

64000

**HP 64000
Logic Development
System**

**Emulation with
Internal Analysis
68000/68008**



CERTIFICATION

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

WARRANTY

This Hewlett-Packard system product is warranted against defects in materials and workmanship for a period of 90 days from date of installation. During the warranty period, HP will, at its option, either repair or replace products which prove to be defective.

Warranty service of this product will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all other cases, products must be returned to a service facility designated by HP.

For products returned to HP for warranty service, Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country.

HP warrants that its software and firmware designated by HP for use with an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environment specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HP SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HP SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

ASSISTANCE

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office.

FOLD HERE



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

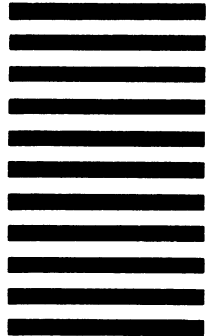
BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD

Logic Product Support Dept.
Attn: Technical Publications Manager
Centennial Annex - D2
P.O. Box 617
Colorado Springs, Colorado 80901-0617



FOLD HERE

Your cooperation in completing and returning this form
will be greatly appreciated. Thank you.

READER COMMENT SHEET

Operating Manual with Service Appendix, Model 64243
Emulation with Internal Analysis 68000/68008
64243-90901, September 1985

Your comments are important to us. Please answer this questionnaire and return it to us. Circle the number that best describes your answer in questions 1 through 7. Thank you.

1. The information in this book is complete:

Doesn't cover enough 1 2 3 4 5 Covers everything
(what more do you need?)

2. The information in this book is accurate:

Too many errors 1 2 3 4 5 Exactly right

3. The information in this book is:

Difficult to find 1 2 3 4 5 Easy to find

4. The Index and Table of Contents are useful:

Missing or inadequate 1 2 3 4 5 Helpful

5. What about the "how-to" procedures and examples:

No help 1 2 3 4 5 Very Helpful

Not enough 1 2 3 4 5 Too many

6. What about the writing style:

Confusing 1 2 3 4 5 Clear

7. What about organization of the book:

Poor order 1 2 3 4 5 Good order

8. What about the size of the book:

Too small 1 2 3 4 5 Too big

Comments: _____

Particular pages with errors? _____

Name: _____

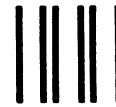
Job title: _____

Company: _____

Address: _____

Note: If mailed outside U.S.A., place card in envelope. Use address shown on other side of this card.

FOLD HERE



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

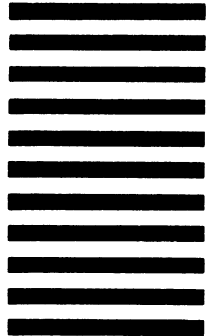
BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD

Logic Product Support Dept.
Attn: Technical Publications Manager
Centennial Annex - D2
P.O. Box 617
Colorado Springs, Colorado 80901-0617



FOLD HERE

Your cooperation in completing and returning this form
will be greatly appreciated. Thank you.

READER COMMENT SHEET

Operating Manual with Service Appendix, Model 64243
Emulation with Internal Analysis 68000/68008
64243-90901, September 1985

Your comments are important to us. Please answer this questionnaire and return it to us. Circle the number that best describes your answer in questions 1 through 7. Thank you.

1. The information in this book is complete:

Doesn't cover enough 1 2 3 4 5 Covers everything
(what more do you need?)

2. The information in this book is accurate:

Too many errors 1 2 3 4 5 Exactly right

3. The information in this book is:

Difficult to find 1 2 3 4 5 Easy to find

4. The Index and Table of Contents are useful:

Missing or inadequate 1 2 3 4 5 Helpful

5. What about the "how-to" procedures and examples:

No help 1 2 3 4 5 Very Helpful

Not enough 1 2 3 4 5 Too many

6. What about the writing style:

Confusing 1 2 3 4 5 Clear

7. What about organization of the book:

Poor order 1 2 3 4 5 Good order

8. What about the size of the book:

Too small 1 2 3 4 5 Too big

Comments: _____

Particular pages with errors? _____

Name: _____

Job title: _____

Company: _____

Address: _____

Note: If mailed outside U.S.A., place card in envelope. Use address shown on other side of this card.

SAFETY SUMMARY

The following general safety precautions must be observed during all phases of operation, service, and repair of this instrument. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the instrument. Hewlett-Packard Company assumes no liability for the customer's failure to comply with these requirements.

GROUND THE INSTRUMENT.

To minimize shock hazard, the instrument chassis and cabinet must be connected to an electrical ground. The instrument is equipped with a three-conductor ac power cable. The power cable must either be plugged into an approved three-contact electrical outlet or used with a three-contact to two-contact adapter with the grounding wire (green) firmly connected to an electrical ground (safety ground) at the power outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE.

Do not operate the instrument in the presence of flammable gases or fumes. Operation of any electrical instrument in such an environment constitutes a definite safety hazard.

KEEP AWAY FROM LIVE CIRCUITS.

Operating personnel must not remove instrument covers. Component replacement and internal adjustments must be made by qualified maintenance personnel. Do not replace components with the power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

DO NOT SERVICE OR ADJUST ALONE.

Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

DO NOT SUBSTITUTE PARTS OR MODIFY INSTRUMENT.

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification of the instrument. Return the instrument to a Hewlett-Packard Sales and Service Office for service and repair to ensure that safety features are maintained.

DANGEROUS PROCEDURE WARNINGS.

Warnings, such as the example below, precede potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

WARNING

Dangerous voltages, capable of causing death, are present in this instrument. Use extreme caution when handling, testing, and adjusting.

SAFETY SYMBOLS

General Definitions of Safety Symbols Used on Equipment or in Manuals.



Instruction manual symbol: the product is marked with this symbol when it is necessary for the user to refer to the instruction manual in order to protect against damage to the instrument.



Indicates dangerous voltage (terminals fed from the interior by voltage exceeding 1000 volts must be so marked).



OR



Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.



Low-noise or noiseless, clean ground (earth) terminal. Used for a signal common, as well as providing protection against electrical shock in case of a fault. A terminal marked with this symbol must be connected to ground in the manner described in the installation (operating) manual, and before operating the equipment.



OR



Frame or chassis terminal. A connection to the frame (chassis) of the equipment which normally includes all exposed metal structures.



Alternating current (power line).



Direct current (power line).



Alternating or direct current (power line).

WARNING

The **WARNING** sign denotes a hazard. It calls attention to a procedure, practice, condition or the like, which, if not correctly performed, could result in injury or death to personnel.

CAUTION

The **CAUTION** sign denotes a hazard. It calls attention to an operating procedure, practice, condition or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product.

NOTE:

The **NOTE** sign denotes important information. It calls attention to procedure, practice, condition or the like, which is essential to highlight.



OPERATING MANUAL WITH SERVICE APPENDIX

**EMULATION WITH INTERNAL ANALYSIS
68000/68008**

**© COPYRIGHT HEWLETT-PACKARD COMPANY 1985
LOGIC SYSTEMS DIVISION
COLORADO SPRINGS, COLORADO, U. S. A.**

ALL RIGHTS RESERVED

Manual Part No. 64243-90901
E1085

PRINTED: October 1985

PRINTING HISTORY

Each new edition of this manual incorporates all material updated since the previous edition. Manual change sheets are issued between editions, allowing you to correct or insert information in the current edition.

The print date changes only when each new edition is published. Minor corrections or additions may be made as the manual is reprinted between editions. Vertical bars in a page margin indicates the location of reprint corrections.

