selection. Once a segment size is selected, any remaining RAM of the 8K block is not available to the user unless the parameter is changed by reentering the MAP command.

MAP (Cont'd)

1K Boundary Addr

```
0000 - 03FF
0400 - 07FF
  .      .
  .      .
  .      .
F800 - FBFF
FC00 - FFFF
```

2K Boundary Addr

```
0000 - 07FF
0800 - 0FFF
  .      .
  .      .
  .      .
F000   F7FF
F800   FFFF
```

4K Boundary Addr

```
0000 - 0FFF
1000 - 1FFF
  .      .
  .      .
  .      .
E000   EFFF
F000   FFFF
```

8K Boundary Addr

```
0000 - 1FFF
2000 - 3FFF
  .      .
  .      .
  .      .
C000 - DFFF
E000   FFFF
```

Figure 4-1. Map Boundary Addresses

NORMAL OUTPUT: If no parameters are entered, the current map remains unchanged and is displayed. If parameters are entered, the map is changed and the new map is displayed (see Examples).

MAP (Cont'd)

EXAMPLES:

Valid Commands

```
D>MAP
NO MAPPED RAM ENABLED

D>MA 1 0 8
(Mapping block 1 starting at 0 and enabling 8K, block 2 not
mapped.)
MAPPED RAM 1 START=0000  END=1FFF

D>MAP 2 2000 1
(Mapping block 2 starting at 2000 and enabling 1K.)
MAPPED RAM 1 START=0000  END=1FFF  RAM2  START=2000  END=23FF
```

Invalid Commands

```
D>M    (Invalid command)
  ^                                    ERROR FF
D>
D>MA C 04FF  (Invalid parameter)
       ^                               ERROR 02
D>
D>MA 2 2000  (RAM 2 not installed)
     ^                                 ERROR 07
D>
```

MOVE

COMMAND NAME: MOVE

FUNCTION: The MOve command moves the contents of memory from one area to another area. The system checks for errors by writing, reading back, and comparing.

COMMAND SYNTAX:

MO)ve { U U / U S / S U / S S } {laddr}{uaddr}{saddr}

The first memory defines the source from where data is to be moved and the destination.

U = User memory (UUT)

S = System memory (emulation memory on comm/RAM card)

INPUT PARAMETERS:

laddr = The starting address in the source memory. It must be within the address space of the processor.

uaddr = The ending address in the source memory. It must be within the address space of the processor and greater than or equal to the laddr.

saddr = The starting address in the destination memory. It must be within the address space of the processor.

All addresses can be absolute or relative.

NORMAL OUTPUT: For each 256 bytes that are moved, the system will display an X while the move is in progress.

MOVE (Cont'd)

EXAMPLES:

Valid Commands

```
D>MO US 0 50 100  (Move from user memory to system memory:
                   laddr=0, uaddr=50, saddr=100.)

D>MO SS 0 FF 100  (Move from system memory to system memory:
                   laddr=0, uaddr=FF (256 bytes) saddr=100.)

X

D>
```

Invalid Commands

```
D>MO US 0 50  (no starting address specified)
           ^                                    ERROR 0D
D>
D>MO SS 50 0 100 (laddr>uaddr)
           ^                                    ERROR 14
D>
```

ONBRK

COMMAND NAME: ONBRK

FUNCTION: The Onbrk command allows the user to specify a list of commands to be executed when a breakpoint is encountered. The commands are not executed for program breaks, which are the result of single step pausing, ESC key, or emulator faults. The maximum number of characters is 128.

COMMAND SYNTAX: O)nbrk [Clear]

INPUT PARAMETERS: Clear = Clear the onbreak commands.

NORMAL OUTPUT: Whenever Onbrk is entered the existing list of commands is displayed, the system enters an interactive mode with the user, and outputs the prompt:

```
0>
```

The user types a command followed by a <CR>. This process is repeated until all commands are entered. The user terminates this mode with a Null Line (a line with no characters preceding a CR). Unless Clear is specified, new commands are appended to the old commands.

ERRORS: If the command name and its parameters are correct, no errors can occur in the command line. During input, it is possible to type more characters than the Onbrk buffer can contain. If that occurs, the message

```
ON BREAK BUFFER FULL
```

is displayed and the command is terminated. The Onbrk command file is not enabled.

NOTE: The commands that are placed in the Onbrk buffer are not checked for accuracy as they are entered into the buffer. A check is made when a breakpoint is encountered and an attempt to execute the commands in the Onbrk buffer is tried.

ONBRK (Cont'd)

EXAMPLES:

Valid Commands

```
D>ONBRK

O>DUMP 0 100
O>DRT GO
O>GO
O>(CARRIAGE RETURN)

D>
```

PORT

COMMAND NAME: PORT

FUNCTION: The Port command is used to examine and optionally write to a target microprocessor port. The 9508 emulator must be in user mode. No check is made after the data is written. In addition, the read or write operation can be repeated until terminated by an ESC.

COMMAND SYNTAX: P)ort haddr [hh] [Rep]

INPUT PARAMETERS:

haddr = a valid port address for the processor that is being emulated (see the Microprocessor Dependent Addendum). This is an absolute address.

hh = Hexadecimal data. If this parameter is present, the data hh is written to the port. If no data is specified, a read from the port is performed.

Rep = Repeat. If this parameter is specified, the read or write operation is repeated until the user aborts the process with an ESC.

OUTPUT: When a port is read,

```
haddr=hh
```

is displayed. In repeat mode, the display is updated only when the data changes.

USER INTERACTION: The ESC key will abort a continous function.

PORT (Cont'd)

EXAMPLES:

Valid Command

```
D>P 4
04=25
```

NOTE: Because the system must check for errors, display a message for the read operation, and reschedule the operation, the next read or write operations do not immediately follow one another in repeat mode.

Invalid Commands

```
D>P 4  (9508 in system mode)
                                    ERROR 12
D>
D>P4  (no delimiter "space")
  ^
                                    ERROR FF
D>
```

QUAL

COMMAND NAME: QUAL

FUNCTION: The Qual command is used to qualify or identify the type of information which is to be collected in the real-time trace (RTT) buffer or to display the current selection. Refer to chapter 3, section III for a description of the RTT buffer.

COMMAND SYNTAX: Q)ual [All | Fetch | I | IR | IW | M | MR | MW | R | W]

INPUT PARAMETERS:

All	= Trace all processor cycles
Fetch	= Fetch cycles only
I	= All I/O
IR	= I/O reads only
IW	= I/O writes only
M	= All memory accesses
MR	= Memory read only
MW	= Memory write only
R	= All read operations
W	= All write operations

If no parameter is entered with the command, the current selection remains unchanged and is displayed.

NORMAL OUTPUT: QUAL = selection
where selection is the current value, from the list of input parameters.

QUAL (Cont'd)

EXAMPLES:

Valid Commands

```
D>q f
EVent Address  Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt  TMode  SEL TRIG
  1   = OFF   = OFF  All   XXXX XXXX  1      0 E1        0 MS T1 IND       T1
  2   = OFF   = OFF  All   XXXX XXXX  2      0 E2        0 MS T2 IND       T2

Break T1=Dsbl  T2=Dsbl    Count=     0 MS    Qual=F
D>q mw
EVent Address  Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt  TMode  SEL TRIG
  1   = OFF   = OFF  All   XXXX XXXX  1      0 E1        0 MS T1 IND       T1
  2   = OFF   = OFF  All   XXXX XXXX  2      0 E2        0 MS T2 IND       T2

Break T1=Dsbl  T2=Dsbl    Count=     0 MS    Qual=MW
D>
```

Invalid Commands

```
D>qua X    (X=Invalid keyword)
     ^                                  ERROR 02
D>
D>q M W    (M W invalid keyword)
     ^                                  ERROR 02
D>
```

RESET

COMMAND NAME: RESET

FUNCTION: The RESet command is used to apply a reset signal to the reset pin of the emulator processor. Any registers that are affected by a reset signal are also reset, e.g., the program counter, PC next. See Microprocessor Dependent Addendum for more details.

COMMAND SYNTAX: RES)et

INPUT PARAMETERS: NONE.

NORMAL OUTPUT: NONE.

```
9508 READY VER X.X  (Where X.X = software revision level)

PROCESSOR=Z80 V1.0
W=0000 X=0000 Y=0000 Z=0000
MAPPED RAM 1 START=0000 END=1FFF
EVent Address  Data  Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt  TMode  SEL TRIG
  1   = OFF    = OFF  All   XXXX XXXX  1      0 E1        0 MS T1 IND       T1
  2   = OFF    = OFF  All   XXXX XXXX  2      0 E2        0 MS T2 IND       T2

Break T1=Dsbl  T2=Dsbl    Count=     0 MS    Qual=ALL
  LOC  MNEM OPRD/EADDR A   SZHPNC   BC   DE   HL  IX/IY    SP  R  I  IM12 PCNEXT
  0000 XOR  A          00  000000  0000 0000 0000  0000   0000 00 00  0DD   0000
                       00  000000  0000 0000 0000  0000
```

REGBRK CONDITIONS:

SYSTEM MODE
D>

ERRORS: NONE.

EXAMPLES: NONE.

REG

COMMAND NAME: REG

FUNCTION: The REG command is used to display, and optionally to alter the registers of the emulator processor. It is an interactive command, similar to the Exam command. This command is also emulator dependent and the user should consult the Microprocessor Dependent Addendum for details.

COMMAND SYNTAX: REG [keyword]

INPUT PARAMETERS: The keywords are emulator dependent. (See Microprocessor Dependent Addendum.) If no keyword is entered, all registers are displayed in sequence, one for each time that the space bar is depressed.

EXAMPLES:

Valid Commands (Z80 Microprocessor)

```
D>REG A

A=00-00 S=0-00 Z=0-  (Each space bar depression will cause the
                      next sequential register to be displayed.)
D>REG A
A=BB-BB
D>REG A
A=BB-BB S=0-00 Z=0-
```

Invalid Commands

```
D>RE A  (Invalid command)
  ^                                   ERROR FF
D>
D>REG X  (Invalid keyword)
     ^                                ERROR 02
D>
```

REGBRK

COMMAND NAME: REGBRK

FUNCTION: The REGBrk command allows the user to define a simple breakpoint, using the conjunction of one to eight conditions involving emulator processor registers; a break occurs only when all conditions are simultaneously satisfied. Conditions are defined in terms of:

register-relation-value

The register names are emulator dependent. See Microprocessor Dependent Addendum for a list of valid register names.

Whenever a REGBrk breakpoint has been set, the emulator operates in single step mode.

COMMAND SYNTAX: REGB)rk [Clear] [condition 1... [condition 8]]

INPUT PARAMETERS: Clear = Clears current conditions. If this parameter is not specified, any new conditions are appended to existing conditions.

condition = register-relation-content of register.

For all registers, the relations are:

.EQ.	=	Equal
.NE.	=	Not Equal
.LT.	=	Two's complement - Less Than
.ULT.	=	Unsigned - Less Than
.GT.	=	Two's complement - Greater Than
.UGT.	=	Unsigned - Greater Than
.LE.	=	Two's complement - Less Than or Equal
.ULE.	=	Unsigned - Less Than or Equal
.GE.	=	Two's complement - Greater Than or Equal
.UGE.	=	Unsigned - Greater Than or Equal

Since a register can appear in more than one condition, a window of values can be established. For example:

```
D>REGBRK  C PC.UGE.100  PC.ULE.1FF
```

causes a break whenever the program counter is between 100 and 1FF.

If no parameters are entered with the command, the current conditions are displayed.

REGBRK (Cont'd)

NORMAL OUTPUT: The current conditions are displayed.

EXAMPLES:

Valid Commands

```
D>REGBRK C PC.UGE.100 PC.ULE.1FF

REGBRK CONDITIONS:

PC.UGE.0100  PC.ULE.01FF

D>
```

Invalid Command

```
D>REGB PCUGE.100  (No period (.) delimiter)
        ^                                        ERROR 02
D>
```

RHEX

COMMAND NAME: RHEX

FUNCTION: The RHex command is used to transfer (download) a user program file from a host system to the 9508 emulator. Refer to chapter 3, section II for a description of emulator-host communications.

COMMAND SYNTAX:

RH)ex [Nosync] { TErm | TRans | 'String }

INPUT

Nosync = No synchronization of the Rhex program routine in the emulator and the Download program in the host is performed before the emulator enters the transfer mode.

TErm = Selects the Interactive Communications (Terminal Passthrough) mode.

TRans = Selects Transfer mode.

'string = Selects Single Command Mode. This string is the name of the file to be downloaded. The Rhex subroutine appends it to the DOWNLOAD command, together with a CR and sends it to the host. The maximum number of characters is 40.

Refer to chapter 3, section II for instructions on how to use the RHex command during downloading.

NORMAL OUTPUT: The Rhex program routine notifies the user when

Interactive Communications mode is entered - "TERMINAL MODE"
Interactive Communications mode is exited - "EXIT TERMINAL MODE"
Start-Synchronization is completed - "SYNC OK"
Data blocks are transferred - "ADDR=hhhh"
Termination block is received - "PC=hhhh"

RHEX (Cont'd)

EXAMPLES:

Valid Command

```
D>rh 'myfile
SYNC OK
TRANSFER MODE
ADDR=0000

PC=0100

EXIT TRANSFER MODE
D>
```

Invalid Command

```
D>rh    (Parameter required)
    ^                                    ERROR 04
D>
```

ERRORS: In addition to the errors that can occur during entry of the command, a number of errors can also can occur during execution. The user is notified of any errors and if Rhex aborts on errors, it will display ERROR RC=hhhh. (See TERMINAL command for description of bit definitions in the error message.) If the user has instructed (with Linkdef command) Rhex to ignore errors, it will continue; otherwise, it will send a host abort character to the host and abort the Transfer Mode.

If a file open error or read error is detected by the Download program, the host will send an abort block. The emulator will then display:

COMM ABORTED BY HOST

STATUS

COMMAND NAME: STATUS

FUNCTION: The Status command is used to display the current status of the emulation. It displays two types of information:

- o Emulation Environment
- o Processor Registers

The emulation environment information is independent of the processor type and includes current:

Bias setting
Counter selection and value
Event definitions
Mapping information
Realtime trace qualification
Trigger mode
Trigger and Breakpoint definitions
Register break definition

The processor register data consists of the contents of all registers and the flags. This information is processor dependent - see Emulator Dependent Addendum for more details. The processor register information is displayed in the same format as with the REG command.

COMMAND SYNTAX: S)tatus [Env / Reg]

INPUT PARAMETERS: If no parameter is entered, all information is displayed.

Env = Display only the emulation environment information.

Reg = Display only the processor register information.

STATUS (Cont'd)

EXAMPLES:

Valid Commands (Z80 Microprocessor)

```
D>s env
W=0000 X=0000 Y=0000 Z=0000
MAPPED RAM 1 START=0000 END=1FFF
EVent Address Data Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt  TMode  SEL TRIG
  1    = OFF  = OFF All   XXXX XXXX  1      0 E1        0 MS T1 IND       T1
  2    = OFF  = OFF All   XXXX XXXX  2      0 E2        0 MS T2 IND       T2

BReak T1=Dsbl  T2=Dsbl   Count=   0 MS    Qual=ALL
USER MODE
D>s reg
PROCESSOR=Z80 V1.0
  LOC  MNEM OPRD/EADDR  A  SZHPNC  BC   DE   HL   IX/IY  SP  R  I  IM12  PCNEXT
  0000 RET  NC          00 000000 0000 0000 0000  0000 0000 00 00  0DD    0000
                        00 000000 0000 0000 0000  0000
REGBRK CONDITIONS:
```

Invalid Command

```
D>S Q  (Invalid parameter)
    ^                                   ERROR 02
D>
```

TERMDEF

COMMAND NAME: TERMDEF

FUNCTION: The TErmdef command is used to display or modify the special control characters recognized by the 9508 emulator (described in the first part of this chapter).

COMMAND SYNTAX: TE)rmdef [keyword=hh]

INPUT PARAMETERS:

AGn=hh — Changes the character that causes the previous command to be repeated. Default is Control A (01H).

Bs=hh — Changes the character that causes backspacing and deletion of the previous keyboard character. Default is Control H (08H).

Rub=hh — Changes the character that causes deletion and echoing of the previous keyboard character to the hardcopy terminal. Default is RUB (7FH).

Line=hh — Changes the line delete character. Default is Control X (18H).

Halt=hh — Changes the halt typing character. Default is Spacebar (20H).

ABort=hh — Changes the abort command character. Default is ESC (1BH).

Pass=hh — Changes the exit Passthrough (Interactive Communications) Mode character. Default is Control \ (backslash) (1CH).

ECho=hh — Changes the character that causes communications between the console terminal and the emulator to be echoed to the J1 connector on back of the 9508. Default is Control P (10H).

Note: hh must be in ASCII code.

OUTPUT: If the command is entered without any parameters, the current special control characters are displayed.

TERMDEF (Cont'd)

EXAMPLES:

Valid Commands

```
D>
AGn= A  Bs= H  Rub=DEL  Line= X  Halt=SP  ABort=ESC  Pass=   ECho= P
D>te agn=10
AGn= P  Bs= H  Rub=DEL  Line= X  Halt=SP  ABort=ESC  Pass=   ECho= P
D>te ag=01
AGn= A  Bs=H   Rub=DEL  Line= X  Halt=SP  ABort=ESC  Pass=   ECho= P
D>
```

Invalid Command

```
D>te Bs=.     (Invalid Hex character)
       ^                                  ERROR 10
D>
```

TERMINAL

COMMAND NAME: TERMINAL

FUNCTION: The TERMINAL command causes the 9508 emulator to enter the Interactive Communcations (Terminal Pass through) mode, in which all characters received from the keyboard are sent out the COMM-LINK connector J1 to the host, and all characters received from the host are displayed on the console terminal. Although the terminal is always connected to the 9508 emulator in full duplex mode, the emulator can emulate a half or full duplex terminal to the host. The default is full duplex, but the selection can be changed with the Linkdef command.

All characters are passed through on a character-by-character basis until the exit pass through character is received from the keyboard or the host. The initial value of this special control character is Control (1CH), but it can be altered using the TErmdef command.

COMMAND SYNTAX: TERMINAL

INPUT PARAMETERS: NONE.

MESSAGES: When the emulator enters the Terminal Pass through mode, it displays:

```
TERMINAL MODE
```

When the emulator receives the exit pass through character from the console or the remote port, it exits and displays:

```
EXIT TERMINAL MODE
```

ERROR: If an error occurs while communicating across the link, the emulator displays:

```
ERROR RC = hhhh
```

Where hhhh are two hexadecimal bytes that indicate the type(s) of error(s).

TERMINAL (Cont'd)

The bit definitions for RC are:

HIGH ORDER BYTE: BINARY BIT	MEANING
7	Termination block was received from host
6	Abort block was received from host
5	Esc key was depressed at 9508 console
4	Checksum Error
3	Memory Write Error
2	Invalid TekHex character
1	Non-recoverable comm error
0	Recoverable comm error

LOW ORDER BYTE: BINARY BIT	MEANING
7	Timeout occurred for a host read
6	Parity Error
5	Data Overrun
4	Frame Error
3	Not Used
2	Loss of Data Carrier
1	(not used)
0	Data Available (not an error condition)

Note that many of these errors cannot occur while the emulator is in Interactive Communications Mode. Instead, such errors (e.g., Checksum Error) can occur and are checked for only during downloading (see Download/Upload-Case 3).

TMODE

COMMAND NAME: TMODE

FUNCTION: The TMode command serves to define the specific manner in which trigger 1 and trigger 2, and event 1 and event 2 conditions are combined relative to each other. These are called Trigger Modes and are described in section III of chapter 3. The command is also used to display the current selection.

COMMAND SYNTAX:

TM)ode [Ind | E12 | Arm | Frz]

INPUT PARAMETERS: The command is used to define the relationship between events E1 and E2 and trigger T1 and T2 (the Trigger Mode). For the purposes of this discussion, these symbols will be used:

E1 -- Event 1	E2 -- Event 2
P1 -- Pass count for Trigger T1	P2 -- Pass count for Trigger T2
D1 -- Delay Event for Trigger T1	D2 -- Delay Event for Trigger T2
T1 -- Trigger 1	T2 -- Trigger 2

Ind = Trigger T1 and T2 are independent. The equations (in algebraic notation) for the independent mode are:

T1 = P1 (E1) + D1
T2 = P2 (E2) + D2

The equation for T1 means P1 occurrences of E1, followed by D1, will cause T1 to occur.

E12 = Trigger T1 is a combination of Events E1 and E2 (this is the E12 mode).

T1 = P1 (E1.E2) + D1
T2 = P2 (E2) + D2

where E1.E2 indicates the simultaneous occurrence of Events E1 and E2.

Arm = T1 is armed and T2 is disarmed (Arm mode). When T1 occurs, it disarms itself and arms T2. When T2 occurs, it disarms itself and arms T1.

Frz = This mode is the same as the Arm mode, except that no more data is stored in the RTT buffer after the occurrence of T1.

TMODE (Cont'd)

NORMAL OUTPUT: After TMode command is entered, the emulator displays the current trigger mode.

TMODE = mode

where mode is IND, E12, ARM, or FRZ.

EXAMPLES:

Valid Commands

```
D>tm a
EVent Address Data Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt  TMode  SEL TRIG
  1    = OFF  = OFF All   XXXX XXXX  1      0 E1        0 MS T1 Arm T2    T2
  2    = OFF  = OFF All   XXXX XXXX  2      0 E2        0 MS T2 Arm T1

BReak T1=Dsbl  T2=Dsbl   Count=   0 MS    Qual=ALL
D>

D>tm f
EVent Address Data Bus   7 CLIPS 0 TRig Pass cnt   Delay cnt  TMode  SEL TRIG
  1    = OFF  = OFF All   XXXX XXXX  1      0 E1        0 MS T1 Arm T2    T2
  2    = OFF  = OFF All   XXXX XXXX  2      0 E2        0 MS Frz AT T1

BReak T1=Dsbl  T2=Dsbl   Count=   0 MS    Qual=ALL
D>
```

Invalid Commands

```
D>tmi  (Invalid command)
  ^                                        ERROR FF
D>

D>tm ia (Invalid parameter)
     ^                                     ERROR 02
D>
```

TRIG

COMMAND NAME: TRIG

FUNCTION: The TRig command is used to define trigger T1 and trigger T2 pass and delay counts. Refer to section III of chapter 3 for a definition and description of pass and delay counts.

COMMAND SYNTAX: TR)ig $\{1 \atop 2\}$ $\left[\begin{matrix}P=0 \\ n\end{matrix}\right]$ $\left[\begin{matrix}D=0 \\ n\end{matrix}\right]$

INPUT PARAMETERS:

1 = Trigger T1

2 = Trigger T2

P 0 = Pass Count. A decimal number that states how many times
n the trigger addressed by the first parameter must occur before the pass count precondition is satisfied. Its value can be from 0 to 65,535. The initial and default value is 0 (it turns the counter off).

D 0 = Delay Count. A decimal number that states how many counts
n must occur before the delay precondition for the trigger addressed by the first parameter is satisfied. The units of the count are defined with the Count command. The value can be from 3 to 65,535; the initial and default value is 0 (it turns the counter off). Note that in the Arm and Freeze trigger modes the delay counter for trigger T1 must be turned off.

NORMAL OUTPUT: After the TRig command is entered, the emulator displays the current emulation status.

TRIG (Cont'd)

EXAMPLES:

Valid Commands

```
D>Tr 2 p=2 d=0
EVent Address  Data  Bus   7 CLIPS 0 TRig Pass cnt  Delay cnt    TMode  SEL TRIG
  1   = OFF   = OFF  All   XXXX XXXX  1      0 E1       0 MS   T1 IND       T1
  2   = OFF   = OFF  All   XXXX XXXX  2      2 E2       0 MS   T2 IND       T2

BReak T1=Dsbl  T2=Dsbl     Count=   0 MS    Qual=ALL
D>tr 1 p=1 d=4
EVent Address  Data  Bus   7 CLIPS 0 TRig Pass cnt  Delay cnt    TMode  SEL TRIG
  1   = OFF   = OFF  All   XXXX XXXX  1      1 E1       4 MS   T1 IND       T1
  2   = OFF   = OFF  All   XXXX XXXX  2      2 E2       0 MS   T2 IND       T2

BReak T1=Dsbl  T2=Dsbl     Count=   0 MS    Qual=ALL
D>
```

Invalid Command

```
D>tr 2 p=2 d=2  (Delay count error)
               ^                          ERROR 08
D>
```

WHEX

COMMAND NAME: WHEX

FUNCTION: The WHex command is used to transfer (upload) a user program file from the 9508 emulator to a host system. The program file can exist in up to four non-contiguous memory segments. Refer to chapter 3 section II for a description of uploading.

COMMAND SYNTAX: WH)ex [Nosync] saddr [addr1 uaddr1....[laddr4 uaddr4] TErm TRans 'String

INPUT PARAMETERS:

Nosync = No synchronization of Whex program routine in the emulator and the Upload program in the host is performed before Whex enters the Transfer Mode.

saddr = Execution address. It is the location (relative or absolute address) at which execution of the program being uploaded is to begin.

laddr1 uaddr1 = Contiguous memory segment from where the program is to be reak for uploading (relative or absolute address). Four such segments can be defined with addr1 through addr4 values.