First Edition August, 1985 (64243-90901 E0885)
Second Edition October, 1985 (64243-90901 E1085)

SOFTWARE VERSION NUMBER

Your HP 64000 software is identified with a version number in the form YY.XX. The version number is printed on a label attached to the software media or media envelope. This manual applies to the following:

Model HP 64243A	Version 1.XX
Model HP 64244A	Version 1.XX

Within the software version number, the digit to the left of the decimal point indicates the product feature set. This manual supports all software versions identified with this same digit.

The digits to the right of the decimal point indicate feature subsets. These feature subsets normally have no affect on the manual. However, if you subscribe to the "Software Material Subscription" (SMS), these subset items are covered in the "Software Response Bulletin" (SRB).

REPAIR NUMBERS

This manual applies to components of the HP 64243 and HP 64244 Emulation Subsystem with the following repair number prefixes:

Model Number	Current Repair Prefix	Other Repair Numbers
HP 64243	2626A	N/A
HP 64244	2626A	N/A

TABLE OF CONTENTS

USING THIS MANUAL

GENERAL	xv
MANUAL CONTENTS GUIDE	xvi
UNDERSTANDING THE EXAMPLES USED IN THIS MANUAL	xvi

Chapter 1: GENERAL INFORMATION

OVERVIEW	1-1
SAFETY CONSIDERATIONS	1-1
GENERAL	1-1
WHAT IS AN EMULATION SYSTEM?	1-1
Physical Description	1-1
Emulation Subsystem Hardware	1-2
Emulation System Software	1-2
Emulation System Manual	1-2
Functional Description	1-3
WHAT DOES THE EMULATOR ALLOW YOU TO DO?	1-5
How Are These Tasks Implemented?	1-5
WILL THE EMULATOR SYSTEM RUN INTERACTIVELY WITH OTHER HP 64000 SYSTEM MODULES?	1-7
WHAT EFFECT WILL THE EMULATOR HAVE ON YOUR PROGRAM?	1-8
WHAT IS HAPPENING WHILE YOUR PROGRAM IS RUNNING?	1-8
During Normal Flow of the Program	1-8
During Emulation Monitor Program Control	1-9
WHAT DOES THE EMULATOR DO TO YOUR MICROPROCESSOR SYSTEM?	1-9
Functional Transparency	1-9
Timing Transparency	1-9
Electrical Transparency	1-10
YOUR PART IN THE EMULATION PROCESS	1-10
SOFTWARE MATERIALS SUBSCRIPTION	1-11
Software Updates	1-11
Reference Manual Updates	1-11
Software Problem Reporting	1-11
Software Release Bulletins	1-11
Software Status Bulletins	1-11
General User Information	1-11

Chapter 2: INSTALLING HARDWARE AND SOFTWARE

OVERVIEW	2-1
INTRODUCTION	2-1
PRE-INSTALLATION INSPECTION	2-4

TABLE OF CONTENTS (Cont'd)

Required Equipment, Recommended Additional Equipment, and Additional Optional Equipment	2-4
INSTALLING EMULATION SYSTEM HARDWARE	
INTO AN HP 64100 LOGIC DEVELOPMENT STATION	2-6
Configuration of the Boards in the Station	2-6
Configuration of the Memory Boards	2-7
Installation Instructions	2-8
INSTALLING EMULATION SYSTEM HARDWARE	
INTO AN HP 64110 LOGIC DEVELOPMENT STATION	2-15
Configuration of the Boards in the Station	2-15
Configuration of the Memory Boards	2-15
Installation Instructions	2-15
Installing the Emulation Probe Cable	
Connector into the Target System Socket	2-20
Target System Microprocessor Connector Compatibility	2-20
Installing Into a DIP Socket (DIP)	2-21
Installing into a Pin Grid Array (PGA) Socket	2-22
TURNING ON THE DEVELOPMENT STATION	2-23
LOADING EMULATION SYSTEM SOFTWARE	2-24
Backing Up Your Software	2-24
Loading Software in Clustered Stations Configuration	2-24
Configuring A Flexible Disc For Stand-Alone Operation	2-25
FORMAT A NEW FLEXIBLE DISC	2-25
DEFINE DISC SOFTWARE CONFIGURATIONS	2-26
COPY THE MODULES ONTO NEW DISCS	2-26
REMOVING EMULATION SOFTWARE	
FROM THE OPERATING SYSTEM	2-29
PERFORMING OPERATION VERIFICATION	2-29
OPERATING THE EMULATION SYSTEM	
IN STAND-ALONE CONFIGURATION	2-30
REMOVING THE EMULATOR CONTROL BOARD	2-31
REPACKAGING FOR SHIPMENT	2-32
Other Packaging Materials	2-32

Chapter 3: GETTING STARTED

OVERVIEW	3-1
GENERAL	3-1
INTRODUCTION	3-1
UNDERSTANDING THE EXAMPLES	3-2
ASSIGNING YOUR USERID	3-2
SINGLE-MODULE SYSTEMS -VS- MULTI-MODULE SYSTEMS	3-3
SOFTKEYS	3-4
Monitor Level Softkeys	3-4
Emulation System Softkeys	3-4

TABLE OF CONTENTS (Cont'd)

YOUR PROGRAM MODULE	3-5
FILES YOU WILL CREATE	3-5
ORGANIZING YOUR PROGRAM MODULES	3-6
Copying the 680XX Programs to Your Userid	3-6
Looking at Your Sample Program	3-9
INITIATING A SYSTEM "LOG" COMMAND FILE	3-9
ASSEMBLING AND LINKING THE PROGRAM MODULES	3-9
Assembling Modules	3-9
Linking Modules	3-10
GAINING ACCESS TO THE EMULATION SUBSYSTEM	3-12
ANSWERING THE EMULATION CONFIGURATION QUESTIONS	3-15
LOADING EMULATION MEMORY	3-16
ENDING THE "LOG" COMMAND FILE	3-16
BUILDING SYSTEM COMMAND FILES	3-16
Logging Commands to a Command File	3-17
Typing The Commands Into a Command File	3-18
Typing a System Command File	3-19
Typing an Emulation Command File	3-20
USING THE EMULATOR	3-20
To See Real-Time Tracing of Processor Activity	3-20
To Break into Emulation Monitor and See Registers	3-21
Stepping	3-22
USING THE COMMAND FILES	3-23

Chapter 4: PREPARING PROGRAM MODULES FOR EMULATION

OVERVIEW	4-1
GENERAL	4-1
INTRODUCTION	4-1
USING THE ASSEMBLER	4-2
The Relationship of Linker Mapping and Emulation Memory Mapping	
USING THE 680XX EMULATION MONITOR PROGRAM	4-4
Emulation Monitor Memory Requirements (68000)	4-4
Emulation Monitor Memory Requirements (68008)	4-5
WHY SO MANY FILES HAVE THE SAME NAME	4-11
WHERE TO LOCATE THE MONITOR	4-12

Chapter 5: ANSWERING EMULATION CONFIGURATION QUESTIONS

GENERAL	5-1
INTRODUCTION	5-1
ADDRESS CONVENTIONS	5-2
ACCESSING THE CONFIGURATION QUESTIONS	5-2
Measurement System/Emulate Command Syntax	5-3
Configuring/Reconfiguring the Emulation Pod	5-3
ANSWERING THE CONFIGURATION QUESTIONS	5-7
Selecting the Card Slot	5-7
Selecting the Clock	5-8
Microprocessor clock rate greater than 10MHz?	5-8

TABLE OF CONTENTS (Cont'd)