TErm = Selects Interactive Communications (Terminal Passthrough) Mode for uploading,

TRans = Selects Transfer Mode for uploading.

'String = Selects Single Command Mode. This string is the name of the file to be uploaded. The Whex subroutine appends it to the UPLOAD command, together with a CR, and sends it to the host.

Refer to chapter 3, section II for instructions on how to use the WHex command for uploading.

ERRORS: In addition to the errors which can occur during entry of the command line, a number of errors can occur during execution. The user is notified of any errors. If the user has indicated that errors are to be ignored, Whex will continue, otherwise, it will send an abort block and terminate Transfer Mode. If the Upload program in the host detects a file open or write error, it will send the host abort character, Control C, and the Whex program will display:

COMM ABORTED BY HOST

WHEX (Cont'd)

EXAMPLES:

Valid Command

```
D>wh 100 0 100 'myfile
SYNC OK
TRANSFER MODE
ADDR=0000
ADDR=003C
ADDR=0078
ADDR=00B4
ADDR=00F0

PC=0100

EXIT TRANSFER MODE
D>
```

Invalid Command

```
D>wh   (Parameter required)
    ^                                    ERROR 0D
D>
```

GENERAL

The 9508 MicroSystem Emulator is a stand-alone software/hardware debug instrument with 8K bytes of static RAM for mapping into the user's memory space. This 8K bytes of RAM is expandable to 16K bytes by the addition of an optional 8K RAM board. With the addition of a 9501 display terminal (or any other compatible display terminal), the user can enter small programs from the console, download directly from a development system and debug the software/hardware. Software debugging can be accomplished either in the memory of the 9508 or in the unit under test (UUT). The following paragraphs provide a functional description of each of the printed circuit modules in the 9508 system and their interface.

HARDWARE ARCHITECTURE

The 9508 is based on a dual processor concept. The control processor module (figure 5-1) acts as the master while the emulator processor is the slave. They will be referred to as the master and slave. Both processors operate in a master and emulator mode over a common bus.

Operating over a common bus, it is necessary to have a controlling element to prevent both processors from accessing simultaneously. This function is performed by the debug module, as directed by controls from the master CPU.

All external communications are received by the master CPU. The emulator obtains control of the bus when the master CPU is halted. Control is transferred back to the master when a system interrupt occurs.

The memory and I/O in the 9508 is separated into master and emulator sections. This allows the user complete use of slave memory and I/O space while protecting the master sections from user access.

There are basically two different master I/O functions. Internal master I/O consists of writing commands to the individual modules such as the debug, an emulator or the real-time trace module and reading status from these modules. External I/O is accomplished by employing the communications RAM module, which has two RS-232C interfaces.

The emulation cable assembly allows in-circuit emulation of user hardware. The cable connects to the slave CPU on one end and on the other replaces the processor in the UUT. This gives total debugging capability in an actual hardware environment.

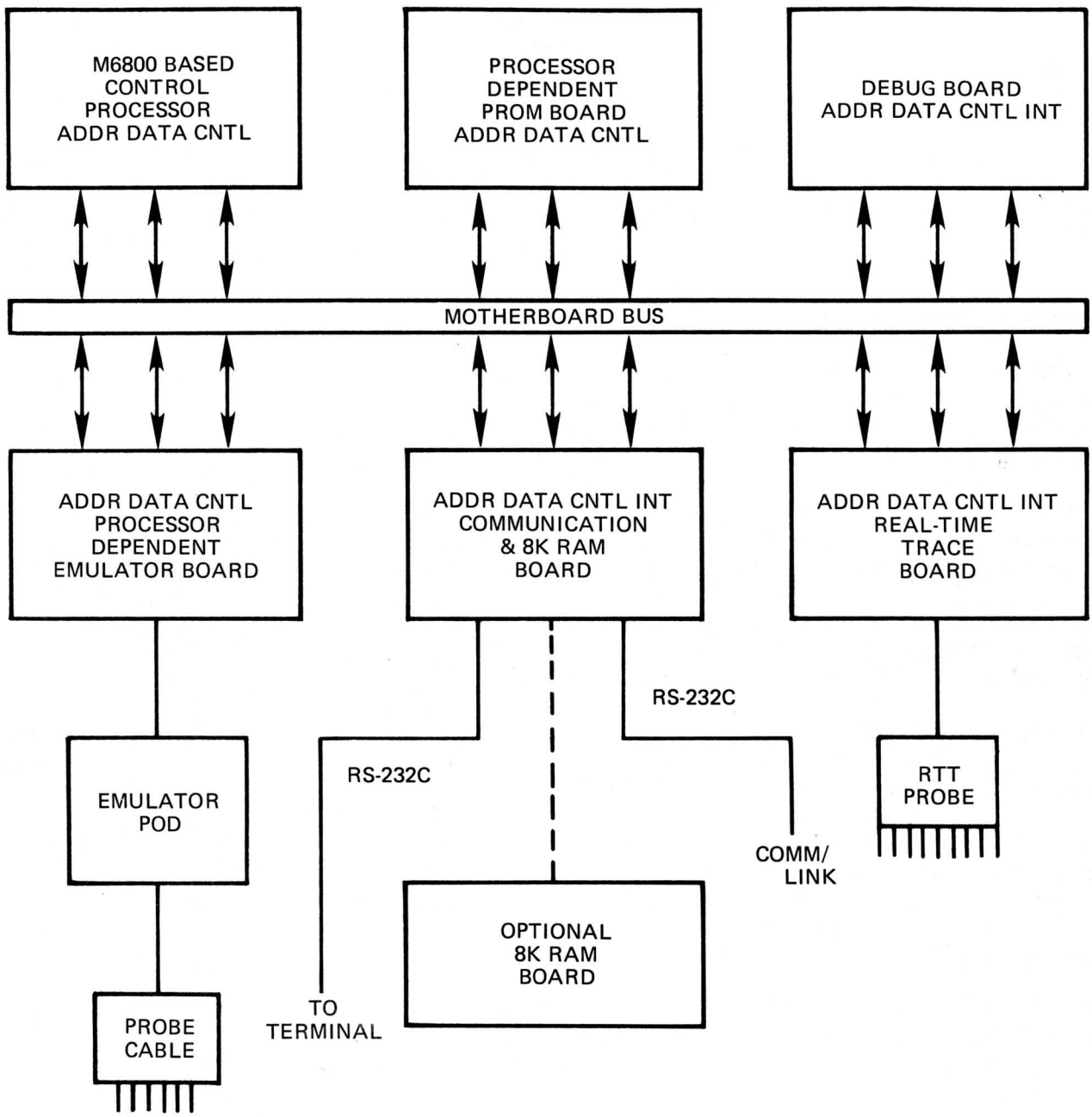


Figure 5-1. 9508 MicroSystem Emulator, Block Diagram

CONTROL PROCESSOR

The 9508 control processor is a single circuit card microcomputer that serves in the system as the master processor. The control processor is a 6800 microprocessor chip and supporting circuits, including a cycle timing generator, various ROM and RAM memory with their address and control signal decoding and generating logic, buffering circuits that interface signals to and from the system bus and I/O connections.

DEBUG MODULE

The debug module is a single circuit card that performs three distinct functions within the 9508: a) controls the interaction and bus timesharing between the master and slave CPU, b) supports such debug features as software and hardware breakpoint, trace, single cycle, and start of slave execution at any memory location, and c) provides a means of generating vectored master CPU interrupts from various sources throughout the 9508. The card is centrally located in the bus structure, providing a dividing line between master and slave halves. Unique control lines are connected to the master CPU and slave CPU along each half of the bus.

EMULATOR MODULE

The control processor module, as directed by the operating system, will set the mode of operation for an emulator, establish an active condition, or enable or disable the utility PROMs. This is accomplished with a master I/O write to the slave I/O port.

Since the 9508 supports different types of emulators and their operating characteristics and instruction sets, it is necessary to have a special interface between the control processor and the emulator. The emulator dependent PROMs (configured as 8K X 8 bits) provide this interface. By reading these PROMs, the operating system has access to register information pertinent to that particular emulator. Message information and emulator specific utilities are also resident in the PROMs. Emulator dependent dump and restore and I/O mode information are also contained in these PROMs.

During the initialization routine the utility PROMs are disabled. They are enabled to allow the control processor to retrieve register information, emulator messages, or when the emulator is dispatched to utility routines. Upon completion, the utility PROMs are once again disabled.

The emulator module consists of the emulator card, emulator dependent PROMs on a separate PROM board, an emulation interface pod, prototype control probe (40-pin connector probe), and interconnecting flat cables. The emulator module is installed in the universal card cage in slot 5, which furnishes the system bus connection. Flat cables connect the control probe to the interface pod and the pod to the emulator card.

The function of the emulator board and pod is to replace the processor in the UUT with the same processor on the emulator card. This replacement allows all prototype programs to be executed by the emulator processor, under control of the 9508, to isolate a malfunctioning UUT processor. In addition, the emulator processor can exercise other circuits in the UUT, thus troubleshooting circuits in the UUT. Different emulator modules are available with the 9508 to replace different processors in the UUT.

COMMUNICATIONS/RAM MODULE

The Communications/RAM module (Comm/RAM) is the remote attachment for the 9508. It contains 4K-bytes of PROM residing in master memory space from D000 to DFFF.

The RAM consists of 8K-bytes of static memory, relocatable in 1, 2, 4, or 8K blocks (refer to figure 4-1). The location of RAM, the enabling and disabling functions and access is through I/O writes to the particular port. An additional 8K bytes of RAM are available with the inclusion of an optional Comm/RAM module inserted in slot 6 of the universal card cage.

Each RS-232 port also functions as a read/write port. All operating parameters including baud rate are programmed via I/O writes. The RS-232 operates in an interrupt driven mode. Availability to transmit a character is signified by an interrupt. Servicing this interrupt causes the control processor to read the status register, which contains transmit buffer empty condition. An interrupt, with a subsequent data available condition signifies a receive condition. Error conditions are also signaled by interrupt, with the nature of the error being available in the status register.

REAL-TIME TRACE MODULE

The real-time trace module (RTT) is designed to provide debugging tools helpful in all phases of microprocessor software and prototype hardware development. It is part of the real-time prototype analyzer subsystem, consisting of the real-time trace circuit assembly, a data acquisition probe interface assembly, and an active probe with 10 test clips. The real-time trace provides a buffer which stores the last 128 selected emulator bus operations as well as test clip data. It contains dual-event recognition logic, utilizing bus data, address, operation, and test clip data. For breakpoint or trigger purposes, the event recognition can be further conditioned by a pass count of n events and or a delay of n-clocks. A general purpose counter measures time, or other selected clocks between events.

The RTT module provides or supports the following debug related features:

1) Realtime storage and display of the last 128 selected bus transactions.

2) Dual breakpoint and trigger capability with event comparison further conditioned by pass count and delay requirements.

3) Eight external test clip states that are stored along with the bus transaction data and can be selectively included in the breakpoint equations.

4) General purpose count capability. Counts of real-time, events, bus transactions, trace stores, or instruction fetches can be made between two points in a program or between events.

INTRODUCTION

This chapter contains an overview of the software architecture in the 9508 MicroSystem Emulator, as well as a description of the specific operating principles of the emulator-host communications interface.

EMULATOR SOFTWARE ARCHITECTURE

The 9508 emulator program structure comprises the following basic elements:

1. An Executive Program
2. A Command Line Processor
3. A Series of Command Routines - one for each command in the 9508 command set
4. A Library of System Subroutines

These program elements are stored in PROM on the control processor card, the real-time trace card, and the PROM board. Figure 6-1 shows a functional block diagram of the 9508 emulator software.

The Executive program outputs the D> prompt to the console terminal and receives the command input from the terminal keyboard. After the Executive receives a command from individual keyboard character inputs, it passes the command to the Command Line Processor (CLP). The CLP calls up the command routine corresponding to the command input. To accomplish specific parts of their tasks, both the Executive and CLP call up system subroutines.

During the time when program control is with the Executive, the 9508 emulator is operating at command level.

As a command routine is executing, it can also call up any one of the system subroutines or it can pass control to another command routine, as shown by the broken lines in figure 6-1 between Rhex or Whex and the Terminal routine (i.e. either Rhex or Whex can invoke the Terminal pass through routine).

Various command routines can perform tasks of different scope. Many routines read or write in the emulation or UUT memory, in the target microprocessor registers, or I/O ports; others store and manipulate test parameters, or monitor bus activities. Virtually all routines output results to the console terminal for display. However, routines such as Assemble (Asm), Examine (Exam), or Register (REG), communicate directly and interactively with the console terminal, in the same manner as the Executive program. The Rhex, Whex and Terminal routines communicate interactively directly with the host. Note that on completion of execution every command routine returns control to the program that initially called it up.

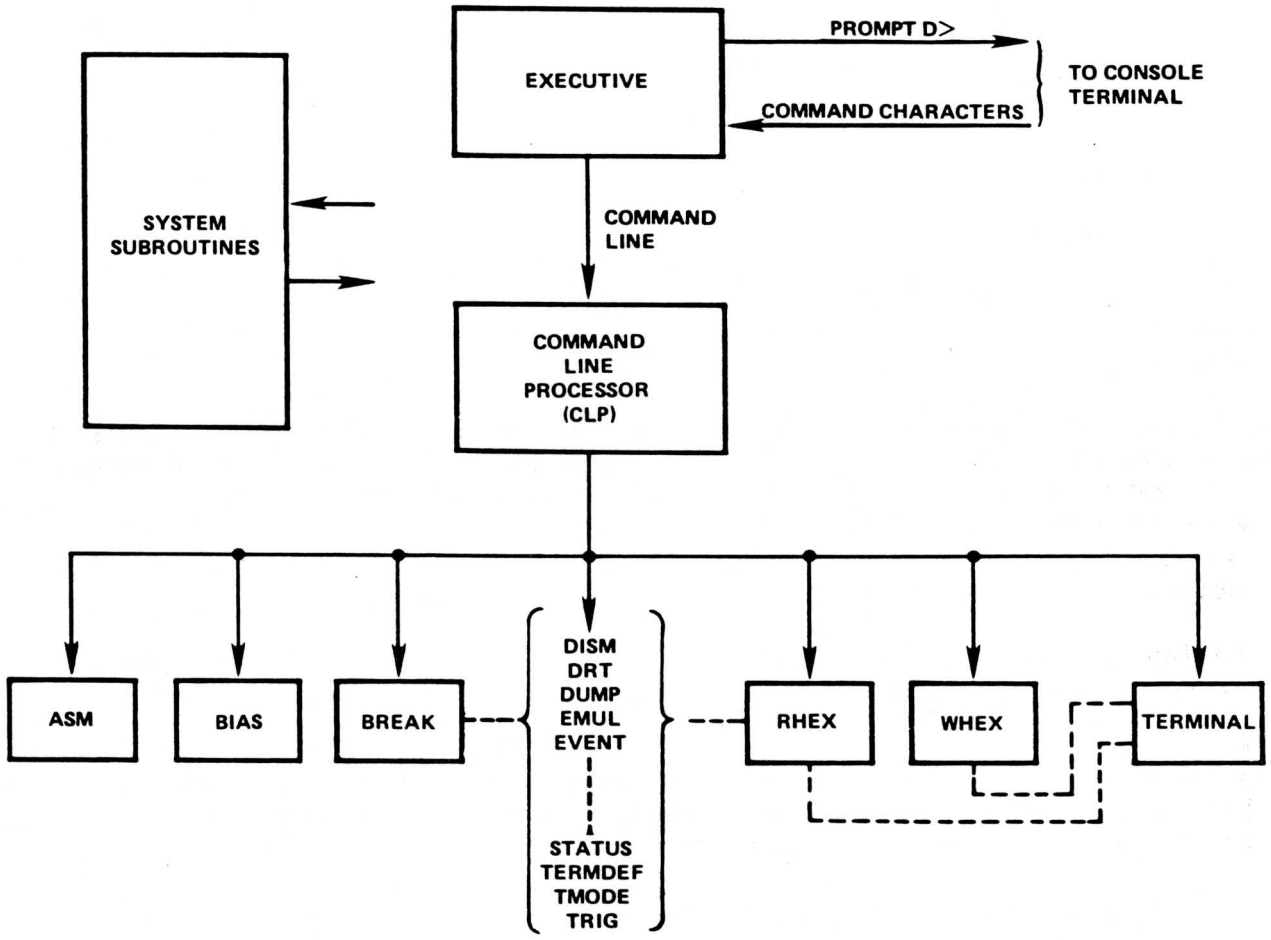


Figure 6-1. Block diagram of 9508 Emulator Software Architecture

EMULATOR-HOST COMMUNICATIONS

The communications link between the 9508 emulator and the host is used for downloading or uploading program material, or for general purpose interactive communications between the 9508 emulator console terminal and the host system. Downloading and uploading is carried out by the Rhex and Whex program routines in the emulator, working in conjunction with the Download and Upload utility modules in the host system. Interactive communications are carried out by the Terminal routine in the emulator, interfacing directly with the host operating system (or a communications interface program) in the host. Note that the Terminal routine may also be invoked in a subroutine capacity by the Rhex/Whex routines.

Data Formats.

The data to be transferred during downloading and uploading is grouped into data blocks. Two data block formats can be used with the 9508 emulator:

- o TekHex (Tektronix Hexadecimal Data Block Format)
- o Binary Data Block Format

The TekHex Data Block Format provides an industry standard for transferring data between systems. It allows data transfer between the 9520 and a user's PROM programmer.

Note: The 9508 and the 9520 (Upload and Download programs) use the Millennium Binary Data Block Format to reduce the number of characters transmitted, thus increasing the transmission speed by about 47%.

A special command (CONVERT) can be given to the 9520 Development System to convert an absolute object file, produced on the Millennium Assembler or Linker, to a TekHex (or Binary) format prior to the transfer. On other host systems a Convert utility program may have to be installed to perform the conversions.

The Binary Data Format is used to download and upload data between a 9520 host and 9508 emulator. The binary format is unique - it is compatible with Millennium Systems products, but may not be compatible with other manufacturer's equipment without special provisions. As with the TekHex format the Convert utility in the 9520 Software Development System can convert absolute object files produced by the Millennium Assembler or Linker into binary format, other host systems typically need a Convert utility program written and installed to perform this task.

Three types of blocks exist, for both the TekHex and Binary formats:

- o Data Block
- o Termination Block
- o Abort Block

The significant information transferred during a download or upload is contained in the data blocks. Each data block contains up to 72 ASCII characters and a series of blocks is typically used to accomplish the transfer. A termination block signifies the end of data. It is the last block in a transfer and does not contain any data except the execution starting address of the code contained in the data blocks. Aborting a transfer can be called for at any time by means of an abort block. The abort block does not contain any data except a message stating the reason for aborting. This can be a string of up to 69 ASCII characters.

Appendix C describes each data format and each block type separately, provides rules for using them, and thus constitutes specifications for writing the Convert utility program.

Downloading, Uploading, and Interactive Communications.

Whenever the RHex or WHex command is entered, the CLP calls up the Rhex or Whex routine in the 9508 emulator, the active routine interacts with the operator, as well as the host to activate the Download or Upload utility module. Download or Upload then performs the data transfer operation. When the interactive communications mode is activated with the TERMINAL command, the Terminal routine in the 9508 is called up. Communication is directly between the 9508 console terminal and the host operating system and/or an interface program module in the host.

The exact sequence of steps required to activate Download or Upload depends on the parameters entered with the RHex or WHex command. The sequences of steps corresponding to different parameters are described as three separate procedures in section II of chapter 3, but an overview of how programs in the 9508 emulator and the host interact during each of the three procedures (Cases 1, 2, and 3) is described as follows. Program interaction is also described for the interactive communications mode.

Case 1. If the Case 1 procedure is specified with RHex or WHex, the 9508 emulator goes directly into the transfer mode without any commands to the host (except for the synchronization character, if invoked with the command). Thus, activating the Upload or Download utility in the host remains with the host operator using a host console. Downloading or uploading continues as under Case 2 or Case 3 procedure.

Case 2. If the Case 2 procedure is specified with RHex or WHex command, the corresponding emulator routine constructs the DOWNLOAD or UPLOAD command to the host and thus activates the Upload or Download utility. The 9508 emulator switches into the transfer mode and the uploading or downloading occurs exactly as described in the Case 3 procedure.

Case 3. If the Case 3 procedure is specified with either the RHex or the WHex command, the Terminal routine is invoked and the communications link operates in the interactive communication mode until such time when the operator enters either an UPLOAD or DOWNLOAD command. At this time, the Upload or Download utility is activated in the host, the interactive communication mode is aborted and the link begins to operate in the transfer mode. The transfer mode remains active until downloading or uploading is completed (as indicated by the presence of a termination block in the data stream). At this time, control returns to the Executive in the 9508 emulator and usually to the operating system in the host.

Interactive Communications Mode. The interactive communications (terminal pass through) mode is activated with the TERMINAL command. The CLP calls up the Terminal subroutine, which sends out a CR to the host to prompt its operating system. From there on, the link operates in the interactive communications mode until a Control \ is detected, causing control in the emulator to be returned to the Executive. Note that neither DOWNLOAD or UPLOAD commands may be sent from the emulator while in the interactive communications mode, because either one will cause a Control \ to occur on the link and thus abort the mode. The Control \ sets the 9508 at the command level while a download/upload program is active in the host.

Data Transfer Protocol

The 9508 emulator can support either of the following handshake protocols during data transfers across the emulator-host communications link:

- Electrical - DSR/DTR (Data Set Ready/Data Terminal Ready)
- ACK/NAK (Acknowledgement/Negative Acknowledgement)

Either of these protocols, or no protocol at all, is selected with the Linkdef command (ACK/NAK is the initial default value). The two protocols are described separately.

Regardless of which protocol is used, the 9508 Emulator checks each data block for the following faults:

1. Loss of Data Carrier Detect Signal
2. Parity (Odd or Even)
3. Data Framing
4. Data Overrun
5. Address or Data Checksum does not match
6. Invalid TekHex Character (during Rhex only, TekHex format).

One or more faults (depending on the protocol in use) causes an error condition. An error condition also occurs if successive data blocks are not received within an established timeout period. The timeout period is set with the Linkdef command and its default value in 65,535 (the unit of measurement is 100 milliseconds). In addition, an error condition will occur if during downloading the Rhex subroutine detects a memory write failure or during uploading the Whex subroutine detects an invalid memory read address.

Each error is classified as non-recoverable or recoverable:

Non-recoverable	Recoverable
o Loss of Data Carrier Detect	o Checksum Error
o Memory Write Error	o Parity Error
o Invalid Memory Address	o Framing Error
	o Data Overrun
	o Invalid TekHex Character
	o Link Timeout

In case of a non-recoverable error, the program subroutine in control (Rhex or Whex) aborts and displays an error message on the lower left of the console screen:

ERROR RC = hhhh

where hhhh is two hexadecimal bytes, as defined in chapter 4 under the TERMINAL command. Program control reverts back to the Executive, which displays the following error message on lower right of the screen:

ERROR 40

(Error 40 = Communications Link Failure)

The Rhex or Whex subroutines also abort when the ESCape key is depressed. The Whex subroutine causes the host program to abort by sending an abort data block, while the Rhex subroutine aborts the host program by transmitting the host abort character. The initial value of the host abort character is CONTROL C (03H), but it can be redefined by the Linkdef command.

In case of a recoverable error, the response depends on the action-on-error parameter selected with the Linkdef command - either the program subroutine aborts (initial default value) or ignores the error. If it ignores the error, it attempts to perform the transfer again and also displays the ERROR RC=hhhh message. If it aborts, it follows the same program logic and displays the same two error messages as previously described.

ACK/NAK Protocol. If the ACK/NAK protocol is selected, each data block that is correctly received is acknowledged by:

ACK <CR>

When a fault is detected in the received data block, retransmission is requested by sending:

NAK <CR>

The receiving station requests retransmission up to five times, but five successive faults are treated as an error and reported to the program routine.

Electrical Protocol. The electrical or DSR/DTR protocol is always active. Whenever the 9508 cannot receive anymore data, it deactivates the DSR signal. If the 9508 is transmitting data, transmission will stop when the host deactivates its DTR signal. As far as the 9508 is concerned, the DSR/DTR protocol is always active. However, for this protocol to be effective, it must be supported by the host.

ERROR CODES

ERROR	DESCRIPTION
FF	INVALID COMMAND
02	INVALID PARAMETER
03	INVALID KEYWORD - Command syntax required the user to select one option from a list. No match was found in the list of valid options.
04 or 0D	PARAMETER REQUIRED - A parameter was required, but not specified
05	TOO MANY PARAMETERS - more parameters were specified than were expected by the command.
06	INVALID OPERATION
07	RAM 2 NOT INSTALLED
08	DELAY COUNT ERROR - Invalid value for delay counter.
09	INVALID SYMBOLIC ADDRESS FORMAT
10	INVALID HEXADECIMAL CHARACTER - A non-hexadecimal character or too large a number was entered.
11	ADDRESS OUT OF RANGE - invalid for the processor.
12	SYSTEM MODE - not allowed for user I/O or Memory.
13	INVALID ASCII STRING
14	LOWER ADDRESS IS NOT LESS THAN OR EQUAL TO UPPER ADDRESS
15	(Not Used)
16	INVALID DECIMAL NUMBER - A non-decimal character was typed.
17	DECIMAL NUMBER OUT OF RANGE
18	INVALID BINARY CHARACTER
19	BINARY VALUE OUT OF RANGE
20	MEMORY BLOCK 2 NOT IN SYSTEM - The user attempted to map memory using block 2, but block 2 memory (optional) did not exist in the system.
21-23	(Not Used)

ERROR	DESCRIPTION
24	INVALID MAPPING ADDRESS - The address is not on a 1, 2, 4, or 8K boundary, or is outside the address space of the processor.
25	SYMBOLS ERROR
26-29	(Not Used)
30	MEMORY WRITE ERROR
31	(Not Used)
32	NO MORE ROOM IN ONBRK BUFFER
33	(Not Used)
34	INVALID PORT NUMBER
35-39	(Not Used)
40	COMMUNICATIONS LINK FAILURE

STARTUP ERRORS

81	MASTER SYSTEM RAM FAILURE
82	EMULATOR UTILITY NOT FOUND
83	MASTER SYSTEM PROM FAILURE
84	EMULATOR DOES NOT RUN
85	MAPPED RAM BLOCK 1 FAILURE
86	MAPPED RAM BLOCK 2 FAILURE
87	EMULATOR SHADOW RAM FAILURE
88	MASTER SYSTEM RAM 2 FAILURE
89	RTT NOT CONNECTED

REASONS FOR STOPPING EMULATION - MESSAGES DISPLAYED ON CONSOLE TERMINAL

Register Break Point	Trigger 1 Break Point
Until Break Point	Trigger 2 Break Point
Emulator Fault	Step Count Complete
Emulator Halt	Trace Count Complete

9508 SYSTEM COMMAND SUMMARY

The following is a list of 9508 emulator commands, with syntax, both in alphabetical order, as well as in functional grouping. Detailed description of all commands is in chapter 4. Each command must be entered as a single line, in response to the system prompt D>, and ending with a carriage return (CR). The following symbols and conventions are used to describe command syntax:

{ } are used to enclose a required parameter, meaning that if it is not entered, the system will respond with an error message or attempt to execute.

[ ] are used to enclose an optional parameter. If it is not entered the system will invoke a fixed default value, use the last value entered by the operator, or simply fail to perform the function represented by that parameter.

{T1 / T2 / Both} are stacked parameters, indicating that one of the parameters must be chosen.

A default value of any parameter is shown underlined. Refer to chapter 4 for a more detailed description of the symbols and conventions used for defining the syntax.

ALPHABETICAL LISTING

COMMAND NAME	SYNTAX
A)sm	Asm [saddr]
Bl)as	Blas [W=haddr X=haddr Y=haddr Z=haddr]
BR)eak	BReak {T1 / T2 / Both} [Disable] [Cont]
C)ount	Count [clear] [Ms / Us / Bus / Emclk / Fetch / Rtt / E1 / E2]
Dl)sm	Dlsm [laddr] [uaddr / N=nnnnn]

ALPHABETICAL LISTING (Cont'd)

COMMAND NAME	SYNTAX
DR)†	DR† [Go | nnn]
D)ump	Dump [laddr] [uaddr]
EM)ul	EMul [S | U]
EV)ent	EVent { 1 | 2 } [clear] [A=Haddr][D=HH][B= A | F | IR | IW | M | MR | MW | R | W][E=cccccccc]
E)xam	Exam [saddr]
F)ill	Fill {laddr} {uaddr} { hexstring | 'char-string }
G)o	Go [saddr1] [Until saddr2 [R | W] | Step [nnnnn]] [Trace { All | Jmp | Clear } [Count nnnnn | From saddr3 | to saddr4]]
L)inkdef	Linkdef [clear] [B=110 | 300 | 600 | 1200 | 2400 | 4800 | 9600] [D=H | F] [Fmt=Hex | Bin] [P=N | O | E] [PR=N | Ack | Dsr] [Ack=hh] [Nak=hh] [Time=nnnn] [Err=I | A] [Sync=hh] [Habt=hh[[TUrn=nnnnn]

ALPHABETICAL LISTING (Cont'd)

COMMAND NAME	SYNTAX
MA)p	MAp [{ 1 {haddr} / 2 / Clear [1 / 2 / Both] } [n]]
MO)ve	MOve { UU / US / SU / SS } {laddr} {uaddr} {saddr}
O)nbrk	Onbrk [clear]
P)ort	Port {haddr} [hh] [Rep]
Q)ual	Qual [All / Fetch / I / IR / IW / M / MR / MW / R / W]
RES)et	RESet
REG	REG [keyword]
REGB)rk	REGBrk [clear] [condition1...condition 10]
RH)ex	RHex [Nosync] { TErm / TRans / 'String }

ALPHABETICAL LISTING (Cont'd)

COMMAND NAME	SYNTAX
S)tatus	Status [Env | Reg]
TE)rmdef	TErmdef [keyword=hh]
TERMINAL	TERMINAL
TM)ode	TMode [Ind | E12 | Arm | Frz]
TR)ig	TRig {1 | 2} [P=0 | n] [D=0 | n]
WH)ex	WHex [Nosync] {saddr} {laddr1} {uaddr1} ...[laddr4 uaddr4] {TErm | TRans | 'String}

FUNCTIONAL LISTING

Memory Display and Modification Commands

ASM	A)sm	Modifies memory using the in-line assembler.
DISM	DI)sm	Displays memory using the in-line disassembler.
DUMP	D)ump	Displays memory in hexadecimal and ASCII.
EXAM	E)xam	Displays one memory location at a time and pauses for modification.
FILL	F)ill	Fills a memory range with a specified data byte.
MOVE	MO)ve	Moves a block of data from one location to another.

Memory Management Commands

BIAS	BI)as	Assigns values to the four base address registers W, X, Y, and Z.
MAP	MA)p	Maps segments of the target microprocessor address space into 9508 emulation memory.

Communication Control Commands

LINKDEF	L)inkdef	Define characteristic of emulator-host communications link.
RHEX	RH)ex	Reads a program from the 9520 development system, or host minicomputer for downloading into the emulator or UUT.
WHEX	WH)ex	Transmits programs or data from the 9508 to the 9520 development system or host minicomputer (performs uploading).
TERMINAL	TERMINAL	Enables the interactive communications (terminal passthrough) mode between the console terminal and the host.

Register and I/O Port Commands

PORT	P)ort	Reads from and displays, or writes to, the target microprocessor I/O port.
REG	REG	Displays and modifies target microprocessor registers.
STATUS	S)tatus	Displays all target microprocessor registers and flags, and emulation environment status.

Logic Analyzer, Trigger, Breakpoint Commands

QUAL	Q)ual	Selects type of data to be recorded during user's program execution in RTT buffer.
DRT	DR)t	Displays trace buffer contents.
EVENT	EV)ent	Defines events for trigger/breakpoint equations.
TRIG	TR)ig	Selects pass count, delay, and trigger mode.
BREAK	BR)eak	Selects the trigger equation on which emulation is halted and trace line displayed.
ONBRK	ON)brk	Used to specify a list of commands to be executed automatically at a breakpoint.
REGBRK	REGB)rk	Used to define a register breakpoint as determined by one to eight test conditions.
COUNT	C)ount	Selects units to be counted by general purpose counter.
TMODE	TM)ode	Defines relationship of trigger 1/trigger 2 and EVent1/EVent2 (selects trigger mode)

Emulation Control Commands

EMUL	EM)ul	Specifies emulation mode; user or system.
RESET	RES)et	Resets target microprocessor.
GO	G)o	Defines starting address for user's program execution, selects single step or realtime execution, sets a simple breakpoint, and enables single step trace.

Terminal Control Commands

TERMDEF	TE)rmdef	Changes the output characters for special function keys; displays the current setting.

CONVERT, DOWNLOAD, AND UPLOAD PROGRAM SPECIFICATIONS

CONVERT UTILITY PROGRAM REQUIREMENTS

The Convert Utility is installed on the host system to convert files from any other format to a TekHex, or Binary data format prior to their downloading to the 9508 emulator. The following paragraphs explain the data and data block formats, both TekHex and Binary, that the Convert utility must generate.

Tektronix Hexadecimal Format

The Tektronix Hexadecimal Data Block Format is a set of message blocks used for encoding data in ASCII form. This format encodes data using a subset (0-9, A-F) of the ASCII character set. This data can be downloaded to the 9508 emulator using the RHex command, or can be uploaded and written to a file or device in the host, using the WHex command.

The message blocks fall under three classes: data blocks, termination blocks, and abort blocks. Data is encoded as a series of one or more data blocks. Normal termination of the data is indicated by one termination block. Abnormal termination can be indicated at any point by means of an abort block. The specific format for each of these types of blocks is described in tables C-1, C-2, and C-3.

General rules for using TekHex format are as follows:

1. Each message block must begin with a slash (/) as a header character.

2. Each message block is terminated by a carriage return.

3. All characters in every block must be printable ASCII characters.

4. All characters except header characters, error information (in the abort block), and carriage return must be ASCII encoded hexadecimal digits.

5. Each message block is limited to a maximum of 72 characters.

6. Characters prior to a slash (/) are ignored unless a <CR> is received. Non-hexadecimal characters in the hex portion of the data block are treated as an error or fault.

7. A termination block or an abort block must be the last block following a series of data blocks.

8. The TekHex format is defined only as described in this appendix; it must not be confused with other hexadecimal formats (e.g., Intel's Hexadecimal Format).

Table C-1. Tektronix Hexadecimal Format Data Block

PURPOSE: Encodes data bytes as a series of hexadecimal digits.

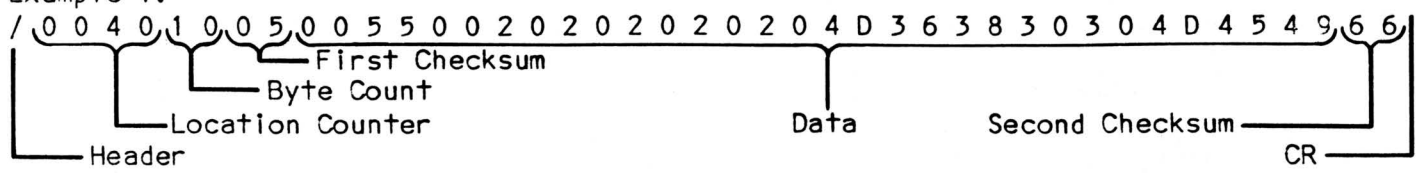
General Format

(Header)	Location Counter	Byte Count	First Checksum	Data	Second Checksum	CR (EOL)

Format Description

NAME	NO. OF ASCII CHARACTERS	CONTENT DESCRIPTION
Header	1	Always a slash (/).
Location Counter	4	Four hexadecimal digits representing the starting memory location of the block.
Byte Count	2	Two-digit hexadecimal value specifying the number of data bytes in the data field of block.
First Checksum	2	Two-digit hexadecimal number representing the hexadecimal sum of the values of the six digits that make up the location counter and the byte count.
Data	2*N	N data bytes, each represented as two hexadecimal digits. Each hex digit is coded as an ASCII character 0-9 or A-F. There can be a maximum of 30 (decimal) data bytes (60 hex digits) per block.
Second Checksum	2	Two-digit hexadecimal number representing the sum, modulo 265, of the hexadecimal values of the digits that make up the N data bytes.
CR	1	Always a carriage return (CR) indicating the end of the block.

Example 1:



Explanation of Example 1: Location Counter 0040 specifies that the first data byte (00) is located at memory address 0040. The next bytes (55, 00, 20...) are located at the next contiguous memory locations (0041, 0042, etc.)

Byte Count 10 indicates there are 10H data bytes, represented as 20H hexadecimal digits. In decimal, this is a total of 16 data bytes represented as 32 digits.

First Checksum 05 is the sum of the hexadecimal digits making up the location counter and the byte count. That is, 0 + 0 + 4 + 0 + 1 + 0 = 05

Data Bytes - each hexadecimal digit is an ASCII character; each pair of digits represents a data byte in memory. For example, the hexadecimal digits 55 represent the data byte 01010101 (binary).

Second Checksum 66 is the sum of the hexadecimal digits making up the data:
0 + 0 + 5 + 5 + 0 + 0 + 2 + 0 + 2 + 0 + 2 + 0 + 2 + 0 + 2 + 0 + 4 + D + 3 + 6
+ 3 + 8 + 3 + 0 + 3 + 0 + 4 + D + 4 + 5 + 4 + 9 = 66H

CR is not printable.

Table C-2. Tektronix Hexadecimal Format Termination Block

PURPOSE: Specifies the end of data (no more data blocks follow).

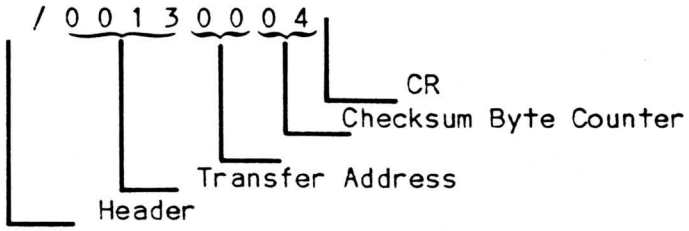
General Format

(Header)	Transfer Address	Byte Count	Checksum	CR (EOL)

Format Description

NAME	NO. OF ASCII CHARACTERS	CONTENT DESCRIPTION
Header	1	Always a slash (/).
Transfer Address	4	Four hexadecimal digits may represent the starting execution address of the code represented in the data blocks.
Byte Count	2	Always 00 identifying a termination block.
Checksum	2	Two-digit hexadecimal number representing the sum, modulo 256, of the hexadecimal values of the six digits that make up the transfer address and the byte count.
CR	1	Always a carriage return.

Example 2:



Explanation:

Transfer Address 0013 specifies that the starting execution address is 0013.

Byte Count 00 indicates that this is a termination block.

Checksum 04 this is the sum of the hexadecimal digits making up the transfer address and the byte count.

0 + 0 + 1 + 3 + 0 + 0 = 04

CR is not printable.

Table C-3. Tektronix Hexadecimal Format Abort Block

PURPOSE: Indicates an abnormal termination of the data blocks.

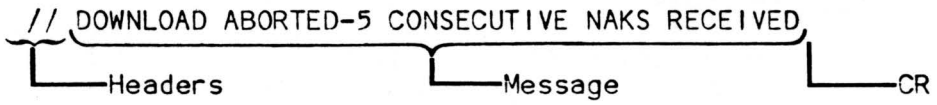
General Format

(Header)	(Header)	Message	CR (EOL)

Format Description

NAME	NO. OF ASCII CHARACTERS	CONTENT DESCRIPTION
Header	1	Always a slash (/).
Header	1	Another slash (/) to identify an abort block.
Message	N	N ASCII characters (a maximum of 69 characters). This is a string of characters making up the error information or message.
CR	1	Always a carriage return.

Example 3:



Binary Message Format

The Binary Data Block Format is similar to the TekHex format, the major difference being that the character set is not restricted to printable ASCII characters. The data transmission can occur at a greater speed since the data does not have to be converted from binary to ASCII.

The binary message blocks fall under three classes: data blocks, termination blocks, and abort blocks. The format of these message blocks is given in tables C-4, C-5, and C-6 respectively.

Table C-4. Binary Format Data Block

PURPOSE: Encodes data bytes as a series of binary numbers.

General Format

(Header)	Location Counter	Byte Count	First Checksum	Data	Second Checksum	CR (EOL)

BYTE	CONTENTS
1-2	<DLE> ASCII slash (/) 2FH.
3-4	LOCATION COUNTER. (Unsigned binary number in the range 0-FFFF.)
5	BYTE COUNT. (Unsigned binary number in the range 0-3CH.)
6	FIRST CHECKSUM. (Checksum of the LOCATION COUNTER and BYTE COUNT.)

NOTE: If a binary DLE is intentionally transmitted in the data string the DLE must be transmitted twice. However, the second DLE does not effect the byte count.

7-N+7	N DATA BYTES. (Where N is the value of the byte count.)
N+8	SECOND CHECKSUM. (Checksum of the N data bytes.)
N+9, N+10	<DLE> EOL CHARACTER = CARRIAGE RETURN.

Table C-5. Binary Format Termination Block

PURPOSE: Specifies the end of data; no more data blocks follow. The format of the termination block is similar to the data block except that the byte count is 0; therefore there are no data bytes or second checksum.

General Format

(Header)	Transfer Address	Byte Count	Checksum	CR (EOL)

BYTE	CONTENTS
1-2	<DLE> ASCII slash (/) 2FH.
3-4	TRANSFER ADDRESS. (The address at which execution will begin.)
5	BYTE COUNT = 0
6	CHECKSUM: (Checksum of the transfer address and byte count.)
7-8	<DLE> EOL CHARACTER = CARRIAGE RETURN (0DH)

Table C-6. Binary Format Abort Block

PURPOSE: Indicates an abnormal termination of the data blocks.

General Format

(Header)	0000	COUNT = FF	Checksum = 1EH

BYTE	CONTENTS
1,2	<DLE>/
3,4	0000
5	BYTE COUNT = FF
6	FIRST CHECKSUM = 1EH

CHECKSUM ALGORITHM:

1. CKSUM = 1EH
2. As each byte is received, add each nibble (without carry) to CKSUM.
3. Note that the <DLE> character in the heading and before the ending <CR> is not included in the checksums.

DOWNLOAD/UPLOAD UTILITY PROGRAM REQUIREMENTS

The Download and Upload utility programs are installed on the host system to transfer files from the host system to the emulator, or to receive and store files transmitted from the emulator to the host. The following paragraphs describe the Download and Upload program requirements, depending on the type of physical connection between the host and emulator (console or I/O), and the format of the data to be transferred (TekHex or Binary). Each case is organized generally as it would occur in a chronological sequence.

Download - TekHex, Console Connection.

1. Send a Control \ character to emulator (this forces the emulator to exit the interactive communications mode).

2. Synchronization (Optional). Perform program synchronization by waiting for:

   S <CR>

   and responding with:

   T <CR>

3. Read next record from the file and send it to the emulator.

4. ACK/NAK Protocol (Optional). Following transmission of each data block wait for any one of the following:

   ACK <CR> (ACK=ASCII 0 CR=300D)
   NAK <CR> (NAK=ASCII 7 CR=370D)
   Host Abort Character <CR>

   If NAK, then retransmit the same block and wait again; if ACK, transmit next block; if host abort character, close file and terminate the program. If the terminating block has been transmitted, close files and terminate the program following the receipt of ACK.

5. If an error occurs at any time while reading the file, transmit the abort block.

Download - TekHex, I/O Connection.

1. Synchronization (Optional). Perform program synchronization by waiting for:

   S <CR>

   and responding with:

   T <CR>

2. Read next record from the file and send it to the emulator.

3. ACK/NAK Protocol (Optional). Following transmission of each data block wait for any one of the following:

   ACK <CR>
   NAK <CR>
   Host Abort Character <CR>

   If NAK, then retransmit the same block and wait again; if ACK, transmit next block; if host abort character, close file and terminate the program. If the terminating block has been transmitted, close files and terminate the program following the receipt of ACK.

Upload - TekHex, Console Connection

1. Send a Control \ character to emulator (this forces the emulator to exit the interactive communications mode.

2. Synchronization (Optional). Perform program synchronization by waiting for:

   S <CR>

   and responding with:

   T <CR>

3. Read the next block. If there are no errors, write the block to the disk.

4. ACK/NAK Protocol (Optional). Following the receipt of a block, send out any one of the following:

   ACK <CR> - if block has no errors

   NAK <CR> - if there is a checksum error in the block

   Host abort character - if a disk write error has occurred

5. When a termination block is received perform step 4, then close the files and terminate the program.

Upload - TekHex, I/O Connection

The procedure is identically the same as Upload - TekHex, Console Connection, except the Control \ is not sent to the emulator (step 1 of the procedure).

Download - Binary, Console Connection

All steps are the same as during Download - TekHex, Console Connection, however, the details of how the characters are transferred differ. For example, given the data record:

Header	addr	Count	CK1	Data	CK2	CR
2F	0110	03	05	871069	2E	0D

it will be transmitted as:

<DLE>2F01<DLE>10030587<DLE>10692E<DLE><CR>

Note that:

- The header 2F is transmitted as <DLE>2F
- The terminating carriage return is transmitted as <DLE>0D
- If the DLE character (10H) occurs anywhere in the body of the record, it is transmitted as

  <DLE>10

  (i.e., it is transmitted twice)

Download - Binary, I/O Connection

The procedure is similar to the Download - TekHex, I/O Connection, except for the details on how the <DLE> character is transferred (see Download - Binary, Console Connection for an example).

Upload - Binary, Console Connection

All steps are the same as for Upload - TekHex, Console Connection, however, for each DLE character that occurs in the body of the block, it is transmitted as

<DLE>10

(Refer to Download - Binary, Console Connection, for an example)

Upload - Binary, I/O Connection

The procedure is similar to the Upload - TekHex, Console Connection, except the Control \ is not sent to the emulator (step 1 of the procedure) and the details of how the <DLE> character is transmitted differ (see Download - Binary, Console Connection, for an example).

ASCII CODE CONVERSION TABLE

		HEXADECIMAL MOST SIGNIFICANT CHARACTER							
	—	0	1	2	3	4	5	6	7
	0	NUL	DLE	SP	0	@	P	`	p
	1	SOH	DC1	!	1	A	Q	a	q
	2	STX	DC2	"	2	B	R	b	r
	3	ETX	DC3	#	3	C	S	c	s
	4	EOT	DC4	$	4	D	T	d	t
LEAST	5	ENQ	NAK	%	5	E	U	e	u
SIGNIFICANT	6	ACK	SYN	&	6	F	V	f	v
CHARACTER	7	BEL	ETB	'	7	G	W	g	w
	8	BS	CAN	(	8	H	X	h	x
	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[	k	{
	C	FF	FS	,	<	L	\	l	¦
	D	CR	GS	-	=	M	]	m	}
	E	SO	RS	.	>	N	^	n	≈
	F	SI	US	/	?	O	—	o	DEL

EXAMPLES

W = 57
H = 48
a = 61
t = 74
@ = 40
NUL = 00
DEL = 7F

CONVERT, DOWNLOAD, AND UPLOAD PROGRAM SPECIFICATIONS

CONVERT UTILITY PROGRAM REQUIREMENTS

The Convert Utility is installed on the host system to convert files from any other format to a TekHex, Intel Hex, Motorola Hex, or Binary data format prior to their downloading to the 9508S emulator. The following paragraphs explain the data and data block formats, TekHex, Intel Hex, Motorola Hex and Binary, that the Convert utility must generate.

Tektronix Hexadecimal Format

The Tektronix Hexadecimal Data Block Format is a set of message blocks used for encoding data in ASCII form. This format encodes data using a subset (0-9, A-F) of the ASCII character set. This data can be downloaded to the 9508S emulator using the RHEX command, or can be uploaded and written to a file or device in the host, using the WHEX command.

The message blocks fall under three classes: data blocks, termination blocks, and abort blocks. Data is encoded as a series of one or more data blocks. Normal termination of the data is indicated by one termination block. Abnormal termination can be indicated at any point by means of an abort block. The specific format for each of these types of blocks is described in tables C-1, C-2, and C-3.

General rules for using TekHex format are as follows:

1. Each message block must begin with a slash (/) as a header character.

2. Each message block is terminated by a carriage return.

3. All characters in every block must be printable ASCII characters.

4. All characters except header characters, error information (in the abort block), and carriage return must be ASCII encoded hexadecimal digits.

5. Each message block is limited to a maximum of 72 characters.

6. Characters prior to a slash (/) are ignored unless a <CR> is received. Non-hexadecimal characters in the hex portion of the data block are treated as an error or fault.

7. A termination block or an abort block must be the last block following a series of data blocks.

8. The TekHex format is defined only as described in this appendix; it must not be confused with other hexadecimal formats (e.g., Intel's Hexadecimal Format).

Table C-1. Tektronix Hexadecimal Format Data Block

PURPOSE: Encodes data bytes as a series of hexadecimal digits.