Selecting Real-Time/Nonreal-Time Runs	5-8
Selecting/Modifying a Memory Configuration	5-9
MODIFYING A CURRENT CONFIGURATION	5-10
SELECTING THE MEMORY CONFIGURATION	5-10
Mapping Memory	5-11
Memory Map Definition	5-12
Emulation Monitor Memory Requirements	5-13
Display Organization	5-13
Entering Memory Blocks	5-15
Setting the Default	5-18
Deleting Blocks	5-19
Ending the Mapping Session	5-19
Configuring Simulated I/O	5-19
Configuring the Emulator Pod	5-21
Interlock emulation memory DTACK with user DTACK?	5-21
Enable Bus Error on emulation memory accesses?	5-22
Enable emulator processor interrupts ?	5-22
Enable emulator DMA transfers ?	5-22
Enable DMA transfers to emulation memory ?	5-22
Enable tracing of DMA memory transfers ?	5-22
Enable tracing of DMA tags ?	5-23
Trap number for software break ?	5-23
Enable emulator use of INT7 ?	5-23
Enable Interrupt level 7 sharing ?	5-23
Enable tristate delay on halts ?	5-23
Configuring for Interactive Measurements	5-24
Naming Your Emulation Configuration Command File	5-26
CONFIGURATION SWITCHES	5-27

Chapter 6: USING THE EMULATOR--EXAMPLES

OVERVIEW	6-1
INTRODUCTION	6-1
EXECUTION	6-1
RUNNING THE PROGRAM	6-2
PROCESSOR RESET	6-2
PROCESSOR STATUS MESSAGES	6-2
ANALYSIS STATUS INFORMATION	6-3
Using the HP 64302A Internal Analysis Board	6-3
MEMORY RESOURCES DURING EMULATION	6-4
OUT-OF-CIRCUIT EMULATION	6-4
Guidelines for Out-of-Circuit Emulation	6-5
IN-CIRCUIT EMULATION	6-5
Emulation Memory and Target System Memory	6-5
Guidelines for In-Circuit Emulation	6-5

TABLE OF CONTENTS (Cont'd)

BEGINNING AND ENDING EMULATION	6-6
EMULATION EXAMPLES	6-7
Demonstration Configuration	6-7
To Display Registers:	6-9
To Step Registers <#STEPS> From <ADDRESS>:	6-10
To Display Memory - Blocked Byte:	6-11
To Display Memory - Mnemonic:	6-12
To Display Memory - Absolute Byte:	6-13
To Display Memory - Real:	6-14
To Display Memory Blocked Byte Offset:	6-15
To Modify and Display Memory - Byte:	6-16
USING ANALYSIS COMMANDS	6-18
ANALYSIS EXAMPLES	6-19
To See Real-time Tracing of Processor Operations	6-19
To Trace After <ADDRESS>:	6-20
To Display Trace Absolute:	6-21
To Display Trace Status Binary:	6-22
To Display Trace Status Mnemonic:	6-23

Chapter 7: COMMAND SUMMARY AND SYNTAX

INTRODUCTION	7-1
OPERATIONAL COMMANDS	7-2
Command Line Comment Delimiter	7-2
DISPLAY AND LIST COMMANDS	7-3
Display and List Command Options	7-3
Memory Data	7-3
Register Contents	7-5
Trace Information	7-5
Global and Local Symbols	7-6
Software Breakpoint Table	7-6
ANALYSIS AND INTERACTIVE COMMANDS	7-7
USING ANALYSIS COMMANDS	7-8
Data Qualification	7-9
ANALYSIS MODES	7-10
BUS_CYCLE_DATA MODE	7-10
EXECUTION_DATA MODE	7-10
BREAK COMMAND SYNTAX	7-16
DISPLAY/LIST COMMAND SYNTAX	7-17
DISPLAY/LIST GLOBAL_SYMBOLS SYNTAX	7-19
DISPLAY/LIST IO_PORT SYNTAX	7-20
DISPLAY/LIST LOC_SYM SYNTAX	7-22
DISPLAY/LIST MEMORY SYNTAX	7-23
DISPLAY/LIST REGISTERS SYNTAX	7-26
DISPLAY/LIST SOFTWARE_BREAKPOINTS SYNTAX	7-28
DISPLAY/LIST TRACE SYNTAX	7-29
EMULATE SYNTAX	7-32
END SYNTAX	7-33
EXECUTE SYNTAX	7-34
HALT SYNTAX	7-35

TABLE OF CONTENTS (Cont'd)

LOAD SYNTAX	7-36
MEASUREMENT_SYSTEM SYNTAX	7-37
MODIFY SYNTAX	7-39
MODIFY ACCESS_SIZE_TO SYNTAX	7-40
MODIFY ANALYSIS_MODE_TO	7-41
MODIFY CONFIGURATION SYNTAX	7-42
MODIFY IO_PORT SYNTAX	7-43
MODIFY MEMORY SYNTAX	7-44
MODIFY REGISTER SYNTAX	7-46
MODIFY SOFTWARE_BREAKPOINTS SYNTAX	7-47
RESET SYNTAX	7-48
RUN SYNTAX	7-49
SPECIFY SYNTAX	7-51
STEP SYNTAX	7-52
STOP_TRACE SYNTAX	7-53
STORE SYNTAX	7-54
TRACE SYNTAX	7-56
WAIT SYNTAX	7-63

Chapter 8: THE EMULATION MONITOR PROGRAM

OVERVIEW	8-1
GENERAL	8-1
INTRODUCTION	8-1
THE BREAK FUNCTION AND THE EMULATION MONITOR	8-2
EMULATION MONITOR	8-2
MODIFYING THE MONITOR TO USE SOFTWARE BREAKPOINTS	8-4
CUSTOMIZING THE EMULATION MONITOR	8-5
Emulation Monitor Memory Requirements (68000)	8-8
Emulation Monitor Memory Requirements (68008)	8-9
Linking The Emulation Monitor	8-9
EMULATION MONITOR FLOWCHART	8-10
EMULATION MONITOR SOURCE PROGRAM	8-10

Chapter 9: EMULATION AND ANALYSIS OPERATIONAL THEORY

OVERVIEW	9-1
GENERAL	9-1
INTRODUCTION	9-1
EMULATION OVERVIEW	9-1
MULTI-MODULE EMULATION AND ANALYSIS	9-5
Plug-in Leads	9-7
EMULATOR OPERATING CONSIDERATIONS	9-11
Software Breakpoints	9-11
Using Software Break in Real Time	9-11

TABLE OF CONTENTS (Cont'd)

Chapter 10: PRINTER SIMULATED I/O

OVERVIEW	10-1
OPENING THE PRINTER FILE	10-2
WRITING TO THE PRINTER	10-2
CLOSING THE PRINTER FILE	10-3
PRINTER EXAMPLE PROGRAM	10-5

Chapter 11: DISPLAY SIMULATED I/O

OVERVIEW	11-1
OPENING THE DISPLAY FILE	11-2
SCROLLING	11-2
Roll To/Write Line 18	11-2
LINE & COLUMN	11-3
Selecting The Starting Line/Column	11-3
Writing From The Starting Line/Column	11-3
CLOSING THE DISPLAY FILE	11-4
DISPLAY EXAMPLE PROGRAM	11-7
PROCEDURE DELAY;	11-8

Chapter 12: KEYBOARD SIMULATED I/O

OVERVIEW	12-1
OPENING THE KEYBOARD FILE	12-2
READ IN PROCESS - HP 64000 RESPONSE	12-2
OUTPUT AVAILABLE -HP 64000 RESPONSE	12-3
CLOSING THE KEYBOARD FILE	12-3