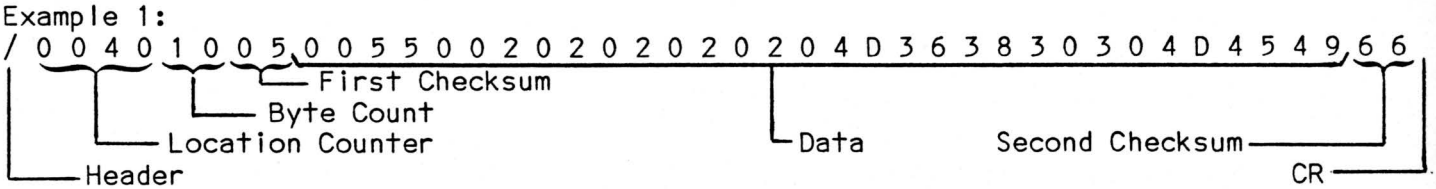
General Format

(Header)	Location Counter	Byte Count	First Checksum	Data	Second Checksum	CR (EOL)

Format Description

NAME	NO. OF ASCII CHARACTERS	CONTENT DESCRIPTION
Header	1	Always a slash (/).
Location Counter	4	Four hexadecimal digits representing the starting memory location of the block.
Byte Count	2	Two-digit hexadecimal value specifying the number of data bytes in the data field of block. The maximum count is 1EH.
First Checksum	2	Two-digit hexadecimal number representing the hexadecimal sum of the values of the six digits that make up the location counter and the byte count.
Data	2*N	N data bytes, each represented as two hexadecimal digits. Each hex digit is coded as an ASCII character 0-9 or A-F. There can be a maximum of 30 (decimal) data bytes (60 hex digits) per block.
Second Checksum	2	Two-digit hexadecimal number representing the sum, modulo 265, of the hexadecimal values of the digits that make up the N data bytes.
CR	1	Always a carriage return (CR) indicating the end of the block.

Example 1:



Explanation of Example 1: Location Counter 0040 specifies that the first data byte (00) is located at memory address 0040. The next bytes (55, 00, 20...) are located at the next contiguous memory locations (0041, 0042, etc.)

Byte Count 10 indicates there are 10H data bytes, represented as 20H hexadecimal digits. In decimal, this is a total of 16 data bytes represented as 32 digits.

First Checksum 05 is the sum of the hexadecimal digits making up the location counter and the byte count. That is, 0 + 0 + 4 + 0 + 1 + 0 = 05

Data Bytes - each hexadecimal digit is an ASCII character; each pair of digits represents a data byte in memory. For example, the hexadecimal digits 55 represent the data byte 01010101 (binary).

Second Checksum 66 is the sum of the hexadecimal digits making up the data:
0 + 0 + 5 + 5 + 0 + 0 + 2 + 0 + 2 + 0 + 2 + 0 + 2 + 0 + 2 + 0 + 4 + D + 3 + 6 + 3 + 8 + 3 + 0 + 3 + 0 + 4 + D + 4 + 5 + 4 + 9 = 66H

CR is not printable.

Table C-2. Tektronix Hexadecimal Format Termination Block

PURPOSE: Specifies the end of data (no more data blocks follow).

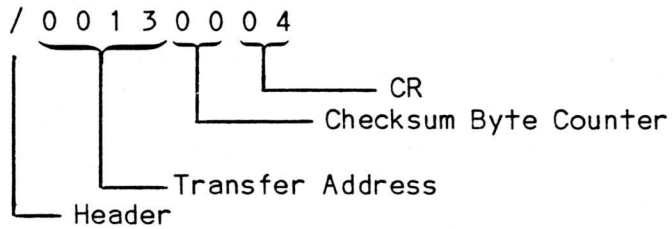
General Format

(Header)	Transfer Address	Byte Count	Checksum	CR (EOL)

Format Description

NAME	NO. OF ASCII CHARACTERS	CONTENT DESCRIPTION
Header	1	Always a slash (/).
Transfer Address	4	Four hexadecimal digits may represent the starting execution address of the code represented in the data blocks.
Byte Count	2	Always 00 identifying a termination block.
Checksum	2	Two-digit hexadecimal number representing the sum, modulo 256, of the hexadecimal values of the six digits that make up the transfer address and the byte count.
CR	1	Always a carriage return.

Example 2:



Explanation:

Transfer Address 0013 specifies that the starting execution address is 0013.

Byte Count 00 indicates that this is a termination block.

Checksum 04 this is the sum of the hexadecimal digits making up the transfer address and the byte count.

0 + 0 + 1 + 3 + 0 + 0 = 04

CR is not printable.

Table C-3. Tektronix Hexadecimal Format Abort Block

PURPOSE: Indicates an abnormal termination of the data blocks.

General Format

(Header)	(Header)	Message	CR (EOL)

Format Description

NAME	NO. OF ASCII CHARACTERS	CONTENT DESCRIPTION
Header	1	Always a slash (/).
Header	1	Another slash (/) to identify an abort block.
Message	N	N ASCII characters (a maximum of 69 characters). This is a string of characters making up the error information or message.
CR	1	Always a carriage return.

Example 3:

```
// DOWNLOAD ABORTED-5 CONSECUTIVE NAKS RECEIVED
└Headers  └Message                            └CR
```

Binary Message Format

The Binary Data Block Format is similar to the TekHex format, the major difference being that the character set is not restricted to printable ASCII characters. The data transmission can occur at a greater speed since the data does not have to be converted from binary to ASCII.

The binary message blocks fall under three classes: data blocks, termination blocks, and abort blocks. The format of these message blocks is given in tables C-4, C-5, and C-6 respectively.

Table C-4. Binary Format Data Block

PURPOSE: Encodes data bytes as a series of binary numbers.

General Format

(Header)	Location Counter	Byte Count	First Checksum	Data	Second Checksum	CR (EOL)

BYTE	CONTENTS
1-2	<DLE> ASCII slash (/) 2FH.
3-4	LOCATION COUNTER. (Unsigned binary number in the range 0-FFFF.)
5	BYTE COUNT. (Unsigned binary number in the range 0-3CH.)
6	FIRST CHECKSUM. (Checksum of the LOCATION COUNTER and BYTE COUNT.)

NOTE: If a binary DLE occurs between the location counter and the second checksum (inclusive), then the DLE must be transmitted twice. However, the second DLE does not effect the byte count, and it is not included in the checksum.

7-N+7	N DATA BYTES. (Where N is the value of the byte count.)
N+8	SECOND CHECKSUM. (Checksum of the N data bytes.)
N+9, N+10	<DLE> <CR>

Table 3-4. Binary Format Data Block (Cont'd)

Checksum #1, CK1, is calculated by adding with carry the nibbles in the ADDRESS and COUNT fields.

Checksum #2, CK2, is calculated by adding with carry the nibbles in the DATA field.

EXAMPLE 1: DATA BLOCK

HEADER	ADDR	COUNT	CK1	DATA	CK2	EOL
102F	0110	03	05	871069	1F	100D

CK1 = 05 = 0+1+1+0+0+3
CK2 = 1F = 8+7+1+0+6+9

Because of the rule for DLE characters (10H) between the HEADER and EOL, this record would be transmitted as 10 2F 01 10 10 03 05 87 10 10 69 1F 10 0D

Note that the two extra DLE characters are not included in the checksums.

EXAMPLE 2: DATA BLOCK

102F 1023 12 09 FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 1C 100D

CK1 = 1+0+2+3+1+2 = 09
CK2 = F+Fn.....F+F=1C

n=36

Table C-5. Binary Format Termination Block

PURPOSE: Specifies the end of data; no more data blocks follow. The format of the termination block is similar to the data block except that the byte count is 0; therefore there are no data bytes or second checksum.

General Format

(Header)	Transfer Address	Byte Count	Checksum	CR (EOL)

BYTE	CONTENTS
1-2	<DLE> ASCII slash (/) 2FH.
3-4	TRANSFER ADDRESS. (The address at which execution will begin.)
5	BYTE COUNT = 0
6	CHECKSUM: (Checksum of the transfer address and byte count.)
7-8	<DLE> <CR>

EXAMPLE: TERMINATION BLOCK

HEADER	ADDR	COUNT	CK1	EOL
102F	1023	00	06	100D

The checksum is calculated by adding with carry the nibbles in the ADDRESS and COUNT fields.

CK1 = 06 = 1+0+2+3

Because of the DLE rule, this would be transmitted as

10 2F 10 10 23 00 06 10 0D

Note that the DLE=10H, which is inserted for transmission, is not included in the checksum calculation.

Table C-6. Binary Format Abort Block

PURPOSE: Indicates an abnormal termination of the data blocks.

General Format

(Header)	0000	COUNT = FF	Checksum = 1EH	EOL

BYTE	CONTENTS
1,2	<DLE>/
3,4	0000
5	BYTE COUNT = FF
6	FIRST CHECKSUM = 1EH
7,8	<DLE> <CR>

CHECKSUM ALGORITHM:

1. CKSUM = 1EH
2. As each byte is received, add each nibble (without carry) to CKSUM.
3. Note that the <DLE> character in the heading and before the ending <CR> is not included in the checksums.

INTEL HEXADECIMAL FORMAT

The Intel Hexadecimal Data Block Format is a set of message blocks used for encoding data in the ASCII form. This data can be downloaded to the 9508S emulator using the RHEX command or Uploaded to a host using the WHEX command.

The message blocks fall into two classes: data blocks and termination blocks. RHEX is capable of accepting two forms of the termination block, but WHEX only produces one form.

Table C-7. Intel Hexadecimal Format Data Block

PURPOSE: Encodes data bytes as a series of hexadecimal digits.

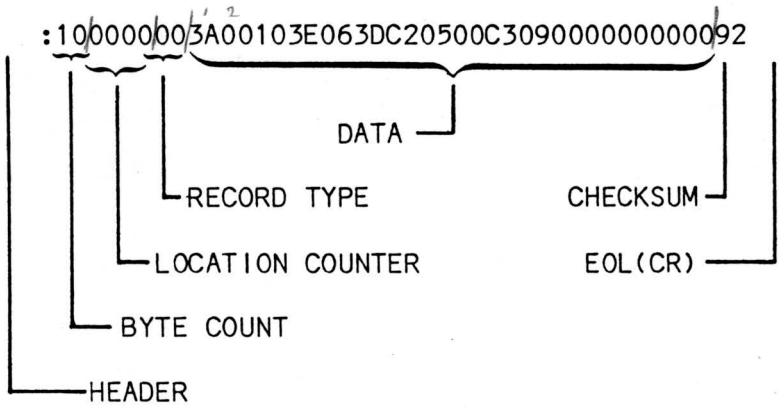
General Format

(HEADER)	BYTE COUNT	LOCATION COUNTER	RECORD TYPE	DATA	CHECKSUM	EOL (CR)

Format Description

NAME	NO. OF ASCII CHARACTERS	CONTENT DESCRIPTION
HEADER	1	ALWAYS A COLON (:)
BYTE COUNT	2	TWO-DIGIT HEXADECIMAL VALUE SPECIFYING THE NUMBER OF BYTES IN THE DATA FIELD OF THE BLOCK. MAXIMUM VALUE=10H. FOR A DATA BLOCK, THE MINIMUM COUNT IS 1.
LOCATION COUNTER	4	FOUR HEXADECIMAL DIGITS REPRESENTING THE STARTING MEMORY LOCATION OF THE BLOCK.
RECORD TYPE	2	TWO HEXADECIMAL DIGITS VALUE = 00 FOR A DATA BLOCK
DATA	2 * N	N DATA BYTES, EACH REPRESENTED AS TWO HEXADECIMAL DIGITS. EACH HEX DIGIT IS ENCODED AS AN ASCII CHARACTER 0-9 or A-F.
CHECKSUM	2	TWO-DIGIT HEXADECIMAL NUMBER REPRESENTING THE TWO'S COMPLEMENT OF THE SUM OF THE BYTE COUNT, LOCATION COUNTER, RECORD TYPE, AND DATA FIELDS (ADD WITHOUT CARRY).
EOL	1	ALWAYS A CARRIAGE RETURN

EXAMPLE 3:



Explanation of Example 3:

The BYTE COUNT=10H indicates that the DATA field contains 10H bytes represented as 20H hexadecimal characters. In decimal, this is a total of 16 data bytes represented as 32 digits.

The LOCATION COUNTER=0000 indicates that the data are to be loaded starting at address 0000.

The RECORD TYPE=00 indicates that the block is a data block.

DATA bytes - each hexadecimal digit is an ASCII character; each pair of digits represents a byte in memory. For example, the hexadecimal digits 3A represent the data byte 00111010 (binary) that is to be placed in memory location 0000.

The CHECKSUM is the two's complement of the sum, without carry, of the BYTE COUNT, LOCATION COUNTER, RECORD TYPE, and DATA Fields:

SUM= 10+00+00+00+3A+00+10+3E+06+3D+C2+05+00+C3+09+00+00+00+00+00=6E

ONE'S COMPLEMENT OF SUM=91
TWO'S COMPLEMENT OF SUM=92

Table C-8. Intel Hexadecimal Format Termination Block

PURPOSE: Specifies the end of data and PCNEXT

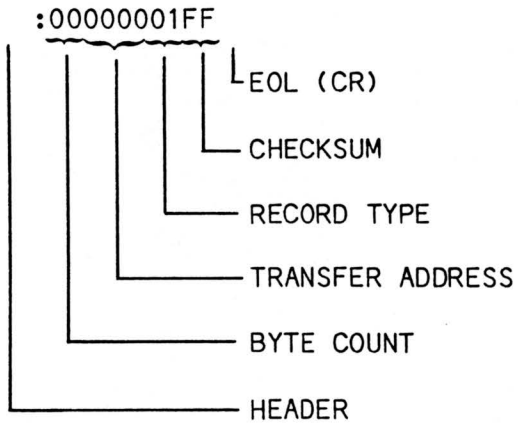
General Format

(HEADER)	BYTE COUNT	TRANSFER ADDRESS	RECORD TYPE	CHECKSUM

Format Description

NAME	NO. OF ASCII CHARACTERS	CONTENT DESCRIPTION
HEADER	1	ALWAYS A COLON (:)
BYTE COUNT	2	TWO-DIGIT HEXADECIMAL NUMBER. ALWAYS 00 FOR A TERMINATION BLOCK.
TRANSFER ADDRESS	4	FOUR HEXADECIMAL DIGITS REPRESENTING THE STARTING EXECUTION ADDRESS OF THE CODE REPRESENTED IN THE DATA BLOCKS. RHEX PLACES THIS VALUE IN PCNEXT. WHEX GETS THIS VALUE FROM THE COMMAND LINE.
RECORD TYPE	2	VALUE = 01. NOT ALL CROSS ASSEMBLERS OR LINKERS CREATE THIS FIELD. RHEX WILL WORK WITHOUT IT. WHEX ALWAYS PRODUCES IT.
CHECKSUM	2	TWO-DIGIT HEXADECIMAL NUMBER REPRESENTING THE TWO'S COMPLEMENT OF THE SUM OF THE BYTE COUNT, TRANSFER ADDRESS, AND RECORD TYPE FIELDS. RHEX WILL WORK WITHOUT THIS FIELD.
EOL	1	ALWAYS A CARRIAGE RETURN

EXAMPLE 4:



Explanation of Example 4:

The BYTE COUNT=00 indicates that there is no DATA FIELD and that this is a Termination Block.

The TRANSFER ADDRESS is 0000. If the program were being downloaded, then RHEX would place this value in the PC register.

The RECORD TYPE=01 indicates that this is a Termination Block.

The CHECKSUM is the two's complement of the sum, without carry, of the BYTE COUNT, TRANSFER ADDRESS, and RECORD TYPE fields.

SUM=00+00+00+00+00+01
ONE'S COMPLEMENT=FE
TWO'S COMPLEMENT=FF

Table C-9. Motorola Hexadecimal Format Sign-on Block

PURPOSE: This block is a comment block. It is ignored by RHEX and is not produced by WHEX.

General Format

(HEADER)	BLOCK TYPE	BYTE COUNT	COMMENT	CHECKSUM	EOL (CR)

Format Description

NAME	NO. OF ASCII CHARACTERS	CONTENT DESCRIPTION
HEADER	1	ALWAYS THE CHARACTER "S"
BLOCK TYPE	1	THE CHARACTER "0"
BYTE COUNT	2	IGNORED FOR THIS BLOCK TYPE SEE THE DATA BLOCK DESCRIPTION.
COMMENT	BYTE COUNT-3	SIGN-ON MESSAGE
CHECKSUM	2	IGNORED FOR THIS BLOCK TYPE. SEE THE DATA BLOCK DESCRIPTION.
EOL	1	ALWAYS A CARRIAGE RETURN

Table C-10. Motorola Hexadecimal Format Data Block

PURPOSE: Encodes data bytes as a series of hexadecimal digits

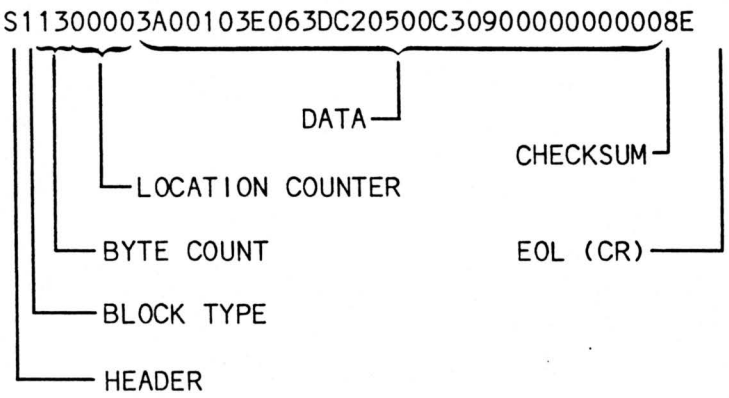
General Format

(HEADER)	BLOCK TYPE	BYTE COUNT	LOCATION COUNTER	DATA	CHECKSUM	EOL (CR)

Format Description

NAME	NO. OF ASCII CHARACTERS	CONTENT DESCRIPTION
HEADER	1	ALWAYS THE CHARACTER "S"
BLOCK TYPE	1	THE CHARACTER "1"
BYTE COUNT	2	TWO-DIGIT HEXADECIMAL NUMBER SPECIFYING THE NUMBER OF BYTES IN THE LOCATION COUNTER, DATA, AND CHECKSUM FIELDS. THE NUMBER OF CHARACTERS IS TWICE THE BYTE COUNT. THE MINIMUM VALUE IS 4. THE MAXIMUM PRODUCED BY WHEX IS 13H.
LOCATION COUNTER	4	FOUR HEXADECIMAL DIGITS REPRESENTING THE STARTING MEMORY LOCATION OF THE BLOCK.
DATA	2 * (BYTE COUNT -3)	EACH BYTE OF DATA IS REPRESENTED AS TWO HEXA-DECIMAL DIGITS. EACH HEX DIGIT IS ENCODED AS AN ASCII CHARACTER 0-9 OR A-F
CHECKSUM	2	TWO-DIGIT HEXADECIMAL NUMBER REPRESENTING THE ONE'S COMPLEMENT OF THE SUM OF THE BYTE COUNT, LOCATION COUNTER, AND DATA FIELDS. ADD WITHOUT CARRY.
EOL	1	ALWAYS A CARRIAGE RETURN

EXAMPLE 5:



Explanation of Example 5:

The BLOCK TYPE of a data block is 1.

The BYTE COUNT is 13H. There are 2 bytes in the LOCATION COUNTER FIELD, 10H bytes in the DATA field and one byte in the CHECKSUM field.

The LOCATION COUNTER is 0000 and indicates that the data is to be loaded starting at address 0.

In the DATA field, each byte is represented by two hexadecimal digits. The digits are encoded as ASCII characters.

The CHECKSUM is 8E and is the one's complement of the sum of the bytes in the BYTE COUNT, LOCATION COUNTER, and DATA fields.

SUM=13+00+00+3A+00+10+3E+06+3D+C2+05+00+C3+09+00+00+00+00+00=71
ONE'S COMPLEMENT = 8E

Table C-11. Motorola Hexadecimal Format Termination Block

PURPOSE: Specifies the end of data and PCNEXT

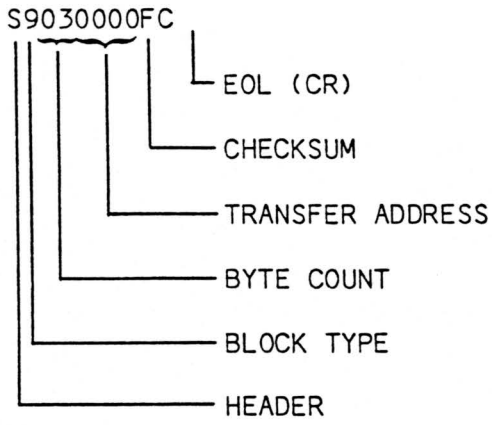
General Format

(HEADER)	BLOCK TYPE	BYTE COUNT	TRANSFER ADDRESS	CHECKSUM	EOL (CR)

Format Description

NAME	NO. OF ASCII CHARACTERS	CONTENT DESCRIPTION
HEADER	1	ALWAYS THE CHARACTER "S"
BLOCK TYPE	1	THE CHARACTER "9"
BYTE COUNT	2	TWO-DIGIT HEXADECIMAL NUMBER. ALWAYS 03.
TRANSFER ADDRESS	4	FOUR HEXADECIMAL DIGITS REPRESENTING THE STARTING EXECUTION ADDRESS OF THE CODE REPRESENTED IN THE DATA BLOCKS. RHEX PLACES THIS VALUE IN PCNEXT WHEX GETS THIS VALUE FROM THE COMMAND LINE.
CHECKSUM	2	TWO-DIGIT HEXADECIMAL NUMBER REPRESENTING THE ONE'S COMPLEMENT OF THE SUM OF THE BYTE COUNT, AND TRANSFER ADDRESS FIELDS.
EOL	1	ALWAYS A CARRIAGE RETURN

EXAMPLE 6:



Explanation of Example 6:

The BLOCK TYPE of a Termination Block is 9

The BYTE COUNT is always 3 for a Termination Block. It is the byte count of the TRANSFER ADDRESS and CHECKSUM fields.

The TRANSFER ADDRESS is 0000. RHEX places this value in the PC register when it receives the Termination Block. WHEX gets this value from the command line.

The CHECKSUM is the one's complement of the sum (add without carry) of the bytes in the BYTE COUNT and TRANSFER ADDRESS fields.

SUM=03+00+00=03
ONE'S COMPLEMENT=FC

Table C-12. Binary Format Abort Block

PURPOSE: Indicates an abnormal termination of the data blocks.

General Format

(Header)	0000	COUNT = FF	Checksum = 1EH	EOL

BYTE	CONTENTS
1,2	<DLE>/
3,4	0000
5	BYTE COUNT = FF
6	FIRST CHECKSUM = 1EH
7,8	<DLE> <CR>

CHECKSUM ALGORITHM:

1. CKSUM = 1EH
2. As each byte is received, add each nibble (without carry) to CKSUM.
3. Note that the <DLE> character in the heading and before the ending <CR> is not included in the checksums.

DOWNLOAD/UPLOAD UTILITY PROGRAM REQUIREMENTS

The Download and Upload utility programs are installed on the host system to transfer files from the host system to the emulator, or to receive and store files transmitted from the emulator to the host. The following paragraphs describe the Download and Upload program requirements, depending on the type of physical connection between the host and emulator (console or I/O), and the format of the data to be transferred (TekHex, Intel Hex, Motorola Hex or Binary). Each case is organized generally as it would occur in a chronological sequence.

Download - TekHex, Intel Hex, or Motorola Hex. Console Connection

1. Send a Control \ character to emulator (this forces the emulator to exit the interactive communications mode).

2. Synchronization (Optional). Perform program synchronization by waiting for:

   S <CR>

   and responding with:

   T <CR>

3. Read next record from the file and send it to the emulator.

4. ACK/NAK Protocol (Optional). Following transmission of each data block wait for any one of the following:

   ACK <CR> (ACK=ASCII 0 CR=300D)
   NAK <CR> (NAK=ASCII 7 CR=370D)
   Host Abort Character <CR>

   If NAK, then retransmit the same block and wait again; if ACK, transmit next block; if host abort character, close file and terminate the program. If the terminating block has been transmitted, close files and terminate the program following the receipt of ACK.

5. If an error occurs at any time while reading the file, transmit the abort block (this is supported only for Tekhex or Binary formats.)

Download - TekHex. I/O Connection.

1. Synchronization (Optional). Perform program synchronization by waiting for:

   S <CR>

   and responding with:

   T <CR>

2. Read next record from the file and send it to the emulator.

3. ACK/NAK Protocol (Optional). Following transmission of each data block wait for any one of the following:

   ACK <CR>
   NAK <CR>
   Host Abort Character <CR>

   If NAK, then retransmit the same block and wait again; if ACK, transmit next block; if host abort character, close file and terminate the program. If the terminating block has been transmitted, close files and terminate the program following the receipt of ACK.

Upload - TekHex, Intel Hex, Motorola Hex. Console Connection

1. Send a Control \ character to emulator (this forces the emulator to exit the interactive communications mode.

2. Synchronization (Optional). Perform program synchronization by waiting for:

   S <CR>

   and responding with:

   T <CR>

3. Read the next block. If there are no errors, write the block to the disk.

4. ACK/NAK Protocol (Optional). Following the receipt of a block, send out any one of the following:

   ACK <CR> - if block has no errors

   NAK <CR> - if there is a checksum error in the block

   Host abort character - if a disk write error has occurred

5. When a termination block is received perform step 4, then close the files and terminate the program.

Upload - TekHex, Intel Hex, Motorola Hex. I/O Connection

The procedure is similiar to Upload - TekHex, Intel Hex, Motorola Hex, Console Connection, except the Control \ is not sent to the emulator (step 1 of the procedure).