Chapter 13: DISC FILE SIMULATED I/O

OVERVIEW	13-1
CREATING A NEW FILE	13-3
WRITING A RECORD	13-3
Writing First Record	13-4
Writing Additional Records	13-4
CLOSING THE FILE	13-4
ACCESSING EXISTING FILES - OPENING THE FILE	13-4
SELECTING A RECORD	13-5
Automatic Selection Of Record 1,2,3...Etc.	13-5
Advancing N Records	13-5
Backup N Records	13-5
Position To Record N	13-5
Rewind To Record One	13-6

TABLE OF CONTENTS (Cont'd)

READING THE RECORD	13-6
CHANGING THE FILE NAME	13-6
DELETING THE FILE	13-7
DISC FILE EXAMPLE PROGRAM	13-15
PROGRAM DISC_1;	13-16

Chapter 14: RS-232 SIMULATED I/O

OVERVIEW	14-1
OPENING THE RS-232 FILE	14-2
INITIALIZING THE 8251 USART	14-3
Asynchronous Mode	14-5
Synchronous Mode/Single Sync Character	14-6
Synchronous Mode/Double Sync Character	14-7
COMMAND TO 8251 USART	14-7
STATUS FROM 8251	14-8
WRITING TO THE 8251 USART	14-9
Writing A Single Byte	14-9
Using A Buffer To Write Multiple Bytes	
Write Record	
Update Write Buffer	14-9
Closing The Write Buffer	14-12
READING FROM THE 8251 USART	14-13
Reading A Single Byte	14-13
Using A Buffer To Read Multiple Bytes	
Read Record	
Update Read Buffer	14-13
Closing The Read Buffer	14-17
UPDATING READ/WRITE BUFFERS	14-17
CLOSING THE RS-232 FILE	14-17

Appendix A: SYNTACTICAL VARIABLE DEFINITIONS

Appendix B: 680XX REGISTER FORMAT AND NAMES

Appendix C: GLOSSARY OF SOFTKEY LABELS

Appendix D: STATUS, ERROR, AND SOFTKEY PROMPT MESSAGES

Appendix E: SIMULATED I/O ERROR CODES

TABLE OF CONTENTS (Cont'd)

Appendix F: RESTRICTIONS WHEN USING EXECUTION_DATA

EFFECTS OF EXECUTION_DATA MODE F-1
USING THE "TRACE FUNCTION" OF THE 680XX F-2
USING UNIMPLEMENTED INSTRUCTIONS F-3
DMA CYCLES USING EXECUTION_MODE F-3

LIST OF ILLUSTRATIONS

Figure 1-1. 68000/68008 Emulation Systems 1-0
Figure 1-2. Emulation System Functional Block Diagram 1-4
Figure 1-3. Development System Tools 1-6
Figure 2-1. HP 64100 Key Features 2-2
Figure 2-2. HP 64110 Key Features 2-3
Figure 2-3. RFI Ground Bracket Assembly Parts 2-10
Figure 2-4. RFI Ground Bracket Installation (HP 64100) 2-12
Figure 2-5. Emulation Control Board Connections 2-12
Figure 2-6. Ground Bar Clamp Installation (HP 64100) 2-14
Figure 2-7. HP 64110 Installation Details 2-17
Figure 2-8. Ground Bar Clamp Installation (HP 64110) 2-19
Figure 2-9. HP 64110 Memory, Emulation, and Intermodule Bus Cabling 2-19
Figure 2-10. Installing the Emulation Probe Microprocessor
Figure 2-11. Installing the Emulation Probe Microprocessor
Figure 2-12. I/O Bus Configuration Display 2-23
Figure 2-13. Option Test Display. 2-30
Figure 3-1. Exception Vector Table (Commented) 3-7
Figure 3-2. Exception Vector Table (Un-commented) 3-8
Figure 3-3. Linker Output Listing 3-11
Figure 3-4. Utility Keys Used For Transportation (from "emulate") 3-13
Figure 3-5. Utility Keys Used For Transportation (from "meas_sys") 3-14
Figure 3-6. Trace After 002000H 3-21
Figure 3-7. RUN Emulation Command File Trace 1 3-23
Figure 3-8. RUN Emulation Command File Trace 2 3-24
Figure 4-1. Example Link Programs 4-3
Figure 4-2A. Mon_680XX Relocatable File - Record #1 4-4
Figure 4-2B. Mon_68008 Relocatable File - Record #1 4-5
Figure 4-3. Memory Map For Test 4-7
Figure 4-4. Example Link_Sym Listing 4-11
Figure 4-5. Example Memory Map 4-12
Figure 5-1. Emulation Map For Test Program 5-12
Figure 5-2. Memory Map Display 5-14
Figure 5-3. Sample Overlay Mapping #1 5-17
Figure 5-4. Sample Overlay Mapping #2 5-18
Figure 5-5. Configuration Switches 5-27
Figure 7-1. Memory Contents - Hexadecimal and ASCII 7-4
Figure 7-2. Memory Contents - Mnemonic 7-5

TABLE OF CONTENTS (Cont'd)

Figure 7-3. Register Contents	7-6
Figure 7-4. Trace Memory Display	7-7
Figure 7-5. Bus_Cycle_Data Trace Page 1	7-11
Figure 7-6. Bus_Cycle_Data Trace Page 2	7-12
Figure 7-7. Execution Cycle Data Page 1	7-13
Figure 7-8. Execution Data Trace Page 2	7-14
Figure 8-1. Sample Modified IO_FUNCTION	8-8
Figure 8-2A. Mon_68000 Relocatable File - Record #1	8-8
Figure 8-2B. Mon_68008 Relocatable File - Record #1	8-9
Figure 8-3. Emulation Monitor Flowchart	8-11
Figure 9-1. HP 64000 Logic Development System Simplified Block Diagram	9-3
Figure 9-2. Emulation Probe (Pod)	9-8
Figure 9-3. Pod Plug-in Leads	9-8
Figure 10-1. Printer Interface	10-1
Figure 10-2. Program PRINT_SIO	10-5
Figure 11-1. Display Interface Diagram	11-1
Figure 11-2. Display Techniques	11-4
Figure 11-3. Display/ Keyboard Simulated I/O Program	11-10
Figure 12-1. Keyboard Interface Diagram	12-1
Figure 12-2. Keyboard Interface Sequence	12-7
Figure 13-1. Disc Drive Interface Diagram	13-1
Figure 13-2. Program Disc_1	13-16
Figure 14-1. RS-232 Interface Diagram	14-2
Figure 14-2. 8251 Initialization Formats	14-3
Figure 14-3. Command Mode Instruction Format	14-4
Figure 14-4. Asynchronous Mode Instruction Format	14-5
Figure 14-5. Synchronous Mode Instruction Format	14-6
Figure 14-6. 8251 Status Word Format	14-8
Figure 14-7. Writing RS-232 Record Interface Sequence	14-10
Figure 14-8. Reading RS-232 Record Interface Sequence	14-15
Figure B-1. Status Register Format	B-2
Figure F-1. Example One Word Branch-Around Code	F-2

LIST OF TABLES

Table 2-1. RFI Ground Bracket Assembly Parts List	2-9
Table 5-1. Number of Mapper Blocks vs. Available Memory	5-15
Table 6-1. 680XX Status Definition	6-3
Table 6-2. Softkey Labels and Mnemonic Status Definitions	6-4
Table 7-1. "And" Function Results	7-8
Table 10-1. Printer Control Codes	10-4
Table 11-1. Display Control Codes	11-5
Table 12-1. Keyboard Control Codes	12-4
Table 12-2. Command Word Codes	12-6
Table 13-1. Disc File Type Numbers and Names	13-2
Table 13-2. Disc File Control Codes	13-8
Table 14-1. RS-232 Control Codes	14-18

TABLE OF CONTENTS (Cont'd)

Table D-1. Status Messages D-1
Table D-2. Error Messages D-3
Table D-3. Softkey Prompt Messages D-5
Table E-1. Simulated I/O Error Codes - General Definitions E-1

NOTICE

CONDUCTIVE FOAM OR PLASTIC OVER EMULATOR PINS MAY CAUSE ERRATIC OPERATION.

The emulator and preprocessor user assembly pins are covered at the time of shipment with either a conductive foam wafer or a conductive plastic pin protector. This is done for two reasons: 1) to protect the user interface circuitry within the emulator or preprocessor from electro-static discharge (ESD), 2) to protect the delicate gold plated pins of the probe assembly from damage due to impact.

Both the foam and plastic protection devices are conductive. This may cause erratic performance of the emulation or analysis system during operation, and also during option_test performance verification. Therefore, it is recommended that the foam or plastic device be removed before using the emulation or analysis system or before running option_test performance verification.

When not using the emulator or preprocessor, the foam or plastic assembly should be replaced to retain protection for the probe pins and protection from ESD.

USING THIS MANUAL

GENERAL

This manual includes both operating instructions for the emulator and reference material. The reference material is gathered in the final chapters: Command Summary and Syntax, The Emulator Monitor Program, Emulation and Analysis Operational Theory, Simulated I/O, and the seven Appendices.

If you have never used a HP 64000 series Emulator, you will want to take special note of Chapter 3, Chapter 4, Chapter 5, and Chapter 6. The tutorial information in Chapter 3 is most effective as a training tool if you sit down at a HP 64000 development station, and, following the manual's instructions, enter and step through each of the examples provided.

If you are an experienced (HP 64000) emulator user, you will want to take special note of Chapter 6 and Chapter 7 which include information specific to the 68000/68008 emulator.

MANUAL CONTENTS GUIDE

Several guides are provided to help you find the information you need:

- A table of contents precedes the body of the manual.
- The first page of each chapter lists the topics that are covered in that chapter. Each chapter is dedicated to a single purpose.
- An index concludes the manual.

The 68000/68008 Emulator/Analyzer Manual is divided into thirteen chapters, defined as follows:

Chapter 1 introduces basic emulator concepts and provides a general overview of an emulation system.

Chapter 2 is the installation information. This chapter describes, in detail, how to install the emulation system in your HP 64100 and HP 64110. Also included is information on how to configure the system for "stand-alone" operation.

Chapter 3 is a tutorial. This chapter walks you through the emulation process and explains, in general terms, some of the more basic emulation features.

Chapter 4 is a review of the assembling and linking processes which are necessary before emulation can begin.

Chapter 5 explains each of the emulation configuration questions and explains the options available to answer the questions. Chapter 6 explains how to enter and exit the emulation program, provides guidelines for various modes of emulation, provides information on the 68000/68008 processor, and concludes with emulation examples.

Chapter 6 explains how to enter and exit the emulation program, provides guidelines for various modes of emulation, provides information on the 68000/68008 processor, and concludes with emulation examples.

Chapter 7 provides a functional description of the emulation system commands and the syntax for using them.

Chapter 8 provides a detailed description of the emulation monitor program provided with your emulation system software.

Chapter 9 provides the emulation and analysis operational theory.

Chapters 10 through 14 describe the simulated I/O feature of the HP 64000 Logic Development System.

UNDERSTANDING THE EXAMPLES USED IN THIS MANUAL

The examples provided throughout this manual use the following structure:

PRESS (or press) *edit* MODULE RETURN.

PRESS or press-- means you should enter a command by selecting the softkeys and/or typing in any file names or other variables which are not provided in the softkey selections.

edit -- softkeys will appear in italics. Usually you will not be prompted to use the ---ETC--- softkey to search for the appropriate softkey template.

MODULE -- this is the name of a file which you must type in. Softkeys are not provided for this type of selection since it is variable. However, a softkey prompt such as, <FILE> will appear as a softkey selection.

RETURN -- this indicates that the RETURN key, located on the keyboard, should be pressed.

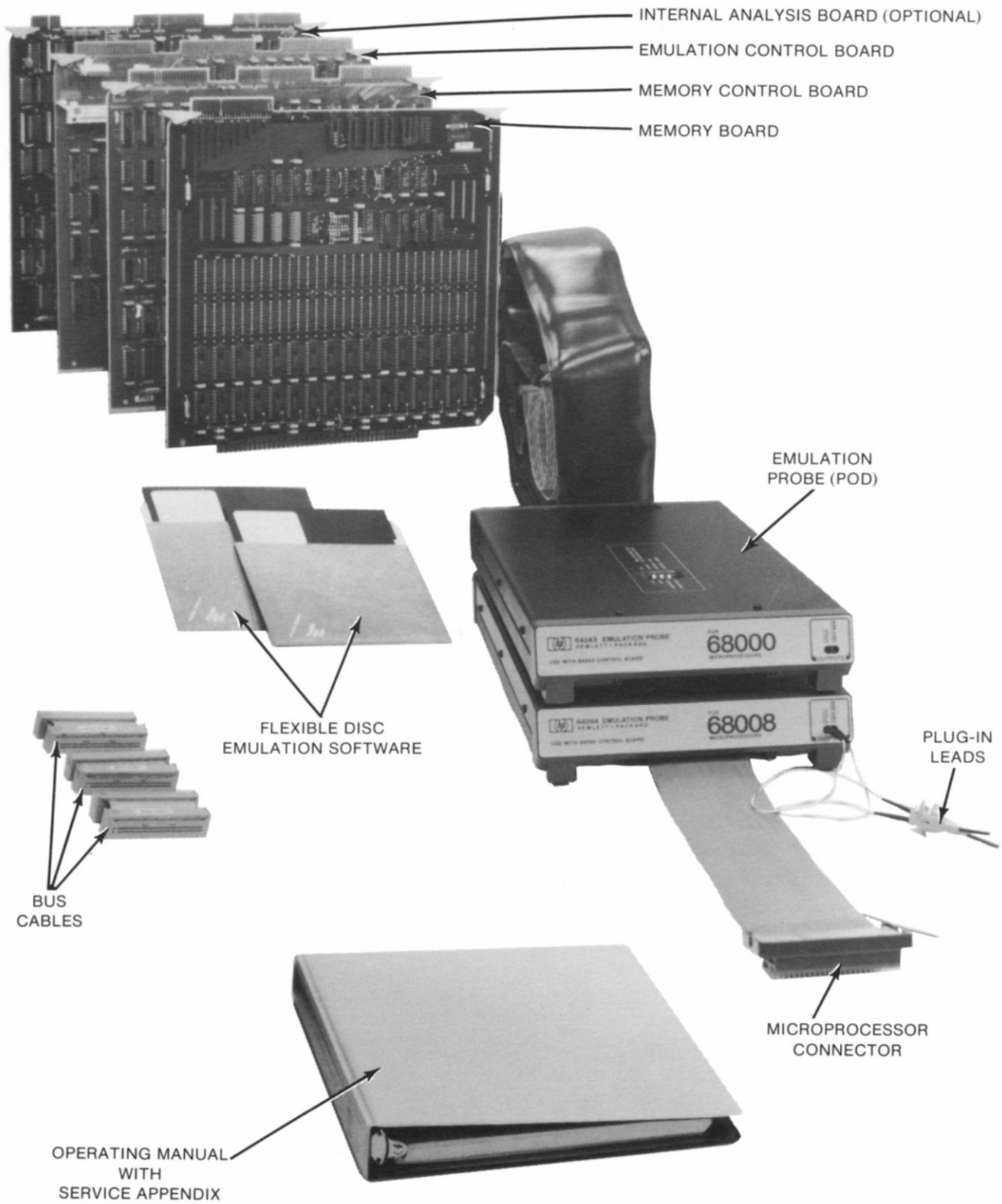


Figure 1-1. 68000/68008 Emulation Systems

Chapter 1

GENERAL INFORMATION

OVERVIEW

This chapter will answer these questions:

- What is an emulation system?
- What does an emulator allow you to do?
- Will the emulator system run interactively with other HP 64000 system modules?
- Will using an emulator have an effect on your program?
- What is happening while your program is running?
- What does the emulator do to your microprocessor system?
- What is your role in the emulation process?
- What can Software Materials Subscription do for you?