Download - Binary. Console Connection

All steps are the same as during Download - TekHex, Intel Hex, Motorola Hex, Console Connection, however, the details of how the characters are transferred differ. For example, given the data record:

Header	addr	Count	CK1	Data	CK2	CR
2F	0110	03	05	871069	2E	0D

it will be transmitted as:

<DLE>2F01<DLE>10030587<DLE>10692E<DLE><CR>

Note that:

- The header 2F is transmitted as <DLE>2F
- The terminating carriage return is transmitted as <DLE>0D
- If the DLE character (10H) occurs anywhere in the body of the record, it is transmitted as

  <DLE>10

  (i.e., it is transmitted twice)

Download - Binary. I/O Connection

The procedure is similar to the Download - TekHex, Intel Hex, Motorola Hex, I/O Connection, except for the details on how the <DLE> character is transferred (see Download - Binary, Console Connection for an example).

Upload - Binary, Console Connection

All steps are the same as for Upload - TekHex, Intel Hex, Motorola Hex, Console Connection, however, for each DLE character that occurs in the body of the block, it is transmitted as

<DLE>10

(Refer to Download - Binary, Console Connection, for an example)

Upload - Binary, I/O Connection

The procedure is similar to the Upload - TekHex, Intel Hex, Motorola Hex, Console Connection, except the Control \ is not sent to the emulator (step 1 of the procedure) and the details of how the <DLE> character is transmitted differ (see Download - Binary, Console Connection, for an example).

Sample Upload and Download programs

These programs support all Hexadecimal formats. They do not perform any checksum error checking. They always send an ACK for each block and are very useful in preventing any data overrun and loss.

```
C
C UPLOAD FROM 9508 TO PDP-11
C

      DIMENSION INPUT(80)
      DATA IREPLY /'0'/
C
      OPEN (UNIT=3,NAME='DOWN.DAT',TYPE='NEW')
C
100   READ (5,1010,END=900)INPUT
      WRITE (3,1020)INPUT
      TYPE 1000,IREPLY
      GO TO 100
C
C
900   CLOSE (UNIT=3)
      CLOSE (UNIT=5)
      STOP
C
C
1000  FORMAT (1X,A1)
1010  FORMAT (80A1)
1020  FORMAT (1H ,80A1)
      END


C
C DOWNLOAD FROM PDP-11 TO 9508
C
      DIMENSION INPUT(80), IREPLY(80)
C
      OPEN (UNIT=3,NAME='DOWN.DAT',TYPE='OLD')
C
100   READ (3,1000,END=900)INFILE
      TYPE 1010,INFILE
      ACCEPT 1000,REPLY
      GO TO 100
C
C
900   CLOSE (UNIT=3)
      STOP
C
C
1000  FORMAT (80A1)
1010  FORMAT (1X,80A1)
      END
```

DEFAULT AREAS

Linkdef and Hostdef Default areas.

LOCATION	DEFAULT VALUES	CONTENTS
780	00	DATA FORMAT 0 = MSI Binary 1 = TEK HEX 2 = INTEL HEX 3 = MOTOROLA HEX
781	02	ACK STRING BYTE COUNT Minimum = 00 Maximum = 03
782-784	300D00	ACK STRING
785-786	0064	TIMEOUT (0-FFFF)
787-788	0006	RETRY COUNT (0-FFFF)
789	00	ACTION ON ERROR 00 = ABORT 01 = CONTINUE/IGNORE
78A	02	NAK STRING BYTE COUNT Minimum = 00 Maximum = 03
78B-78D	370D00	NAK STRING
78E	02	SYNC STRING BYTE COUNT Minimum = 00 Maximum = 03
78F-791	530D00	SYNC STRING
792	01	BUSY PROTOCOL 00 = NONE 01 = DSR 02 = XON/XOFF 03 = ENQ/ACK
793	11	XON CHARACTER DEFAULT=11H
794	13	XOFF CHARACTER DEFAULT=13H

DEFAULT AREAS

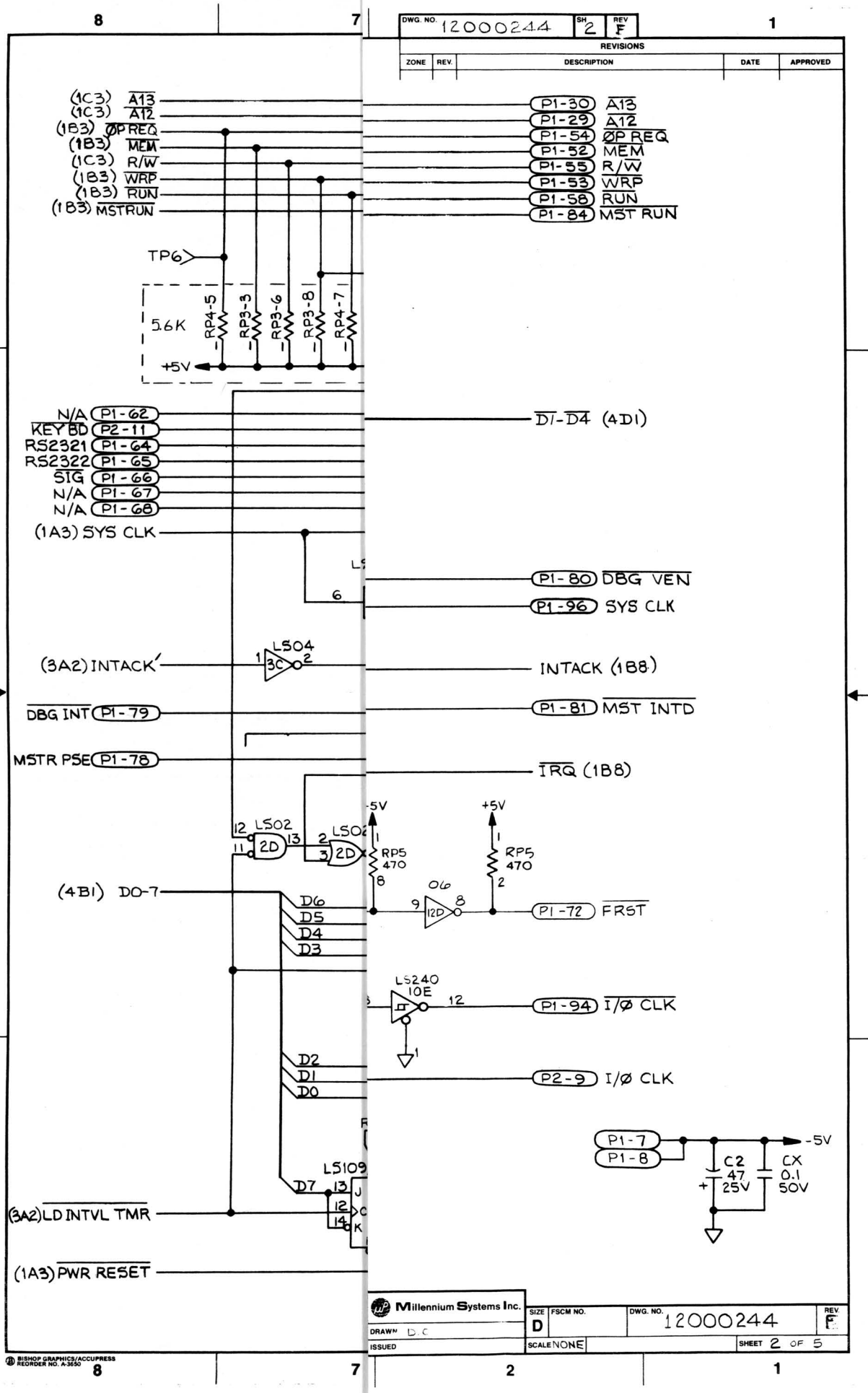
LOCATION	DEFAULT VALUES	CONTENTS
795	05	ENQ CHARACTER DEFAULT=05
796	06	EAK CHARACTER DEFAULT=06
797	00	DUPLEX 00=FULL 01=HALF
798	02	EOL STRING BYTE COUNT Minimum=00 Maximum=03
799-79B	100D00	EOL STRING*
79C-79D	0000	TURNAROUND TIME (0-FFFF) AT EOL. UNITS ARE 100ms
79E-79F	0000	DELAY TIME BETWEEN CHARACTERS (0-FFFF) UNITS ARE 100 MICROSECONDS
7AQ	00	EOF STRING BYTE COUNT Minimum=00 Maximum=03
7A1-7A3	000000	EOF STRING
7A4	00	ECHO PARAMETER 00=HOST WILL NOT ECHO DATA 01=HOST WILL ECHO DATA
7A5	00	PROMPT STRING BYTE COUNT Minimum=00 Maximum=03
7A6-7A8	000000	PROMPT STRING
7A9	01	HOST-ABORT STRING BYTE COUNT Minimum=00 Maximum=03
7AA-7AC	030000	HOST-ABORT STRING

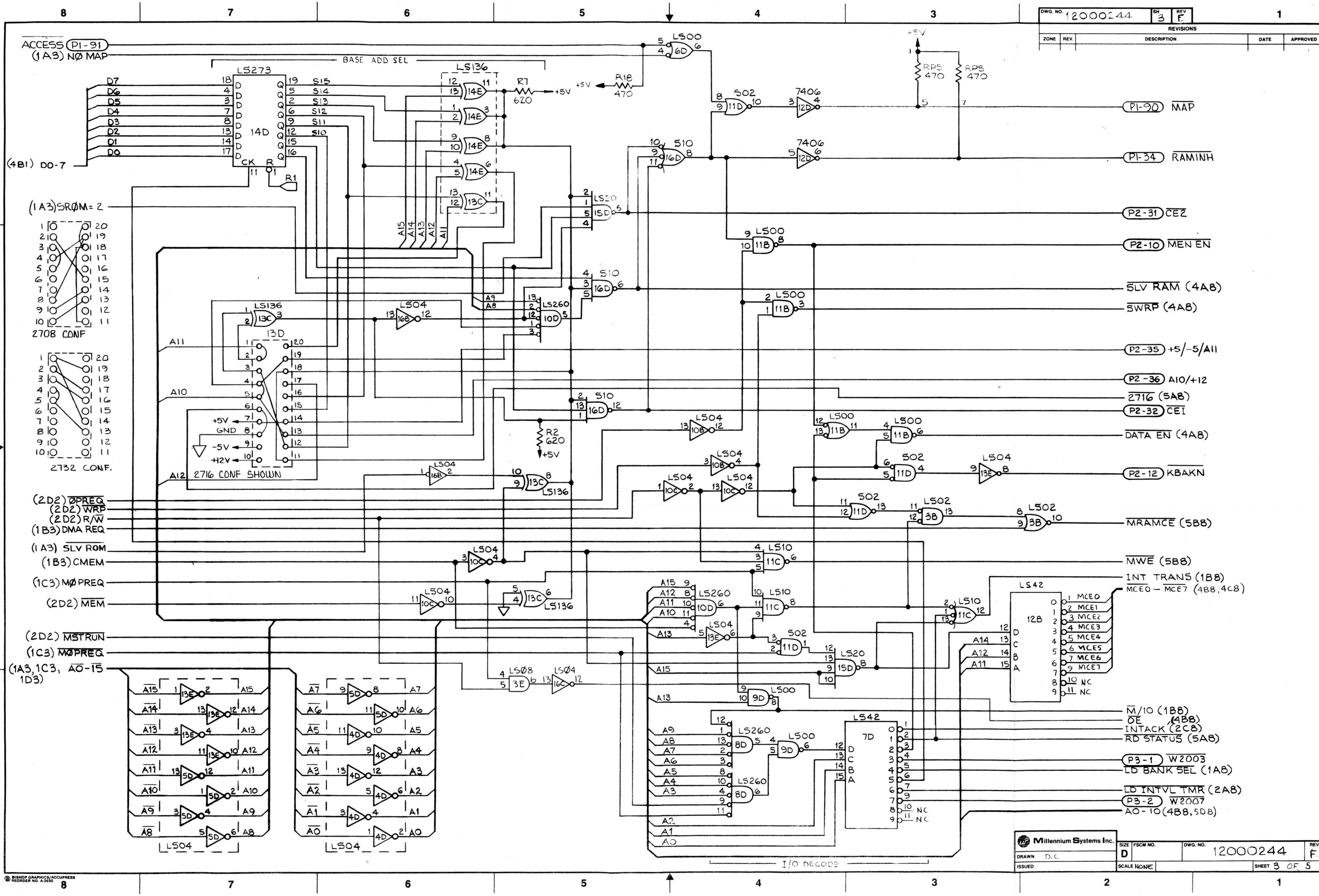
NOTE: *The EOL string 100D is for binary format only. If the host is other than a 9520, or hexadecimal format is used, the EOL string must be changed to 0D.

SCHEMATIC DIAGRAMS

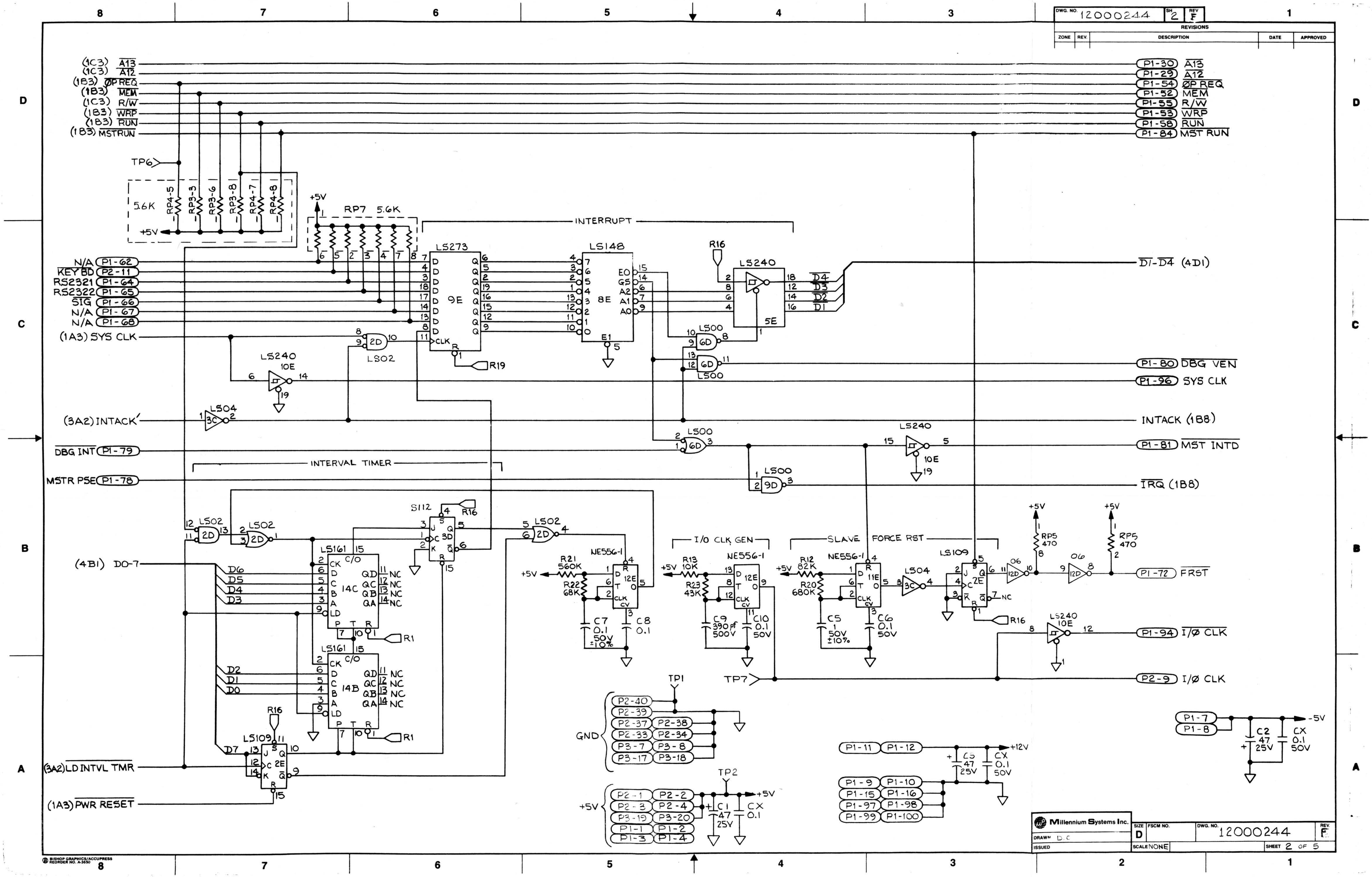
This appendix contains schematic diagrams for the 9508 MicroSystem Emulator. The schematic diagrams are subject to change and therefore are to be used only for reference purposes. The following is a list of the drawings.

NAME	DWG. NO.
Control Processor, Schematic Diagram	12000244
Debug, Schematic Diagram	12000044
Comm/Ram, Schematic Diagram	12000015
RealTime Trace, Schematic Diagram	12000170
Motherboard, Schematic Diagram	12000001
Restart/Emul S/W, Schematic Diagram	12000236
AC Distribution, Schematic Diagram	12000033
RTT Probe Circuit Card, Schematic Diagram	12000190

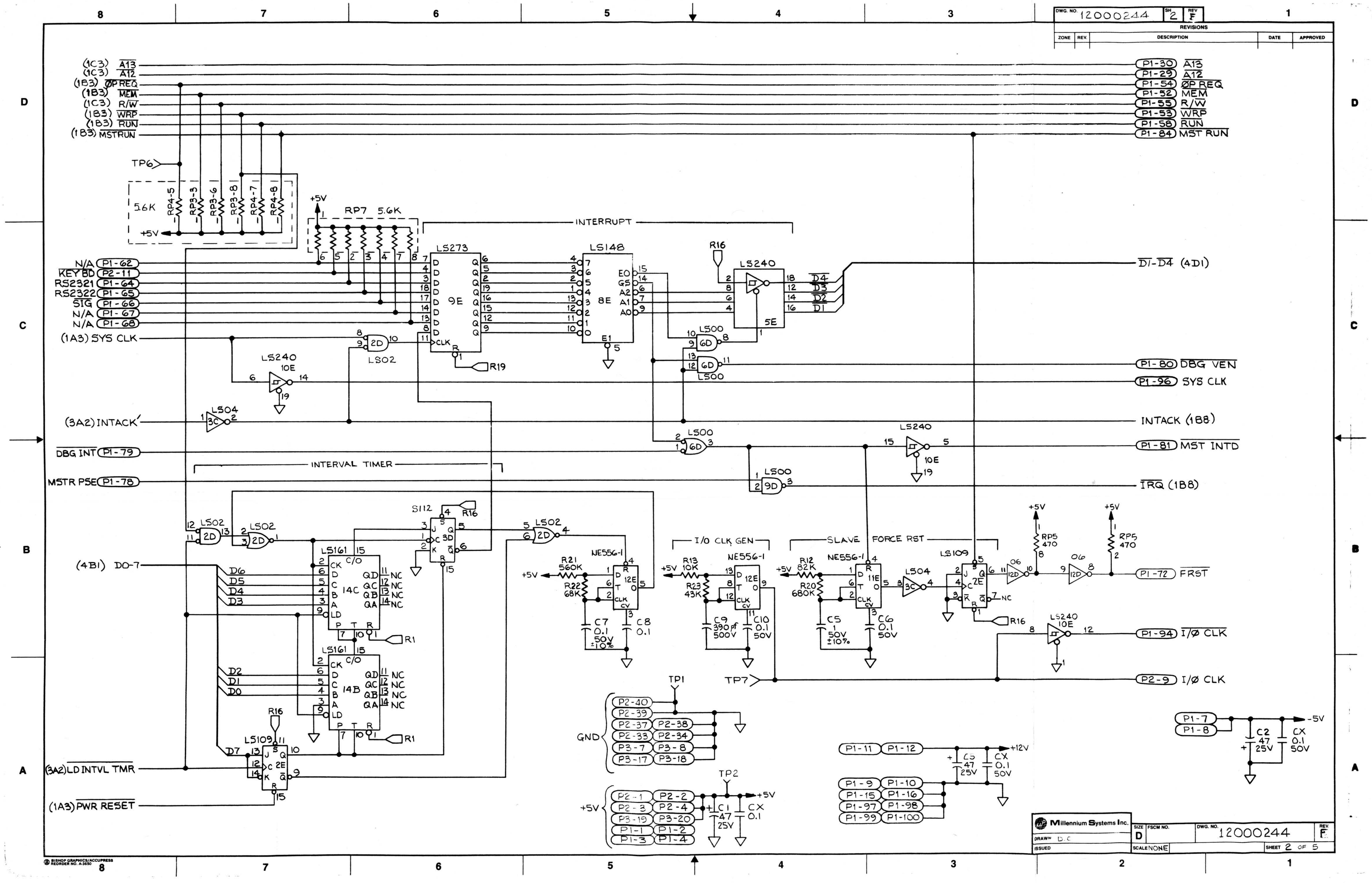

DWG. NO. 12000244 SH 2 REV F
REVISIONS
ZONE REV. DESCRIPTION DATE APPROVED
(1C3) A13
(1C3) A12
(1B3) ØP REQ
(1B3) MEM
(1C3) R/W
(1B3) WRP
(1B3) RUN
(1B3) MSTRUN
P1-30 A13
P1-29 A12
P1-54 ØP REQ
P1-52 MEM
P1-55 R/W
P1-53 WRP
P1-58 RUN
P1-84 MST RUN
TP6
5.6K
RP4-5
RP3-3
RP3-6
RP3-8
RP4-7
+5V
N/A P1-62
KEYBD P2-11
RS2321 P1-64
RS2322 P1-65
SIG P1-66
N/A P1-67
N/A P1-68
D1-D4 (4D1)
(1A3) SYS CLK
P1-80 DBG VEN
P1-96 SYS CLK
LS04
(3A2) INTACK
INTACK (1B8)
DBG INT P1-79
P1-81 MST INTD
MSTR PSE P1-78
IRQ (1B8)
LS02
-5V
+5V
RP5 470
LS240 10E
(4B1) D0-7
D6
D5
D4
D3
D2
D1
D0
P1-72 FRST
P1-94 I/Ø CLK
P2-9 I/Ø CLK
LS109
D7
(3A2) LD INTVL TMR
(1A3) PWR RESET
P1-7
P1-8
C2 47 25V
CX 0.1 50V
Millennium Systems Inc.
DRAWN D.C
ISSUED
SIZE D FSCM NO.
DWG. NO. 12000244
REV F
SCALE NONE
SHEET 2 OF 5
BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-3650



DWG. NO. 12000244 SH 3 REV F
REVISIONS
ZONE REV. DESCRIPTION DATE APPROVED
ACCESS P1-91
(1A3) NØ MAP
BASE ADD SEL
LS273
14D
LS136
R7 620
R18 470
+5V
(4B1) D0-7
D7 D6 D5 D4 D3 D2 D1 D0
S15 S14 S13 S12 S11 S10
(1A3) SRØM = 2
2708 CONF
2732 CONF.
2716 CONF SHOWN
LS00
S02
7406
S10
LS20
LS260
LS04
RP5 470
P1-90 MAP
P1-34 RAMINH
P2-31 CE2
P2-10 MEN EN
SLV RAM (4A8)
SWRP (4A8)
P2-35 +5/-5/A11
P2-36 A10/+12
2716 (5A8)
P2-32 CE1
DATA EN (4A8)
P2-12 KBAKN
MRAMCE (5B8)
MWE (5B8)
INT TRANS (1B8)
MCE0 – MCE7 (4B8, 4C8)
LS42
12B
M/IO (1B8)
OE (4B8)
INTACK (2C8)
RD STATUS (5A8)
P3-1 W2003
LD BANK SEL (1A8)
LD INTVL TMR (2A8)
P3-2 W2007
A0-10 (4B8, 5D8)
(2D2) ØPREQ
(2D2) WRP
(2D2) R/W
(1B3) DMA REQ
(1A3) SLV ROM
(1B3) CMEM
(1C3) MØ PREQ
(2D2) MEM
(2D2) MSTRUN
(1C3) MØPREQ
(1A3, 1C3, 1D3) A0-15
R2 620
I/O DECODE
Millennium Systems Inc.
SIZE D
DWG. NO. 12000244
REV. F
DRAWN D.C.
SCALE NONE
SHEET 3 OF 5
BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-3650