SAFETY CONSIDERATIONS

This product is a Safety Class 1 instrument (provided with a protective earth terminal) and meets safety standards IEC 348. Review the instrument and this manual for safety markings and instructions before operating the instrument.

GENERAL

In this chapter, whenever 680XX is mentioned (even as part of a filename) both the 68000 and the 68008 are assumed unless there are differences between the two processors. In that case, the differences will be noted. When the name 680XX appears in any program listings, just insert the appropriate name of your processor.

WHAT IS AN EMULATION SYSTEM?

Physical Description

The emulation system is a separate functional module within the HP 64000 development station structure. The operator controls the emulation system via the station keyboard and display. The emulation system consists of four hardware modules plus the flexible disc emulation software package and technical manuals. The four hardware modules are:

- Emulation Subsystem
- Emulation Memory Subsystem
- Internal Analyzer
- Emulation/Memory/Analysis Bus Cable(s)

The emulation system hardware listed above consists of circuit boards which are installed in the development station card cage, the emulation probe (pod) which is connected by cabling to the development station, and bus cables to connect the boards. These parts, along with the software and technical manuals required, are shown in figure 1-1 and are listed below.

Emulation Subsystem Hardware

- (1) Emulator Probe (Pod)/Control Board HP 64243 (68000) or HP 64244 (68008)

Emulation Memory Subsystem Hardware

- (1) Emulation Memory Control Board HP 64155A
- (1) Memory (Static RAM) Board(s) HP 64152/153/154A or HP 64161/162/163A

Internal Analyzer

To perform the internal analysis functions described in this manual you must also have the following option:

- (1) Wide Internal Analysis Board HP 64302A

NOTE

The entire emulation system may be used in conjunction with "external" analysis and timing systems for more sophisticated measurements. External analysis and timing systems are available as plug-in options for the HP 64000 system. See your local HP Sales & Service office for more details on the external analysis systems.

Emulation/Memory/Analysis Bus Cable(s)

- (3) Bus Cable(s)

Emulation System Software

- (2) Flexible Disc Emulation System Software (HP 64243-120XX) (68000) or (HP 64244-120XX) (68008)

Emulation System Manual

- (1) 68000/68008 Emulation with Internal Analyzer Operating Manual with Service Appendix HP 64243-90901

Accessory Software Required

In order to assemble/compile and link your program modules you will need to have the following software:

- (1) Flexible disc containing 68000/08/10 Assembler/Linker HP 64845AF
- (3) Flexible disc containing 68000/08/10 Pascal Compiler HP 64815AF
- (2) Flexible disc containing 68000/08/10 C Compiler HP 64819AF

Functional Description

The goal of the emulator is to look just like the microprocessor which will eventually control your system, as seen by your target system hardware. At the same time it must be capable of giving you complete and immediate insight into the clock-by-clock operation of the system. The function, signal quality, signal timing, loading, drive capacity, and other factors at the plug-in connector should be indistinguishable from the same factors that would be present if the actual processor were being used. This characteristic is referred to as transparency, which is discussed later in this chapter under the heading entitled "What Does The Emulator Do To Your Microprocessor System?"

The emulation system and its components, functionally illustrated in figure 1-2, are described briefly in the following paragraphs. A further, in-depth description, is given in Chapter 8, "Emulation and Analysis Operational Theory".

Interaction between the HP 64000 editor, the assembler/linker, and the emulator run-time environment is so easy that error producing programming practices, such as code patching, are unnecessary.

An important feature is that the emulation and analysis functions can operate independently of the HP 64000 operating system. That is, once the emulation and analysis equipment has been configured and set into operation, the equipment can operate without interaction from the operating system.

After your (target system) hardware is developed, the emulator can be used in-circuit, alone, or with other products to debug your target system hardware and to integrate the program modules with your target system hardware.

Emulation is accomplished by a two-processor system. The host microprocessor, located in the development station, is used by the HP 64000 operating system. The emulation processor, located in the emulation pod, is used to emulate your target system processor. Since the host and emulation processors do not share the operating system resources, the HP 64000 can easily be adapted to support different target systems by changing the emulation processor.

The emulator allows you to replace the 680XX processor in your target system with a device which performs like the 680XX, but which is easily controlled by you from the development station. This is done through the emulation pod and microprocessor connector which are part of the cable extending from the emulation control board. The pod contains the processor that drives your target system. The microprocessor connector is plugged into your target system processor socket. (Your target system processor must be removed to accommodate the connector.)

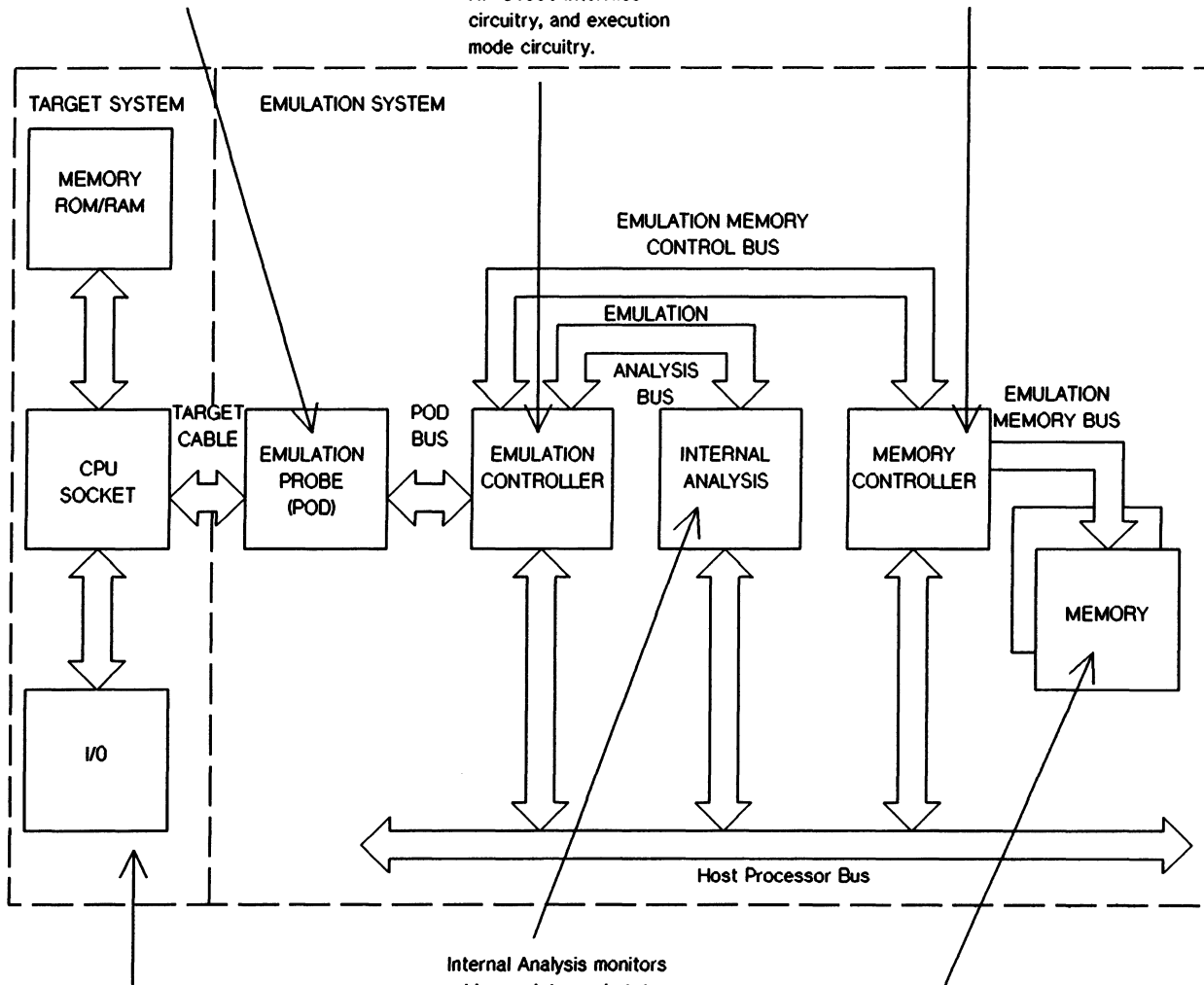
The emulator allows you to access and modify internal processor registers, and locations or blocks of memory. In addition, you can access code instruction-by-instruction. The emulation control board controls the interaction between the HP 64000 operating system software and the emulation