DWG. NO. 12000244 SH 2 REV F
REVISIONS
ZONE REV. DESCRIPTION DATE APPROVED
(1C3) A13
(1C3) A12
(1B3) ØP REQ
(1B3) MEM
(1C3) R/W
(1B3) WRP
(1B3) RUN
(1B3) MSTRUN
P1-30 A13
P1-29 A12
P1-54 ØP REQ
P1-52 MEM
P1-55 R/W
P1-53 WRP
P1-58 RUN
P1-84 MST RUN
TP6
5.6K
RP4-5 RP3-3 RP3-6 RP3-8 RP4-7 RP4-8
+5V
RP7 5.6K
INTERRUPT
LS273
9E
LS148
8E
R16
LS240
5E
D1-D4 (4D1)
D4 D3 D2 D1
N/A P1-62
KEYBD P2-11
RS2321 P1-64
RS2322 P1-65
SIG P1-66
N/A P1-67
N/A P1-68
(1A3) SYS CLK
LS02
2D
R19
LS500
6D
P1-80 DBG VEN
P1-96 SYS CLK
LS240
10E
(3A2) INTACK'
LS04
3C
INTACK (1B8)
DBG INT P1-79
P1-81 MST INTD
INTERVAL TIMER
MSTR PSE P1-78
LS500
9D
IRQ (1B8)
S112
3D
R16
LS161
14C
14B
NC
R1
(4B1) D0-7
D6 D5 D4 D3 D2 D1 D0 D7
NE556-1
12E
R21 560K
R22 68K
C7 0.1 50V ±10%
C8 0.1
I/O CLK GEN
R13 10K
R23 43K
C9 390pf 500V
C10 0.1 50V
SLAVE FORCE RST
R12 82K
R20 680K
11E
C5 1 50V ±10%
C6 0.1 50V
LS109
2E
RP5 470
12D
P1-72 FRST
R16
LS240 10E
P1-94 I/Ø CLK
TP7
P2-9 I/Ø CLK
TP1
GND
P2-40 P2-39 P2-37 P2-38 P2-33 P2-34 P3-7 P3-8 P3-17 P3-18
TP2
+5V
P2-1 P2-2 P2-3 P2-4 P3-19 P3-20 P1-1 P1-2 P1-3 P1-4
C1 47 25V
CX 0.1
P1-11 P1-12 +12V
C3 47 25V
CX 0.1 50V
P1-9 P1-10 P1-15 P1-16 P1-97 P1-98 P1-99 P1-100
P1-7 P1-8 -5V
C2 47 25V
CX 0.1 50V
(1A3) PWR RESET
(3A2) LD INTVL TMR
LS109
2E
Millennium Systems Inc.
SIZE D
FSCM NO.
DWG. NO. 12000244
REV. F
DRAWN D.C
ISSUED
SCALE NONE
SHEET 2 OF 5
BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-3650



DWG. NO. 12000244 SH 2 REV F
REVISIONS
ZONE REV. DESCRIPTION DATE APPROVED
(1C3) A13
(1C3) A12
(1B3) ØP REQ
(1B3) MEM
(1C3) R/W
(1B3) WRP
(1B3) RUN
(1B3) MSTRUN
P1-30 A13
P1-29 A12
P1-54 ØP REQ
P1-52 MEM
P1-55 R/W
P1-53 WRP
P1-58 RUN
P1-84 MST RUN
TP6
5.6K
RP4-5 RP3-3 RP3-6 RP3-8 RP4-7 RP4-8
+5V
RP7 5.6K
INTERRUPT
LS273
9E
LS148
8E
R16
LS240
5E
D1-D4 (4D1)
N/A P1-62
KEYBD P2-11
RS2321 P1-64
RS2322 P1-65
SIG P1-66
N/A P1-67
N/A P1-68
(1A3) SYS CLK
LS02
2D
R19
LS500
6D
P1-80 DBG VEN
P1-96 SYS CLK
LS240
10E
(3A2) INTACK'
LS04
3C
INTACK (1B8)
DBG INT P1-79
P1-81 MST INTD
INTERVAL TIMER
MSTR PSE P1-78
LS00
9D
IRQ (1B8)
S112
3D
LS161
14C
14B
(4B1) D0-7
NE556-1
12E
R21 560K
R22 68K
C7 0.1 50V ±10%
C8 0.1
I/O CLK GEN
R13 10K
R23 43K
C9 390pf 500V
C10 0.1 50V
SLAVE FORCE RST
R12 82K
R20 680K
11E
C5 1 50V ±10%
C6 0.1 50V
LS109
2E
RP5 470
P1-72 FRST
P1-94 I/Ø CLK
P2-9 I/Ø CLK
TP1
TP7
GND
P2-40 P2-39 P2-37 P2-38 P2-33 P2-34 P3-7 P3-8 P3-17 P3-18
TP2
+5V
P2-1 P2-2 P2-3 P2-4 P3-19 P3-20 P1-1 P1-2 P1-3 P1-4
C1 47 25V
CX 0.1
P1-11 P1-12 +12V
C3 47 25V
CX 0.1 50V
P1-9 P1-10 P1-15 P1-16 P1-97 P1-98 P1-99 P1-100
P1-7 P1-8 -5V
C2 47 25V
CX 0.1 50V
(3A2) LD INTVL TMR
(1A3) PWR RESET
Millennium Systems Inc.
SIZE D
DWG. NO. 12000244
REV F
DRAWN D.C
ISSUED
SCALE NONE
SHEET 2 OF 5
BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-3650

DWG. NO.	SH	REV
12000244	1	F

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED
	01	PROTOTYPE		
	99	PILOT RELEASE ECO 2515	4/13/81	
	A	PRODUCTION RELEASE ECO 2560	5/19/81	
	B	PER ECO 2710	9-11-81	
	C	PER ECO # 27101	9/22/81	
	D	PER ECO # 2790	10-13-81	
	E	PER ECO # 2942	1-5-82	
	F	INCREASE RAM SPACE @7A PER ECO# 3033	2-24-82	

NOTE: UNLESS OTHERWISE SPECIFIED

1. RESISTOR VALUES ARE IN OHM 5% 1/4 W CC.
2. CAPACITOR VALUES ARE IN UF.
3. ▷ = PULL UPS AND ARE 5.6K
4. IC'S ARE 7400 FAMILY.
5. RESISTOR PAKS ARE 5.6K

P3				
TP7		NE556	11E	1
Y1		LS08	3E	2
R23				
C11	C4	7406	12D	1
RP7	R4	LS240	9C	2
JUMPER J8		LS109	3A	1

13000244	9508				
NEXT ASSY	USED ON	LAST USED	NOT USED	TYPE	REF DES	QTY
FIRST APPLICATION		REF DESIGNATIONS		SPARE GATES		

APPROVALS	DATE
DRAWN D.C	2-25-81
CHECKED Ray K	4/13/81
ENGR C Zarlenga	4/13/81

Millennium Systems Inc.

SCHEMATIC CONTROL PROCESSOR

SIZE D | DWG. NO. 12000244 | REV F

SHEET 1 OF 5

DWG. NO.	SH.	REV.
12000244	4	F

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED

(3B2) ØPREQ

(5B2) DMO-7

MASTER PROM

2716 16A

2716 11A

2716 15A

2716 10A

2716 13A

2716 8A

2716 12A

2716 7A

(3B2) MCE4-7 — MCE7, MCE6, MCE5, MCE4

(3A2) AO-9

(3B2) MCEO-3 — MCE3, MCE2, MCE1, MCEO

(3A2) A10

+5V

KBD7 P2-14
KBD6 P2-13
KBD5 P2-16
KBD4 P2-15
(1C3) WRP
KBD3 P2-18
KBD2 P2-17
KBD1 P2-20
KBD0 P2-19

(3A2) OE

(3C2) DATA EN
(3C2) SLVRAM
(3C2) SWRP
(1B3) DATA IN
(1B3) DATA OUT

SLV RAM

6116-2, (120NS) RAM

2101-2 15B

2101-2 15C

+5V R8 470

DO D1 D2 D3 — DO1 DO2 DO3 DO4

D4 D5 D6 D7 — DO1 DO2 DO3 DO4

CE1 R/W OD

81LS98 6E

81LS98 7E

DM0 DM1 DM2 DM3 DM4 DM5 DM6 DM7

P1-36 DO
P1-37 D1
P1-38 D2
P1-39 D3
P1-40 D4
P1-41 D5
P1-42 D6
P1-43 D7

D1-D4 (2C2)

(1A8) DO-7 (2B8) (3D8)

KEY BOARD/SLV ROM IN

LS244 15E

81LS98 16E

KEY BOARD / SLV ROM IN

L504 16C

L504 16C 16B

R17 +5V 5.6K

RP6 5.6K

DWG. NO.	SH.	REV.
12000244	5	F

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED

(3A2) A0-9

P2-29 A0
P2-30 A1
P2-27 A2
P2-28 A3
P2-25 A4
P2-26 A5
P2-23 A6
P2-24 A7
P2-21 A8
P2-22 A9

DA0 P3-25
DA1 P3-23
DA2 P3-26
DA3 P3-24
DA4 P3-22
DA5 P3-21

LS157 6A
LS157 5A
RP1-3
A0
A1
A2
A3
A4
A5
A6
A7

LS04 10B
LS00 9B
LS04 10B
LS00 9B

2114L3 6B
2114L3 5B

11 DM0
12 DM1
13 DM2
14 DM3
11 DM4
12 DM5
13 DM6
14 DM7

(1C3) DMA REQ
(3B2) MRAMCE
(3B2) MWE

P/O 6800 5C
33 DM0
32 DM1
31 DM2
30 DM3
29 DM4
28 DM5
27 DM6
26 DM7

DM0 - DM7 (4B8)

+5V
RP2 5.6K
R3 5.6K

8 POS DIP SW 8B
NC

LS240 7B

(3C2) 2716
(3A2) RD STATUS

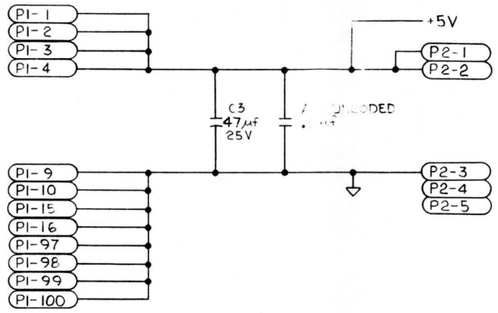
LS04

P3-15 DM7
P3-9 DM6
P3-16 DM5
P3-10 DM4
P3-11 DM3
P3-12 DM2
P3-13 DM1
P3-14 DM0

Millennium Systems Inc.	SIZE	FSCM NO.	DWG. NO.	REV.
DRAWN R.C.	D		12000244	F
ISSUED	SCALE NONE		SHEET 5 OF 5	

BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-3650

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED
	A	PRODUCTION REL ECO1191	10-9-78	
	B	REVISED PER ECO*1237	12-30-78	
	C	REVISED PER ECO 1299	1-24-79	
	D	ECO 1308	3-21-79	
	E	PER ECO 1737	11-27-79	
	F	PER ECO 1847	2-22-80	
	G	PER ECO 2596	7-31-81	



NOTES:

1. SYMBOL ▷ INDICATES A PULL UP THROUGH A DISCRETE RESISTOR OR A RESISTOR PACK. THE SAME RESISTOR MAY BE SHOWN MORE THAN ONCE.
2. RESISTORS ARE IN OHMS, 1/4 W, 5%.

NEXT ASSY	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE:
FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±

MATERIAL

FINISH

DO NOT SCALE DRAWING

CONTRACT NO.

APPROVALS	DATE
DRAWN	
CHECKED	11-9-79

Millennium Information Systems, Inc.

SCHEMATIC
DEBUG

SIZE	CODE IDENT NO.	DRAWING NO.	REV
D		12000044	G

SCALE —— SHEET 1 OF 4

DWG. NO.	SH	REV.
12000044	2	G

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED
		SEE SHT 1		

SH4 D4 CLR BINT
SH4 C4 BUF JMPACK
SH4 B4 CLR ENAB
BP2EN
BP1EN
SIN CYEN SH4 B1
EN COMP
BP2 SH4 B1
BP1 SH4 B1
SH4 C4 BUF RESET
R/W P1-55
SH4 C4 INH BRK
SLV OP REQ P1-75
SH4 C4 INT REQ
FETCH P1-82
SH4 B4 MASTER
FETCH SH4 B1
BUF SOR SH4 B1, SH3 C8
SAVE LAST SH3 B8
TP4 BP2 COMP
TP5 BP1 COMP
PCLD SH3 B8
SH3 DI AD ⩵ EQ
SH3 AI AD LEQ
WRP P1-53
SH4A4 SLAVE
BPLD SH3 D8
LD JA⩵ SH3 D8
LD JAL SH3 B8
OUT CMD SH4 B1
MAST RD SH3 B8
RD AD⩵ SH3 D8
RD ADL SH3 B8
P1-62 (SLV) INTØ
TP3
SVC REQ SH4 DI
MSK DBG INT SH4 C1
GND
MEM/IO P1-52
OPREQ P1-54
SLV RESET EN SH4 B1
SLV JMPEN SH4 A1
SH3 D8 BUF ADDRESS BUS
SH3 D8 BDØ-BD7

QTY REQD	CODE IDENT	PART OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	MATERIAL SPECIFICATION

PARTS LIST

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±

MATERIAL

FINISH

NEXT ASSY | USED ON

APPLICATION

DO NOT SCALE DRAWING

CONTRACT NO.

APPROVALS	DATE
DRAWN	
CHECKED	
ISSUED	

Millennium Systems Inc.

SCHEMATIC
DEBUG

SIZE	FSCM NO.	DWG. NO.	REV.
D		12000044	G

SCALE — SHEET 2 OF 4

DWG. NO.	SH	REV.
12000044	3	G

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED
		SEE SHT 1		

SH2 B1 LD JAL
SH2 B1 RD ADL
SH2 A8 BUF ADDRESS BUF
SH2 B1 PCTD
SH2 C1 BUF SOR
SH2 B1 MAST RD
SH2 B1 RD ADL
SH2 C1 SAVE LAST
SH2 B1 PCLD
SH2 B1 LD JAL
SH4 C4 BUF JMPACK

AD LEQ SH2 B8

BA0 3C LS04 BA1 3C LS04

LS670 WB WA GW — D3 D4 D1 D2 4x4 FILE REG 6C — RB RA GR — BD7 BD6 BD5 BD4
LS670 WB WA GW — D3 D4 D1 D2 4x4 FILE REG 5C — RB RA GR — BD3 BD2 BD1 BD0
LS670 WB WA GW — D4 D3 D1 D2 4x4 FILE REG 1C — RB RA GR — BD7 BD6 BD5 BD4
LS670 WB WA GW — D4 D3 D1 D2 4x4 FILE REG 2C — RB RA GR — BD3 BD2 BD1 BD0

S30 4B · S30 3B · S86 16 PLACES (1B, 2B)

LS670 GW WA WB — D4 D3 D2 D1 4x4 FILE REG 6A — GR RA RB — BA15 BA14 BA13 BA12 — BD7 BD6 BD5 BD4
LS670 GR RA RB — D4 D3 D2 D1 4x4 FILE REG 5A — GW WA WB — BA11 BA10 BA9 BA8 — BD3 BD2 BD1 BD0
LS670 GW WA WB — D4 D3 D1 D2 4x4 FILE REG 1D — GR RA RB — BA7 BA6 BA5 BA4 — BD7 BD6 BD5 BD4
LS670 GR RA RB — D3 D4 D1 D2 4x4 FILE REG 2D — GW WA WB — BA3 BA2 BA1 BA0 — BD3 BD2 BD1 BD0

LS374 OE — 4D 8D 7D 3D 4Q 5Q 8Q 7Q 3Q 4Q 1Q 2Q CK — BD7 BD6 BD5 BD4 BD3 BD2 BD1 BD0 — JA15 JA14 JA13 JA12 JA11 JA10 JA9 JA8
LS374 OE — 6D 5D 8D 7D 3D 4D 1D 2D 6Q 5Q 8Q 7Q 3Q 4Q 1Q 2Q CK — BD7 BD6 BD5 BD4 BD3 BD2 BD1 BD0 — JA7 JA6 JA5 JA4 JA3 JA2 JA1 JA0

LS244 4E — 1 19 — JA15 JA14 JA13 JA12 JA11 JA10 JA9 JA8 — A15 A14 A13 A12 A11 A10 A9 A8
LS244 3E — 1 19 — JA7 JA6 JA5 JA4 JA3 JA2 JA1 JA0 — A7 A6 A5 A4 A3 A2 A1 A0
LS240 5E — 1 19 — BA15 BA14 BA13 BA12 BA11 BA10 BA9 BA8 — A15 A14 A13 A12 A11 A10 A9 A8
LS240 2E — 1 19 — BA7 BA6 BA5 BA4 BA3 BA2 BA1 BA0 — A7 A6 A5 A4 A3 A2 A1 A0

P1-32 A15 · P1-31 A14 · P1-30 A13 · P1-29 A12 · P1-28 A11 · P1-27 A10 · P1-26 A9 · P1-25 A8 · P1-24 A7 · P1-23 A6 · P1-22 A5 · P1-21 A4 · P1-20 A3 · P1-19 A2 · P1-18 A1 · P1-17 A0

LS240 6D — D7 P1-43 SH4 D3 D7 — D6 P1-42 SH4 D3 D6 — D5 P1-41 SH4 D3 D5 — D4 P1-40 SH4 D3 D4 — BD7 BD6 BD5 BD4
LS240 5D — D3 P1-39 SH4 D3 D3 — D2 P1-38 SH4 D3 D2 — D1 P1-37 SH4 D3 D1 — D0 P1-36 SH4 D3 D0 — BD3 BD2 BD1 BD0

SH2 B1 MAST RD — 3C LS04

BD0-BD7 SHT 4 A1 SHT 2 A8

BA2 SH4 D1
BA1 SH4 D1
BA0 SH4 D1
AD LEQ SH2 B8

QTY REQD	CODE IDENT	PART OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	MATERIAL SPECIFICATION

PARTS LIST

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS ± · DECIMALS .XX ± .XXX ± · ANGLES ±

MATERIAL

FINISH

NEXT ASSY	USED ON
APPLICATION	

DO NOT SCALE DRAWING

CONTRACT NO.

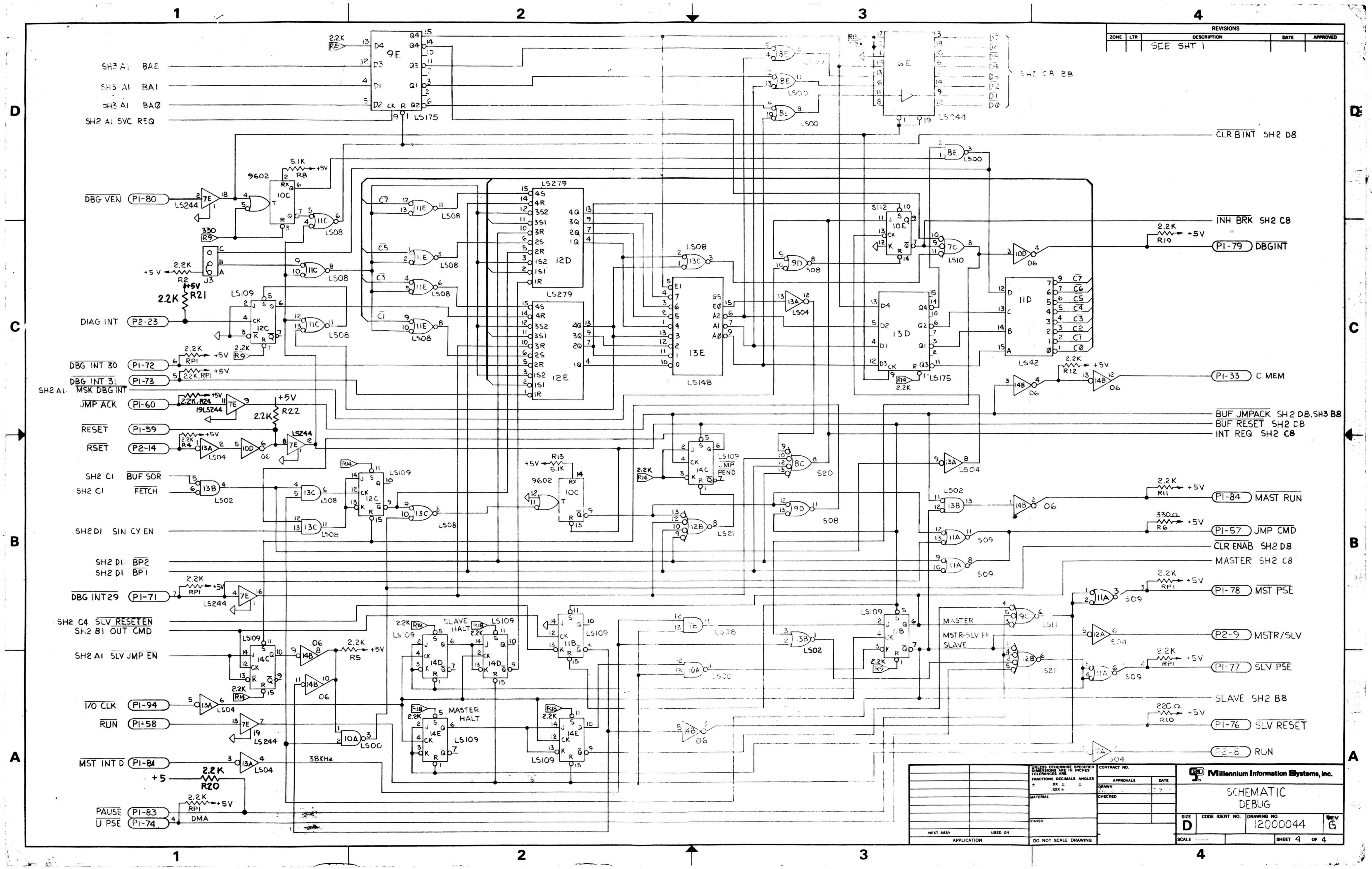
APPROVALS	DATE
DRAWN	
CHECKED	
ISSUED	

Millennium Systems Inc.