hardware. This board is the major interface between the emulation pod and the memory/analysis buses. The HP 64000 host processor can control the emulation processor "reset" and the bus activity stopping functions directly through this board. However, the two main functions of the control board are: to convert unique pod timing signals to compatible memory and analysis bus signals and to provide a channel to the pod for hardware configuration.

The Emulator Probe includes the emulation processor buffers, control circuitry, the timing section (converts timing signals of the emulation processor to standard requirements for HP 64000 system), and JAM circuitry.

Emulation Controller includes buffers, HP 64000 interface circuitry, and execution mode circuitry.

Memory Controller provides a dual port (emulation/HP 64000) to the emulation memory cards and also contains an address map to distinguish between emulation mapped and user mapped addresses.



Target System represents the system under design. Probably includes microprocessor, control circuitry, memory, and I/O circuits.

Internal Analysis monitors address, data, and status on emulation analysis bus. Can capture status and cause breakpoint on occurrence of some trigger or end of analysis measurement.

Emulation Memory consists of static RAM circuitry.

Figure 1-2. Emulation System Functional Block Diagram

The emulation system includes emulation memory implemented in static RAM. Emulation memory can be used in place of your target system ROM or RAM. Thus, program modules which will ultimately reside in your target system ROM may be developed and thoroughly tested, then permanently stored in ROM. The memory requirements of the emulation system are satisfied by the Static RAM Board(s). Each static RAM board has the capacity to hold a maximum of 128K words of random access memory. Options to these memory boards are available in which only a fourth or half of the board is loaded. These options are discussed in appendix H2 in the paragraph entitled "Configuration of the Memory Boards". Emulation memory may be configured to exist at any location in the memory address space of the emulation processor.

Emulation/user memory is controlled by the memory control board. The memory control board monitors the emulation memory bus to determine the type of memory that is to be accessed, (i.e., emulation memory or user memory). The memory control board is the major interface point from emulation memory to the HP 64000 operating system. The host processor communicates with the emulation processor by transferring data to and from emulation memory. Data transfer is accomplished through the memory controller board into the static RAM boards. In addition to providing an access port for the host processor into emulation memory, the memory controller contains a hardware mapper that is programmed to map the emulation processor memory address space into emulation/user/guarded and RAM/ROM memory spaces.

The emulator may be used for software development before your target system hardware is finished for out-of-circuit emulation. Program modules may be run by the emulator, and trace measurements may be made by the emulation system internal analysis board. The internal analysis board is the equivalent of a logic analyzer. It accepts your trigger specifications, then monitors the emulation analysis bus to determine if the specified event has occurred. When the event occurs, the analysis board makes a trace of 256 states of program execution and stores them in a trace memory. Trace data is available to the HP 64000 and is displayed on the development station screen. The analysis board is required for hardware breakpoints and for 'run_until' action.

WHAT DOES THE EMULATOR ALLOW YOU TO DO?

An emulator is just one of the tools available in a complete development system (see figure 1-3). The tasks facilitated by an emulator are software debug, hardware debug, and hardware and software integration.

How Are These Tasks Implemented?

These tasks are implemented via the basic emulator features:

- **Program Loading and Execution.**
Your code developed on the HP 64000 using the editor, compilers, assembler, and linker can be loaded into memory by means of the emulator.
- **Run/Stop Controls.**
- **Memory Display/Modification.**
You can display locations or blocks of memory and modify any that can be changed.