SCHEMATIC
DEBUG

SIZE	FSCM NO.	DWG. NO.	REV.
D		12000044	G

SCALE — · SHEET 3 OF 4


REVISIONS
ZONE LTR DESCRIPTION DATE APPROVED
SEE SHT 1
SH3 A1 BA2
SH3 A1 BA1
SH3 A1 BAØ
SH2 A1 SVC REQ
9E
LS175
LS244
SH2 C8 B8
CLR BINT SH2 D8
DBG VEN P1-80
9602
10C
LS279
12D
12E
LS148
13E
13D
11D
LS42
INH BRK SH2 C8
P1-79 DBGINT
DIAG INT P2-23
DBG INT 30 P1-72
DBG INT 31 P1-73
SH2 A1 MSK DBG INT
JMP ACK P1-60
RESET P1-59
RSET P2-14
P1-33 C MEM
BUF JMPACK SH2 D8, SH3 B8
BUF RESET SH2 C8
INT REQ SH2 C8
SH2 C1 BUF SOR
SH2 C1 FETCH
SH2 D1 SIN CY EN
SH2 D1 BP2
SH2 D1 BP1
DBG INT29 P1-71
SH2 C4 SLV RESETEN
SH2 B1 OUT CMD
SH2 A1 SLV JMP EN
I/O CLK P1-94
RUN P1-58
MST INT D P1-81
PAUSE P1-83
U PSE P1-74
DMA
38KHz
SLAVE HALT
MASTER HALT
JMP PEND
MASTER
MSTR-SLV FF
SLAVE
P1-84 MAST RUN
P1-57 JMP CMD
CLR ENAB SH2 D8
MASTER SH2 C8
P1-78 MST PSE
P2-9 MSTR/SLV
P1-77 SLV PSE
SLAVE SH2 B8
P1-76 SLV RESET
P2-8 RUN
Millennium Information Systems, Inc.
SCHEMATIC
DEBUG
SIZE D
DRAWING NO. 12000044
REV G
SHEET 4 OF 4
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED
		SEE HISTORY		
	R	PER ECO 2516 MJR	5/20/81	
	S	PER ECO 2539 MJR	7-15-81	
	U	PER ECO 2998 MJR	9-14-81	
	V	PER ECO # 2760 MJR	9/22/81	
	W	P/L CHANGE TO REMOVE TRI-S CONNECTOR PER ECO 2939	1-5-81	
	X	ADDED 2732 OPT & 19.2K BAUD OPT. PER ECO 3032A MJR	3/15/82	

ADDR BUS

ADDR BUS

(C3, SH-3) R2011

(C1, SH-3) R/W

R/W P1-55

(B1, SH-3) A5 - A9

A15 P1-32

A14 P1-31

A13 P1-30

A12 P1-29

A11 P1-28

A10 P1-27

A9 P1-26

A8 P1-25

A7 P1-24

A6 P1-23

A5 P1-22

A4 P1-21

A3 P1-20

A2 P1-19

A1 P1-18

A0 P1-17

(C1, SH-3) A3/A3

OPREQ P1-54

(B1, SH-3) OPREQ

CMEM P1-33

WRP P1-53

M/IO P1-52

(C1, SH-2) RAM INH

RAM INH P1-34

(B4, SH-2) D2 D1 D0 D7 D6 D5 D4 D3

(A9 SH-2) D2' D1' D0'

(C3, SH-3) W2010

(C3, SH-3) W2011

RESET P1-59

LS04

LS00

LS02

LS08

LS10

LS30

LS153 1C

LS240 10D

LS273 8D

LS86

LS133 7E

LS109 13D

7406 11E

LS42 DECODER 1E

2716/2732 ROM1 5C

2716/2732 ROMØ 8C

RAM 7A 1C 2114

RAM 7B 1D 2114

RAM 6A 2C

RAM 5A 3C

RAM 4A 4C

RAM 3A 1A

RAM 2A 2A

RAM 1A 3A

RAM ØA 4A

RAM 6B 2D

RAM 5B 3D

RAM 4B 4D

RAM 3B 1B

RAM 2B 2B

RAM 1B 3B

RAM ØB 4B

R7, 2.2K

R6, 3K

+5V

R10

R24, 10K

RZ1 560

R8 2.2K

R20 2.2K

W1 W2 W4 W5 W10

TP5

A0 - 15 (C1, SH-3)

A10 - 15' (B1, SH-3)

OPREQ (D1, SH-2)

CMEM (A1, SH-3)

R/W (D1, SH-3; B1, SH-2)

CS (C1, SH-2)

RAM EN (B1, SH-2)

P1-91 ACCESS

(SEE SHEET 3 OF 3 FOR NOTES)

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS DECIMALS ANGLES ± / XX ± / XXX ± / ±	CONTRACT NO.		Millennium Information Systems, Inc.
MATERIAL	APPROVALS	DATE	SCHEMATIC COMM/RAM
FINISH	DRAWN	12/77	
13000015 NEXT ASSY	CHECKED	5-20-78	SIZE D · CODE IDENT NO. · DRAWING NO. 12000015 · REV X
USED ON		5-26-81	SCALE — · SHEET 1 OF 3
APPLICATION	DO NOT SCALE DRAWING		

P2 # J1 # com. port (Host)
P3 # J2 # console

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED
		SEE SH-1		

BB RXDATA P2-5
CF DCD P2-15
CB CTS P2-9
CC DSR P2-11
(C3,SH-3) E2014 W2019
(C3,SH-3) E2013 W2013
(C3,SH-3) E2016 W2016
(C3.SH-3) E2017 W2017
(C4 SH-1) OPREQ
BA TXDATA P3-3
CA RTS P3-7
CD DTR P3-14
+5V P1-1,23A
+12V P1-11,12
GND P1-97,98
GND P1-99,100
-5V P1-7,8
-12V P1-13,14

D7 P1-43
D6 P1-42
D5 P1-41
D4 P1-40
D3 P1-39
D2 P1-38
D1 P1-37
D0 P1-36

(B1,SH-1) RAMINH
(C4,SH-1) CS
(C4,SH-1) RAMREN
(B3,SH-3) SEL
(C4,SH-1) R/W
SYS CLK P1-96

(C3,SH-3) W2012
(C3,SH-3) W2015

(B3,SH3) R2012
(B3,SH3) R2015

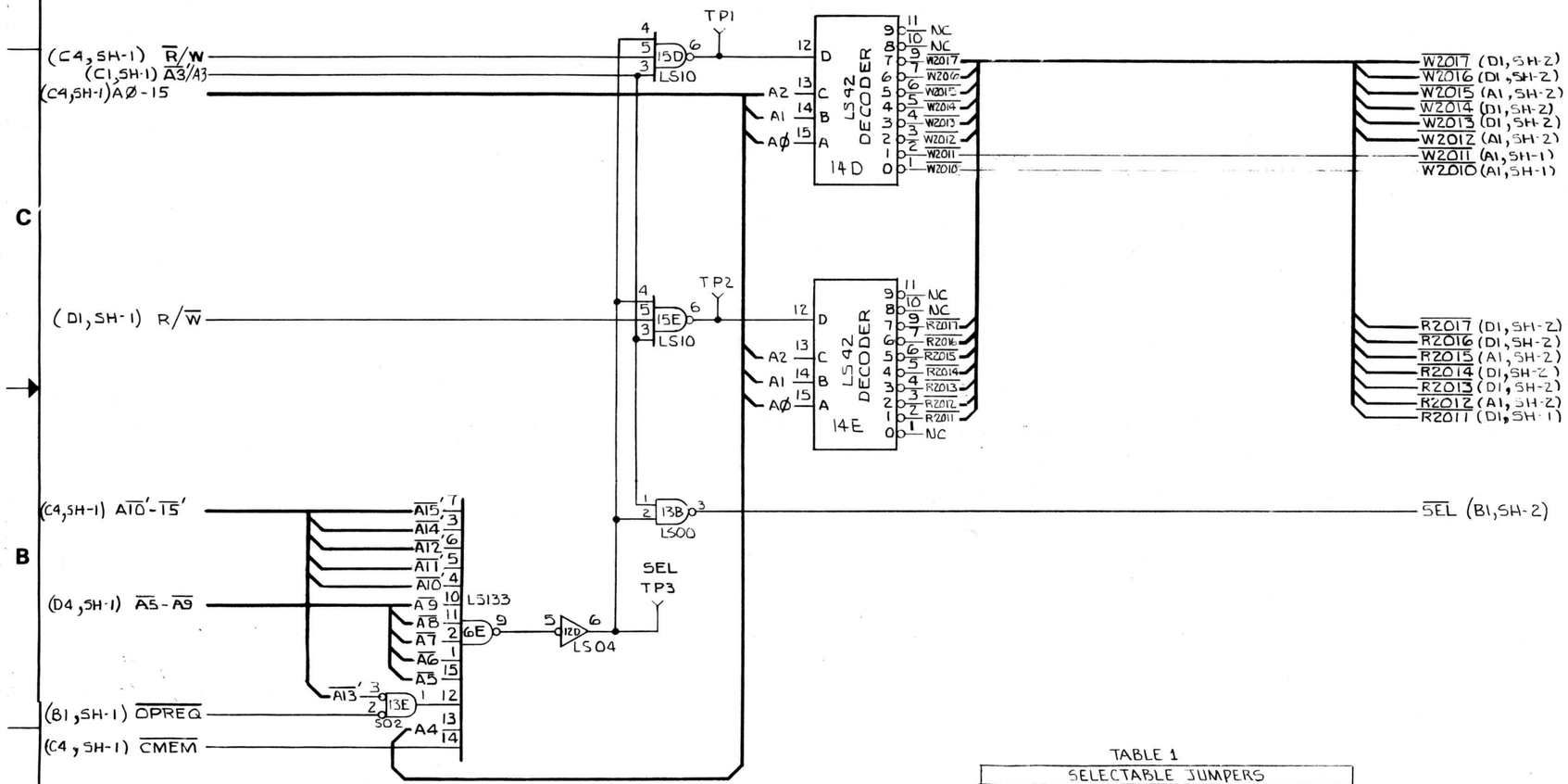
(B1,SH-1) D7 D6 D5 D4 D3 D2 D1 D0

(A1,SH-1) D2' D1' D0'

(SEE SHEET 3 OF 3 FOR NOTES)

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS DECIMALS ANGLES .XX ± .XXX ±	CONTRACT NO.	Millennium Information Systems, Inc.	
MATERIAL	APPROVALS / DRAWN TMW/11 12/7/77 / CHECKED MJS 3-30-78	SCHEMATIC COMM/RAM	
FINISH		SIZE D · CODE IDENT NO. · DRAWING NO. 1200015 X	
NEXT ASSY / USED ON / APPLICATION	DO NOT SCALE DRAWING	SCALE —	SHEET 2 OF 3

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED
		SEE SHEET 1		



LAST REF. DESIG. USED				
11A	11B	17C	17D	17E
C33	W10	R25	RP3	TP6

REF. DESIG. NOT USED	
12A - 17A	R22
6B, 9B, 12B-17B	
6C, 7, 9, 16C	

TABLE 1

SELECTABLE JUMPERS										
A SY NO.	JUMPER W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
-01	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B
-02	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B
-03	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B
-04	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B
-05	A-B	B-C	A-B	B-C	B-C	A-B	A-B	A-B	A-B	A-B
-06	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B
-07	A-B	A-B	B-C	A-B	A-B	B-C	B-C	B-C	B-C	A-B
-08	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B	A-B
-09	B-C	A-B	B-C	A-B	OPEN	B-C	B-C	B-C	B-C	B-C
-10	B-C	A-B	B-C	A-B	OPEN	B-C	B-C	B-C	B-C	B-C

4. FOR 2716 PIN 21 IS +5V, FOR 2732 PIN 21 IS A11
3. ALL RES PULL UP'S ARE 2.2K
2. ALL CAP VALUES ARE IN UF.
1. ALL RES VALUES ARE IN OHMS, 5%, 1/4W.

NOTES: (UNLESS OTHERWISE SPECIFIED:)

12000015	USA	UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ARE: FRACTIONS DECIMALS ANGLES ± XX ± XXX ±	CONTRACT NO.
NEXT ASSY	USED ON	MATERIAL	APPROVALS / DATE
APPLICATION	DO NOT SCALE DRAWING	FINISH	DRAWN 12/77
			CHECKED 3-20-78

Millennium Information Systems, Inc.

SCHEMATIC
COMM/RAM

SIZE	CODE IDENT NO.	DRAWING NO.	REV
D		12000015	X

SCALE —— SHEET 3 OF 3

DWG. NO.	SH	REV.
12000170	1	C

REVISIONS				
ZONE	REV.	DESCRIPTION	DATE	APPROVED
	A	PRODUCTION RELEASE ECO 1822	1-9-80	
	B	PER ECO 1964	3/19/80	19 Mar 80
	C	PER ECO 2597	7/29/81	ace

TRACE STORAGE

NOTES: UNLESS OTHERWISE SPECIFIED.
1. RESISTANCE VALUES ARE IN OHMS 1/4W 5%
2. CAPACITANCE VALUES ARE IN MICROFARADS
3. SYMBOL ▭ INDICATES A COMMON PULL-UP RESISTOR, 2.2K.

		S112	1E	1
U17E		S03	9E	1
TP16		LS08	7A	2
RP2		8T98	16D	2
R22				
C5		LS04	12E	1
LAST USED	NOT USED	TYPE	REF. DES.	QTY
REF. DESIGNATIONS		SPARE GATES		

TRACE ADDRESS POINTER

÷3 SHIFT REGISTER

QTY REQD	CODE IDENT	PART OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	MATERIAL SPECIFICATION

PARTS LIST

13000170	RTT
NEXT ASSY	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS ± DECIMALS .XX ± .XXX ± ANGLES ±
MATERIAL
FINISH
DO NOT SCALE DRAWING

CONTRACT NO.

APPROVALS	DATE
DRAWN T. TRACEY	10-5-79
CHECKED	10/26/79
ISSUED	

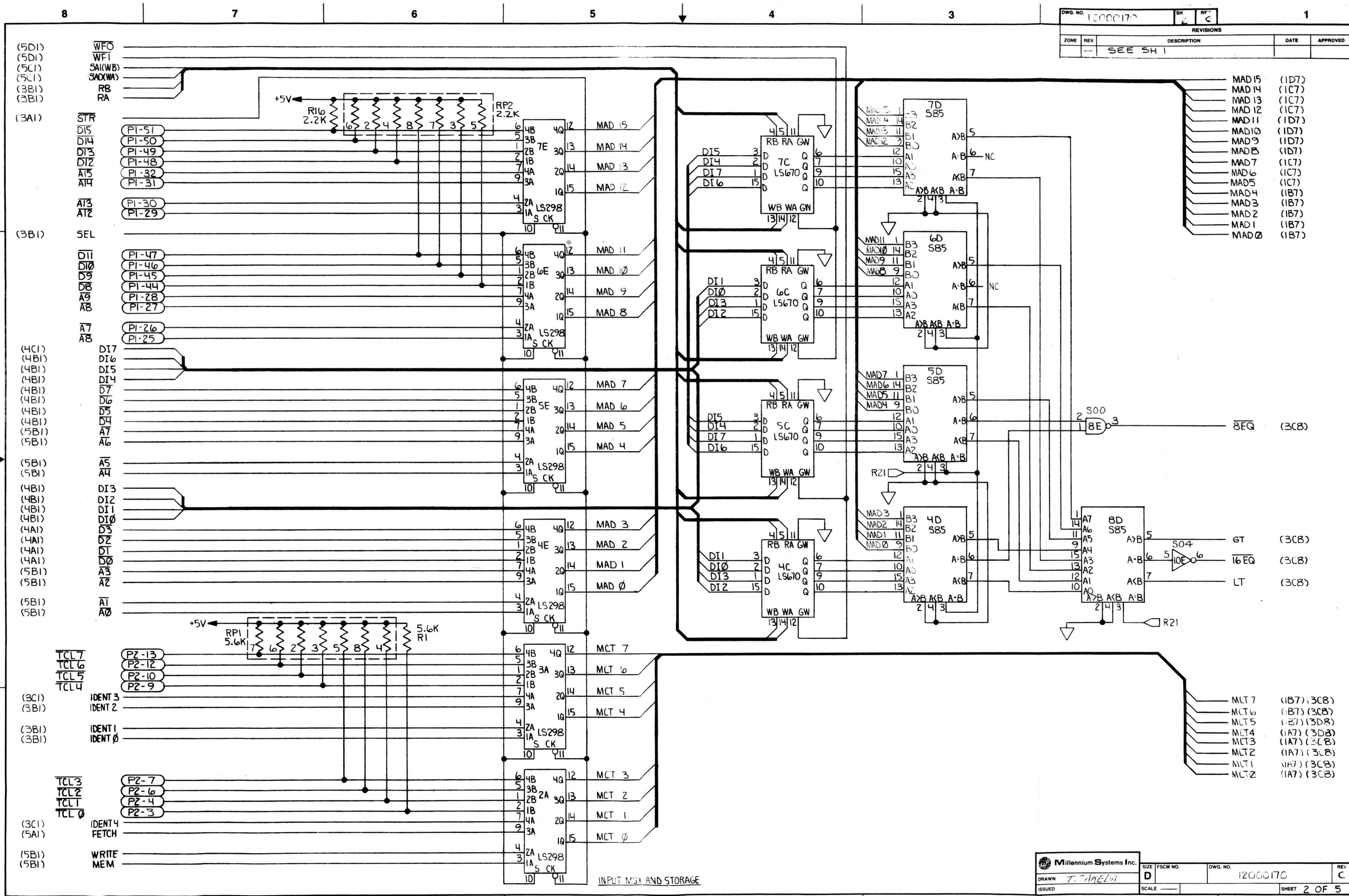
Millennium Systems Inc.

SCHEMATIC
REAL TIME TRACE

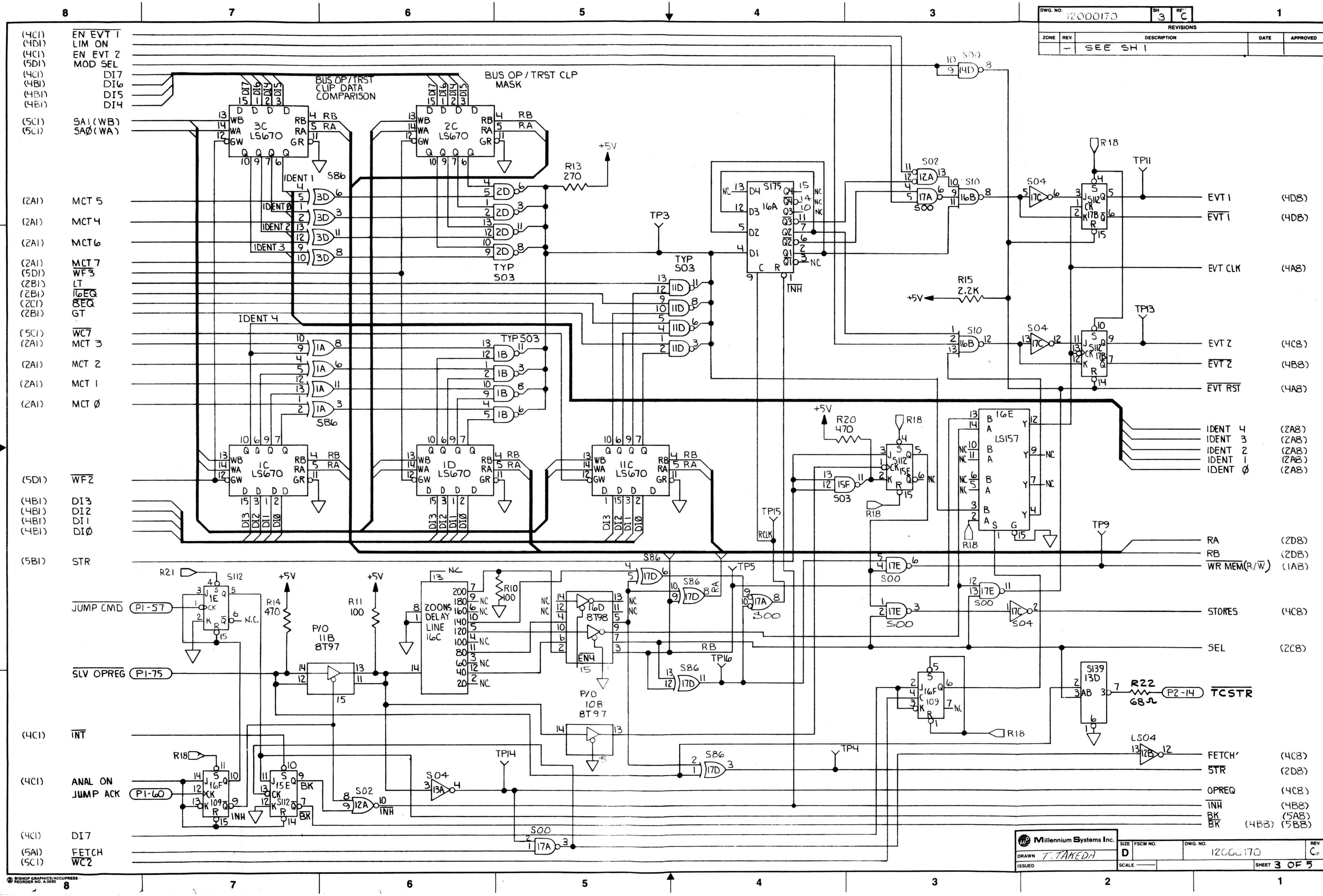
SIZE	FSCM NO.	DWG. NO.	REV.
D		12000170	C

SCALE — SHEET 1 OF 5

BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-5649



DWG. NO. 12000170 SH 2 REV C
REVISIONS
ZONE | REV. | DESCRIPTION | DATE | APPROVED
— SEE SH 1
(5D1) WF0
(5D1) WF1
(5C1) SA1(WB)
(5C1) SA0(WA)
(3B1) RB
(3B1) RA
(3A1) STR
(3B1) SEL
+5V
R16 2.2K
RP2 2.2K
RP1 5.6K
R1 5.6K
P1-51 P1-50 P1-49 P1-48 P1-32 P1-31 P1-30 P1-29
P1-47 P1-46 P1-45 P1-44 P1-28 P1-27 P1-26 P1-25
P2-13 P2-12 P2-10 P2-9
P2-7 P2-6 P2-4 P2-3
TCL7 TCL6 TCL5 TCL4
TCL3 TCL2 TCL1 TCL0
(3C1) IDENT 3
(3B1) IDENT 2
(3B1) IDENT 1
(3B1) IDENT 0
(3C1) IDENT 4
(5A1) FETCH
(5B1) WRITE
(5B1) MEM
7E LS298
6E LS298
5E LS298
4E LS298
3A LS298
2A LS298
7C LS670
6C LS670
5C LS670
4C LS670
7D S85
6D S85
5D S85
4D S85
8D S85
8E S00
10E S04
MAD 15 MAD 14 MAD 13 MAD 12 MAD 11 MAD 10 MAD 9 MAD 8
MAD 7 MAD 6 MAD 5 MAD 4 MAD 3 MAD 2 MAD 1 MAD 0
MCT 7 MCT 6 MCT 5 MCT 4 MCT 3 MCT 2 MCT 1 MCT 0
8EQ (3C8)
GT (3C8)
16EQ (3C8)
LT (3C8)
R21
NC
INPUT MUX AND STORAGE
Millennium Systems Inc.
DRAWN T. TAKEDA
ISSUED
SIZE D
FSCM NO.
DWG. NO. 12000170
REV C
SCALE
SHEET 2 OF 5
BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-3650


DWG. NO. 12000170 SH 3 REV C
REVISIONS
ZONE REV. DESCRIPTION DATE APPROVED
— SEE SH 1
(4C1) EN EVT 1
(4D1) LIM ON
(4C1) EN EVT 2
(5D1) MOD SEL
(4C1) DI7
(4B1) DI6
(4B1) DI5
(4B1) DI4
(5C1) SA1(WB)
(5C1) SAØ(WA)
BUS OP/TRST CLIP DATA COMPARISON
BUS OP/TRST CLP MASK
3C LS670
2C LS670
(2A1) MCT 5
(2A1) MCT 4
(2A1) MCT 6
(2A1) MCT 7
(5D1) WF3
(2B1) LT
(2B1) 16EQ
(2C1) 8EQ
(2B1) GT
(5C1) WC7
(2A1) MCT 3
(2A1) MCT 2
(2A1) MCT 1
(2A1) MCT Ø
(5D1) WF2
(4B1) DI3
(4B1) DI2
(4B1) DI1
(4B1) DIØ
(5B1) STR
IDENT 1
IDENT Ø
IDENT 2
IDENT 3
IDENT 4
S86
R13 270
+5V
TYP S03
TP3
S175 16A
INH
1C LS670
1D LS670
11C LS670
S02
S10
S00
S04
R18
TP11
TP13
S112 17B
EVT 1 (4D8)
EVT 1 (4D8)
EVT CLK (4A8)
R15 2.2K
EVT 2 (4C8)
EVT 2 (4B8)
EVT RST (4A8)
R20 470
S112 15E
15F S03
16E LS157
IDENT 4 (2A8)
IDENT 3 (2A8)
IDENT 2 (2A8)
IDENT 1 (2A8)
IDENT Ø (2A8)
TP15
TP9
RA (2D8)
RB (2D8)
WR MEM(R/W) (1A8)
R21
S112 1E
JUMP CMD P1-57
R14 470
R11 100
P/O 11B 8T97
SLV OPREG P1-75
200NS DELAY LINE 16C
R10 100
16D 8T98
EN4
TP5
TP16
P/O 10B 8T97
STORES (4C8)
SEL (2C8)
16F 109
S139 13D
R22 68Ω
P2-14
TCSTR
(4C1) INT
(4C1) ANAL ON
JUMP ACK P1-60
16F 109
15E S112
BK
TP14
13A
12A
TP4
LS04 12B
FETCH' (4C8)
STR (2D8)
OPREQ (4C8)
INH (4B8)
BK (5A8)
BK (4B8) (5B8)
(4C1) DI7
(5A1) FETCH
(5C1) WC2
17A
Millennium Systems Inc.
DRAWN T. TAKEDA
ISSUED
SIZE D
FSCM NO.
DWG. NO. 12000170
REV. C
SCALE
SHEET 3 OF 5
BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-3650

DWG. NO.	SH	REV
12000170	4	C

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED
	–	SEE SH 1		

(3D1) EVT 1
(3D1) $\overline{\text{EVT 1}}$
(5C1) $\overline{\text{WC2}}$
(3C1) EVT 2
(5C1) $\overline{\text{WC3}}$
(5A1) $\overline{\text{RST}}$
5LV CLK
(3A1) OPREQ
(3A1) FETCH'
(3B1) STORES
10MHZ P1-96
(5C1) SA1 (A1)
(5C1) SA2 (A0)
(5C1) $\overline{\text{RC3}}$ ($\overline{\text{RD}}$)
(5C1) $\overline{\text{WC5}}$ ($\overline{\text{WR}}$)
(5C1) SA0 ($\overline{\text{CS}}$)
(5C1) $\overline{\text{WC0}}$
(3C1) $\overline{\text{EVT 2}}$
(3A1) $\overline{\text{INH}}$
(4C1) EN EVT 2
(5C1) $\overline{\text{RC2}}$
(3A1) $\overline{\text{BK}}$
$\overline{\text{RUN}}$ P1-58
MST RUN P1-84
(1D3) BD7
(1D3) BD6
(1D3) BD5
(1D3) BD4
(1C3) BD1
(1C3) BD3
(1C3) BD$\emptyset$
(1C3) BD2
(5B1) $\overline{\text{RD MOD SEL}}$
(5D1) $\overline{\text{WR MOD SEL}}$
(3C1) $\overline{\text{EVT CLK}}$
(3C1) $\overline{\text{EVT RST}}$

ARM ON
LIM ON
BK ON TRIG ON
DELAY 1 ON
PASS 1 ON
BK ON TRIG 2
DELAY 2 ON
1MHZ
CLOCK SELECT
1KHZ

LIM ON (3D8)
$\overline{\text{DBG INT 29}}$ P1-71
TRIG 1 P3-1
EN EVT 2 (3D8X4B8)
$\overline{\text{EN EVT 1}}$ (3D8)
EN EVT 1 (5B8)
ANAL ON (3A8)
$\overline{\text{INT}}$ (3A8)
TRIG 2 P3-5

STATUS REGISTER
DI7 (2C8)(3D8)(3A8)
DI6 (2C8)(3D8)(5C8)
DI5 (2C8)(3D8)(5B8)
DI4 (2C8)(3D8)(5B8)
DI3 (2B8)(3B8)(5D8)
DI2 (2B8)(3B8)(5D8)
DI1 (2B8X3B8X5D8)
DI$\emptyset$ (2B8X3B8)(5D8)

P1-43 $\overline{\text{D7}}$ D7 (2C8)
P1-42 $\overline{\text{D6}}$ D6 (2C8)
P1-41 $\overline{\text{D5}}$ D5 (2C8)
P1-40 $\overline{\text{D4}}$ D4 (2C8)
MSTR RUN (5D8)
P1-39 $\overline{\text{D3}}$ D3 (2B8)
P1-38 $\overline{\text{D2}}$ D2 (2B8)
P1-37 $\overline{\text{D1}}$ D1 (2B8)
P1-36 $\overline{\text{D$\emptyset$}}$ D$\emptyset$ (2B8)

Millennium Systems Inc.	SIZE D	FSCM NO.	DWG. NO. 12000170	REV C
DRAWN T. TAKEDA	SCALE —			SHEET 4 OF 5
ISSUED				

BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-5650

DWG. NO.	SH	REV.
12000170	5	C

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED
	-	SEE SH 1		

(4B1) DI3
(4B1) DI2
(4B1) DI1
(4B1) DIØ
OPREQ P1-54
(4B1) MSTR RUN
A7 P1-24
A6 P1-23
A5 P1-22
A4 P1-21
A3 P1-20
WRP P1-53
A2 P1-19
A1 P1-18
AØ P1-17
R/W P1-55
(4C1) DI6
M/IO P1-52
(4C1) EN EVT1
(3A1) BK
(4B1) DI5
(4B1) DI4
FETCH P1-82
(3A1) BK
RESET P1-59

S02 13E
30 14E
C3 82PF
R8 14.7K 1% 1/8 W
%02 13C
LS04 12B
450 NS
S00 14D
LS04 12E
S02 13E
S04 15D
S00 14D
S04 15D
LS04 12E
S00 8E
STOP ON EVT 1
MEMS
WRITES
READS
IO'S
FETCH'S
9D LS174
WC1
S04 10E
S04 10E
8T97
P/O 10B
TP8
+5V
TYP S03
10D
10D
10D
10D
9E
9E
9E
TP16
R12 270
+5V

WRITE DECODER
7442A 12D
READ DECODER
P/O S139 14C

RVT FILE REGISTER WRITR DECODR
P/O S139 13D

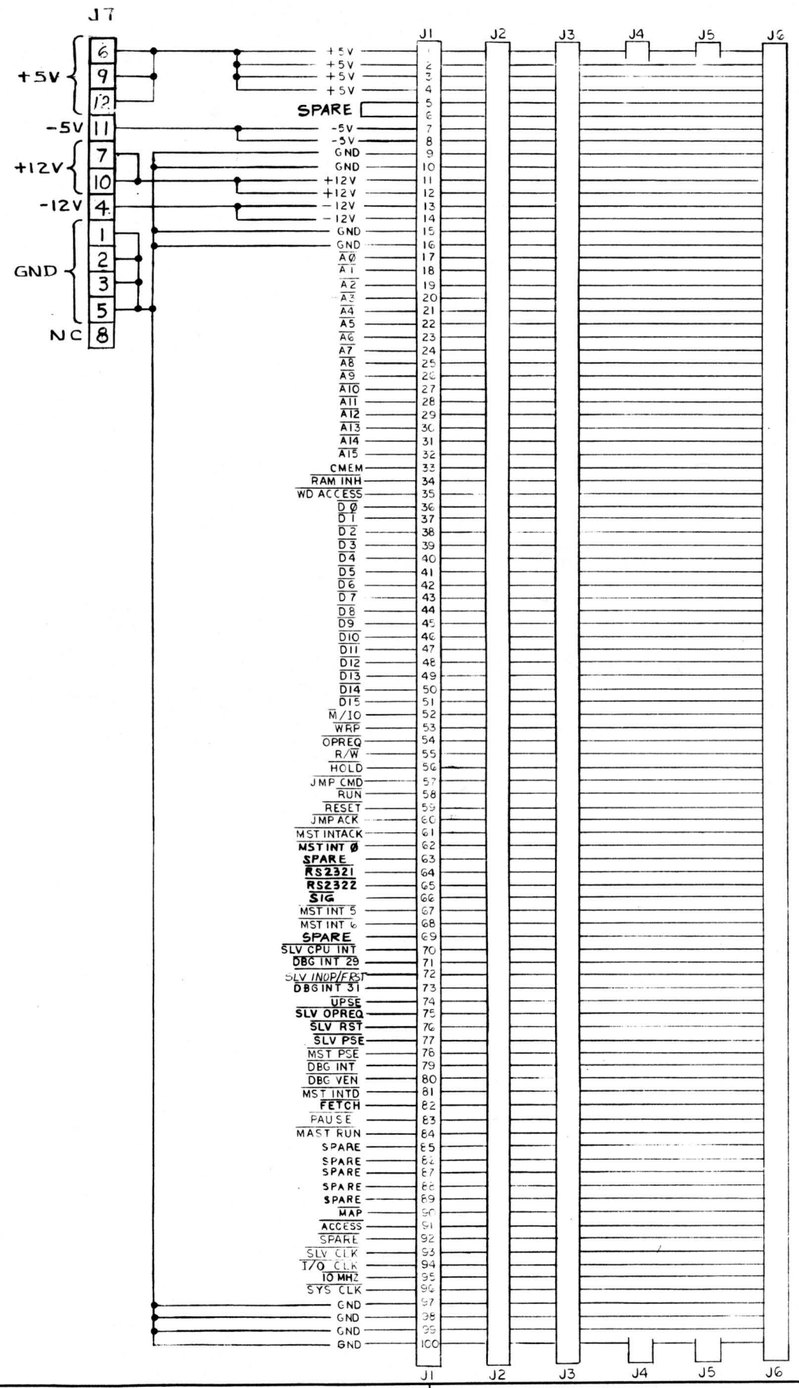
CRY 193 12C
BRW
DC UC
R18
SUBADDRESS COUNTER
S00 14D
LS04 12B
S04 15D
S00 15C

WR. MOD SEL (1A8)(4A8)
MOD SEL (3D8)
WF3 (3C8)
WF2 (3B8)
WF1 (2D8)
WFØ (2D8)
SA2 (4B8)
SA1 (2D8)(3D8)(4B8)
SAØ (2D8)(3D8)(4B8)
WC7 (3C8)
WC5 (4B8)
WC3 (4C8)
WC2 (3A8)(4D8)
WCØ (4B8)
PTRST (1A8)
RC3 (4B8)
RC2 (4B8)
RC1 (1A8)
READ. MOD SEL (4A8)
STR (3B8)
A7 (2C8)
A6 (2C8)
A5 (2B8)
A4 (2B8)
A3 (2B8)
A2 (2B8)
A1 (2B8)
AØ (2B8)
MEM (2A8)
WRITE (2A8)
FETCH (2A8)(3A8)
RST (4C8)

Millennium Systems Inc.	SIZE	FSCM NO.	DWG. NO.	REV.
DRAWN T. TAKEDA	D		12000170	C
ISSUED	SCALE —		SHEET 5 OF 5	

BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-3680

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED
✓	A	PROD REL ECO 1082	5/30/78	
	B	PER ECO 1260	11-21-79	
	C	PER ECO 1595	11-21-79	
	D	PER ECO 1842	19Feb80	



13000001	USA
NEXT ASSY	USED ON

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE
FRACTIONS DECIMALS ANGLES
.XX ± °
.XXX ±

MATERIAL

FINISH

APPLICATION

DO NOT SCALE DRAWING

CONTRACT NO.

APPROVALS	DATE
DRAWN	5/6/77
CHECKED	2-17-78
ENG	2-28-78

Millennium Information Systems, Inc.

SCHEMATIC,
μSA MOTHERBOARD

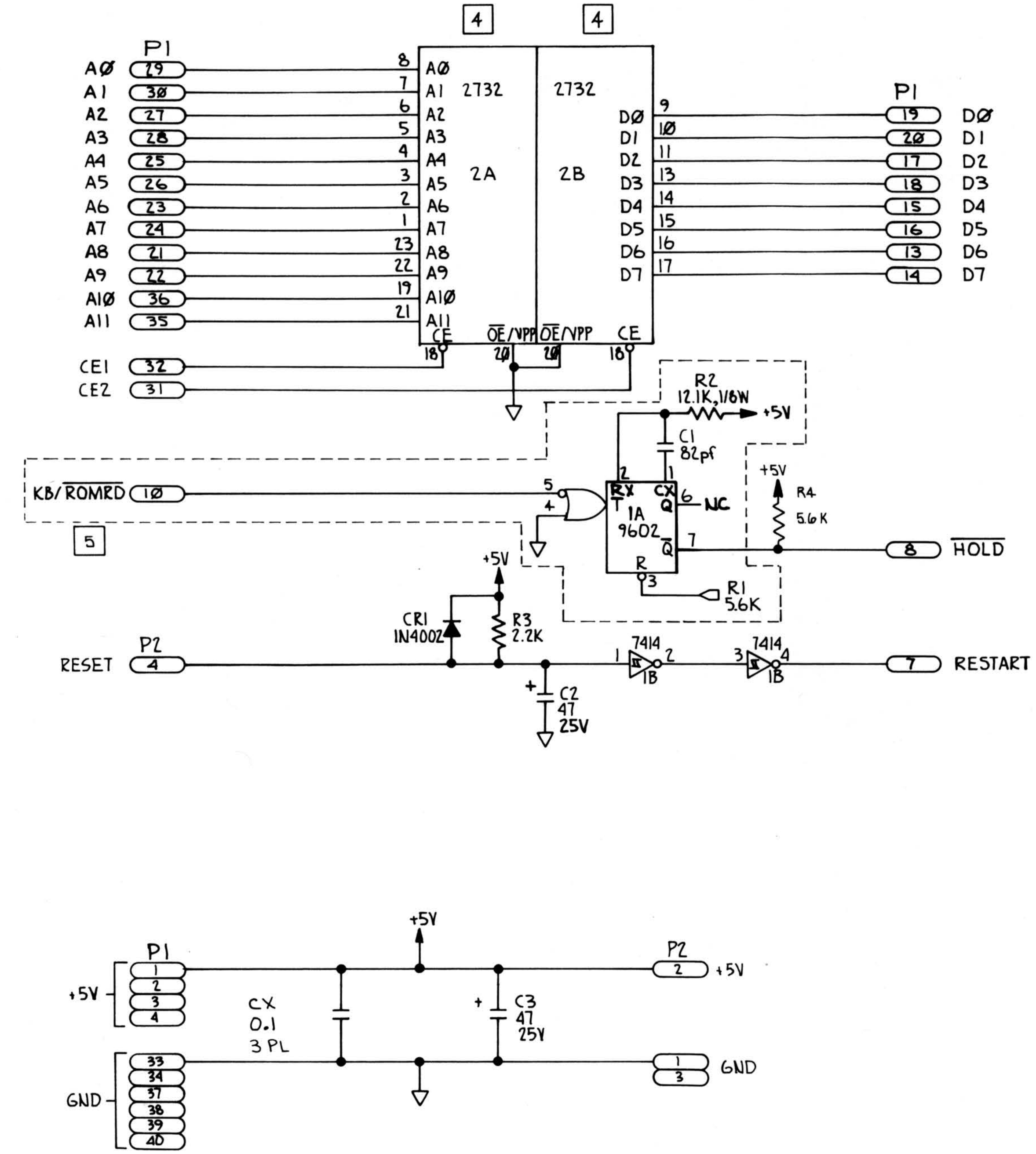
SIZE	CODE IDENT NO.	DRAWING NO.	REV
D		12000001	D

SCALE NONE SHEET 1 OF 1

DWG. NO. 12000236	SH 1	REV F

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED
	01	PROTOTYPE		
	02	GEN. CHG 3-5-81		
	50	PILOT RELEASE ECO #2486 3/10/81	3-10-81	acz
	51	PER ECO 2508 Ray K 5/11/81	5-11-81	acz
	A	PRODUCTION RELEASE ECO# 2568	5/19/81	acz
	B	PER ECO 2626	7/13/81	acz
	C	PER ECO 2659	7/29/81	acz
	D	PER ECO# 2759 Ray K 9/21/81	9/21/81	acz
	E	PER ECO # 2943 1/5/82	1-5-82	
	F	FIX HOLD CKT PER ECO# 2961 1-28-82	2-8-82	



NOTES: (UNLESS OTHERWISE SPECIFIED)

1. RESISTANCE VALUES ARE IN OHMS, 1/4W, 5%.
2. CAPACITANCE VALUES ARE IN MICROFARADS.
3. SYMBOL ▷— INDICATES A COMMON +5V PULL-UP RESISTOR CONNECTED TO ONE OR MORE I.C DEVICES.

4 PROM IN 2A AND 2B ARE EMULATOR DEPENDENT.

5 PARTS INSIDE DOTTED LINES ARE FOR REFERENCE ONLY AND WILL NOT APPEAR ON ASSEMBLY.

R4	C1				
P2	R2				
CR1	1A				
C3	R1	7414	1B	4	
LAST USED	NOT USED	TYPE	REF DES	QTY	
REF DESIGNATIONS		SPARE GATES			

NEXT ASSY	USED ON
13000236	9508
FIRST APPLICATION	

Millennium Systems Inc.

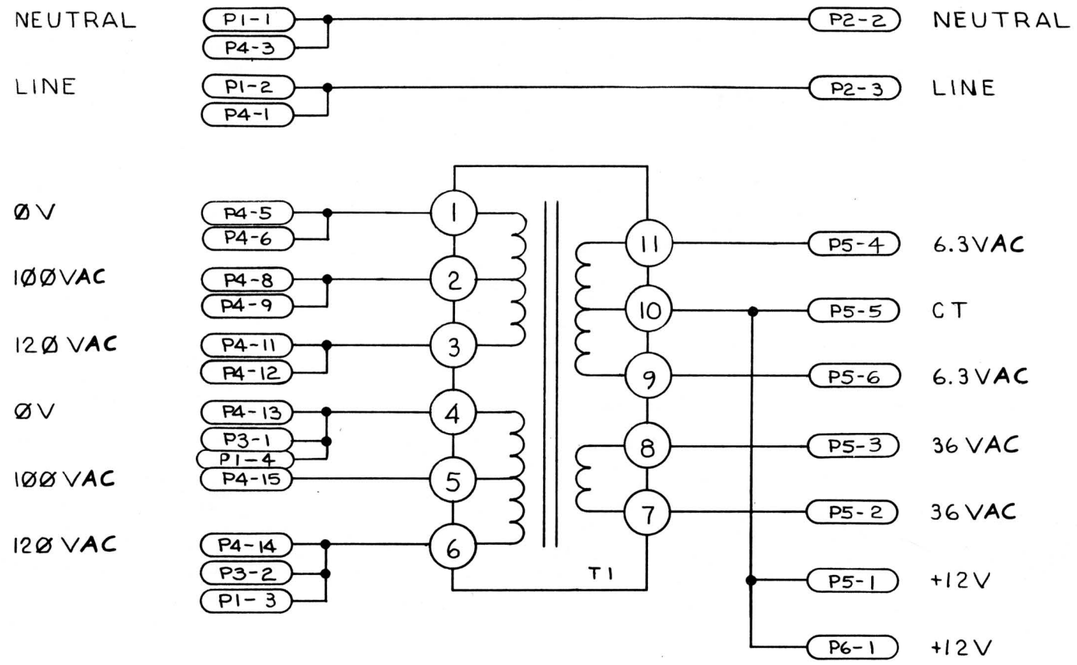
APPROVALS	DATE
DRAWN Ray K	1/14/81
CHECKED	
ENGR	3-11-81

SCHEMATIC
RESTART/EMUL S/W

SIZE D | DWG. NO. 12000236 | REV. F

SHEET 1 OF 1

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED
	A	PRODUCTION RLSE ECO 1082	5/30/78	S. Dillon
	B	REVISE PER ECO 1152	8/23/78	M. Baker
	C	PER ECO 1974	5.28.80	CJL



QTY REQD	CODE IDENT	PART OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION

PARTS LIST

1300 0033	USA
NEXT ASSY	USED ON

APPLICATION

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS ± / DECIMALS .XX ± / .XXX ± / ANGLES ± /

MATERIAL

FINISH

DO NOT SCALE DRAWING

CONTRACT NO.

APPROVALS	DATE
DRAWN	2/20/78
CHECKED	3/8/78
	3/3/78

Millennium Information Systems, inc.

SCHEM.
AC DISTRIBUTION

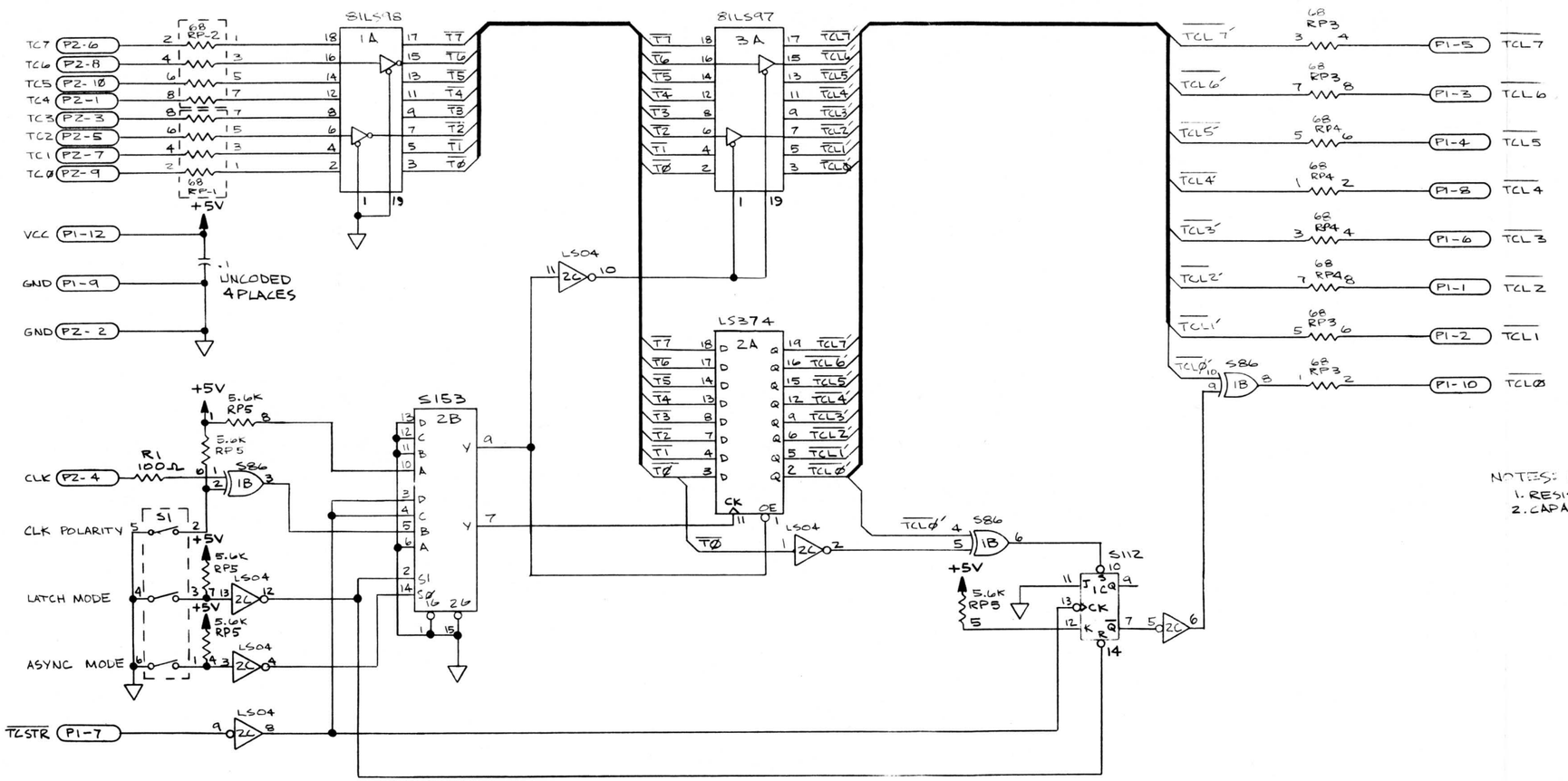
SIZE	CODE IDENT NO.	DRAWING NO.	REV.
C		12000033	C

SCALE | SHEET 1 OF 1

DWG. NO. 12000190	SH 1	REV. B

REVISIONS

ZONE	REV.	DESCRIPTION	DATE	APPROVED
	EO	PILOT RELEASE ECO1822 NK	11/11/80	S Dunlap
	A	PROD. REL ECO 1966	3-18-80	
	B	PER ECO 2200 EG 3-14 80	8-14-80	



NOTES: UNLESS OTHERWISE SPECIFIED.
1. RESISTANCE VALUES ARE IN OHMS, 1/8 W, 5%.
2. CAPACITANCE VALUES ARE IN MICROFARADS.

R1				
RP5		S86	1B	1
S1		S112	1C	1
P2				
LAST USED	NOT USED	TYPE	REF. DES	QTY
REFERENCE DES		SPARE GATES		

QTY REQD	CODE IDENT	PART OR IDENTIFYING NO.	NOMENCLATURE OR DESCRIPTION	MATERIAL SPECIFICATION

PARTS LIST

13000190	RTT
NEXT ASSY	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS DECIMALS ANGLES
XX ±
XXX ±
MATERIAL
FINISH
DO NOT SCALE DRAWING

CONTRACT NO.

APPROVALS	DATE
DRAWN J.B. TORIAS	10/2/79
CHECKED	10-2-79
ISSUED S Dunlap	1/3/80

Millennium Systems Inc.

SCHEMATIC
REAL TIME TRACE PROBE

SIZE	FSCM NO.	DWG. NO.	REV.
D		12000190	B
SCALE —		SHEET 1 OF 1	

BISHOP GRAPHICS/ACCUPRESS REORDER NO. A-3649

DOCUMENT REVIEW FORM

9508 MICROSYSTEM EMULATOR
USERS MANUAL

Publication No. 87000063
January 1982
Release 3.0

Dear Customer:

Your comments concerning this document help us to produce better documentation for you.

GENERAL COMMENTS

- ☐ Easy to read?
- ☐ Well organized?
- ☐ Accurate?
- ☐ Complete?
- ☐ Well illustrated?
- ☐ Suitable for your needs?

SPECIFIC COMMENTS AND CORRECTIONS

Reference	Page

Please send your comments to:

MILLENNIUM SYSTEMS, INC.
19050 Pruneridge Avenue
Cupertino, Calif. 95014
ATTN: Publications

MILLENNIUM SYSTEMS, INC.

PRODUCT ENHANCEMENT REQUEST

1. DATE

2. CUSTOMER INFORMATION

   NAME OF COMPANY
   NAME OF EMPLOYEE
   ADDRESS
   PHONE # _____ EXT. _____

3. PRODUCT NAME (9508, 9520, 9516, 9580, ETC.)

4. DESCRIPTION OF ENHANCEMENT

5. DESCRIPTION OF HOW ENHANCEMENT WOULD BE USED

6. HOW LONG BEFORE YOU MUST HAVE THIS ENHANCEMENT?

   NOW _____
   6 MONTHS _____
   1 YEAR _____
   2 YEARS _____

7. WHAT SHOULD BE THE COST OF THIS ENHANCEMENT?

(ATTACH ADDITIONAL SHEETS AS REQUIRED)

MILLENNIUM SYSTEMS, INC.

SOFTWARE/SYSTEM PROBLEM REPORT

1. DATE

2. CUSTOMER INFORMATION

   NAME OF COMPANY
   NAME OF EMPLOYEE
   ADDRESS — TELEPHONE # _____
   CITY/STATE/ZIP CODE — EXTENSION _____

3. HARDWARE/SOFTWARE CONFIGURATION

   9520

   SERIAL # _____ VOLTAGE _____ HERTZ _____
   MEMORY SIZE: 64K? _____ 112K? _____
   OPERATION SYSTEM _____ VERSION # _____

   9508

   SERIAL # _____ VERSION # _____
   16K? _____
   EMULATOR Z80 _____
   OTHER _____

   8080 _____ 6800 _____
   8085 _____ 6801 _____
   8048/49 _____ 6802 _____
   8021 _____ 6809 _____
   8041 _____

4. HOW ARE THE HOST, 9508, AND UNIT UNDER TEST CONNECTED? PLEASE INDICATE PORT NUMBERS

5. COMMAND OR PROGRAM WHERE PROBLEM OCCURS

   NAME _____ VERSION # _____

6. AFTER POWER ON/RESET, WHAT COMMAND OR SERIES OF COMMANDS WILL REPRODUCE THE ERROR.

7. DESCRIPTION OF THE ERROR

8. PLEASE ATTACH AS MUCH OF THIS OTHER HELPFUL INFORMATION AS POSSIBLE

   SOURCE FILES _____ OBJECT FILES _____
   COMMAND FILES _____ LISTINGS _____
   CONSOLE HARDCOPY _____

(ATTACH ADDITIONAL SHEETS AS REQUIRED)