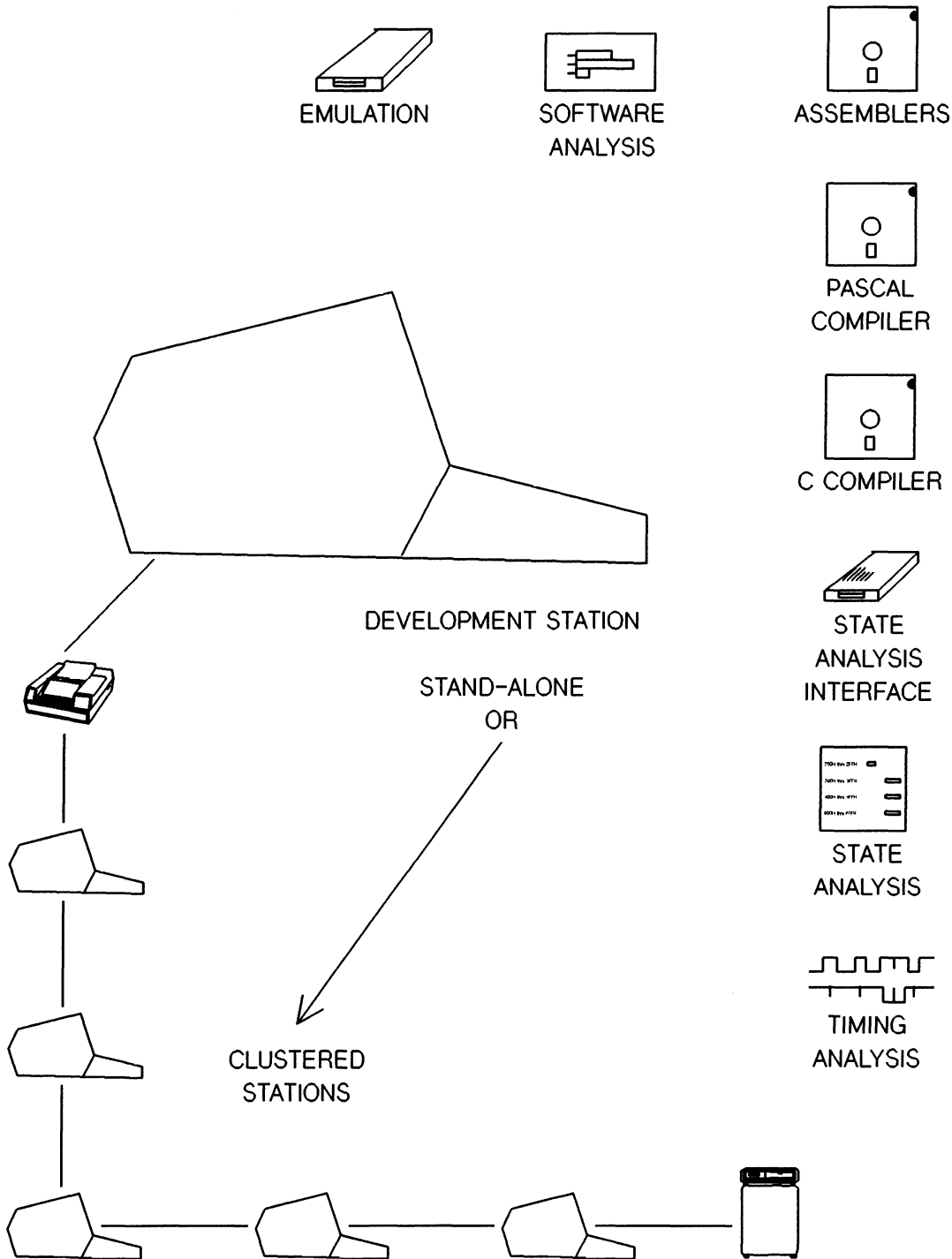


Figure 1-3. Development System Tools

- **I/O Ports Display/Modification.**
You can display I/O port address locations and values, and modify the values.
- **Global and Local Symbols Display.**
- **Internal Resource Display/Modification.**
Allows you to display internal resources of the processor, such as registers, and to modify them, if desired.
- **Analysis (with optional internal analyzer board).**
Allows you real time observation and display of activity on the emulation processor bus.
- **Program Stepping.**
Allows you to execute code instruction-by-instruction, gaining access to the internal machine states between instructions.
- **Resource Mapping.**
Allows you to map memory to emulation processor address space.
- **Memory Characterization.**
You can assign emulation memory ROM or RAM. You can test "ROM" code without using ROM hardware.
- **Breakpoint Generation.**
You can transfer program execution to a monitor routine with the occurrence of a particular machine state or range of states.
- **Clock Source Selection.**
Provides internal clock generation, which can be used in lieu of your target system clock.

WILL THE EMULATOR SYSTEM RUN INTERACTIVELY WITH OTHER HP 64000 SYSTEM MODULES?

The HP 64000 system allows the use of emulation and analysis features in an interactive manner between an emulator and another module; a module that could be another emulator or a state or timing analyzer. Interaction allows the integration of development work on multi-processor designs, or more elaborate and detailed analysis of a design, or both.

In particular, the capabilities that are supported include: simultaneous initiation of multiple measurements, using the results of one measurement to control another, and coordinating execution of a program with the initiation of a measurement.

WHAT EFFECT WILL THE EMULATOR HAVE ON YOUR PROGRAM?

The effect which the emulator will have on your program will depend upon the emulator operation which you select for execution at a particular time. The emulator will never permanently alter your program, but it may effect the execution of your program.

Depending upon the emulator operation selected for execution, the emulator will be operating in one of two modes: real-time or nonreal-time. "Real-time" in this case is not based on whether wait states are inserted; instead real-time refers to the continuous execution of your target system program without interference from the development system (except as instructed by you - and then, only for specific operations).

Interference occurs when a break to the emulation monitor program is initiated either by you or automatically. The emulation monitor is a program which allows you access to the internal registers and memory of the 680XX processor.

Whenever the emulator is running under control of the emulation monitor program it is no longer executing your program in real time. The emulation monitor program is described in more detail in later paragraphs in this section.

Features that cannot be performed in real-time mode without breaking into the emulation monitor program are the following:

Memory Accesses--display, list, load, modify, and store.

Register Accesses--display, list, and modify.

Symbol Accesses--display and list. For local symbols these commands will cause the present values to be displayed/listed for symbols in emulation memory. For symbols in target memory, present values are displayed as "*****".

The above features can be accessed while the emulator is configured for real-time mode by causing a break to the emulation monitor. This happens when you press the *break* softkey and the **(RETURN)** key, or when you attempt to access guarded memory, or when you create an analysis break condition, such as "trace break_on meas_comp" or "trace break_on trigger".

Basically, features that can be performed in real-time mode are listed below. A more detailed explanation of these features is given in chapter 4.

run, some display, some modify, specify, execute,

trace, step, break, load trace, stop_trace

WHAT IS HAPPENING WHILE YOUR PROGRAM IS RUNNING?

During Normal Flow of the Program

During the normal flow of your program, the emulation processor in the emulation pod generates address information for each cycle. One function of this hardware differentiates your target system and emulation resources based on the address. If the pod identifies a target system resource with

the current address, the data path buffers between your target system and the emulator processor are enabled. Likewise, if the address has been mapped to emulation resource space, the data path buffers between the emulation processor and the emulation bus resources are enabled.

As your program runs, the internal analysis board circuitry is observing the activity on the emulation analysis bus. Under your control, the analysis board can be instructed to store this program flow which can later be displayed without interrupting the real-time flow of the program.

During Emulation Monitor Program Control

The main emulation functions of the emulator are achieved by seizing control of the emulation processor from your program and transferring control to a monitor program that can extract the processor's internal information.

The emulator monitor program provides the link between the emulation processor and the HP 64000 Logic Development System Station. This emulation monitor is a program written in 680XX code. The monitor is located totally within emulation memory because this is the only memory directly accessible by the development station.

The monitor program is actually constructed of a number of separate routines. Some of these routines are executed automatically whenever the monitor program is entered. These routines extract the internal processor information that existed at the time of entry. The station can then display this extracted information on the station screen for examination by the operator. If, for instance, the monitor program was entered after the execution of a certain one of your program instructions, the internal machine state that existed at that time would be available.

WHAT DOES THE EMULATOR DO TO YOUR MICROPROCESSOR SYSTEM?

As mentioned earlier, in the functional description of the emulator, the goal of the emulator is to look just like the microprocessor which will eventually control your system, as seen by your target system hardware. At the same time it must be capable of giving you complete and immediate insight into the clock-by-clock operation of the system. The function, signal quality, signal timing, loading, drive capacity, and other factors at the plug-in pins should be indistinguishable from the same factors that would be present if the actual processor were being used. This characteristic is referred to as transparency.

Functional Transparency

Functional transparency refers to the ability of the emulator to function in the same way as the processor you have selected to use in your (target) system when the emulator is connected to the target system. Total functional transparency requires that the emulator will execute your program, generate outputs, and respond to inputs in the same manner as the actual target processor.

Timing Transparency

Timing transparency deals with the timing relationships between signals at your target system plug-in. The internal clock in the emulator is 10 MHz. and will allow (68000 only) the target clock to be up to 12.5 MHz. The 68008 emulator will allow a maximum clock speed of 10MHz.

Electrical Transparency

Electrical transparency refers to the electrical characteristics of the emulator target plug pins compared to the pins of the actual target processor. These characteristics include such things as rise and fall times, input loading, output drive capacity, and transmission line considerations. The electrical requirements and timing of the 680XX emulation target plug pins are designed to be compatible with the 680XX microprocessor it replaces in your target system.



When the emulator detects a guarded memory access or other illegal condition, or when you request an access to memory which causes the emulator to break into the monitor, the emulator stops executing user code and enters the monitor. Thus, if you have circuitry that can be damaged because the emulator is not executing code, you should exercise special caution. For example, you should configure the emulator to restrict to real-time runs, and you should not break into the monitor.

YOUR PART IN THE EMULATION PROCESS

What are the steps which you must go through to use the emulator?

- Write the program modules.
- Compile or assemble your program modules.
- Assemble the 680XX emulation monitor program.
- Understand the memory map of your system.
- Understand where to map the emulation monitor program for your target system.
- Link your program modules and the monitor program.
- Obtain a list of the addresses assigned by the linker.
- Answer the emulation configuration questions.
- Understand how the emulator features can be used to solve your debugging problems.

All of the above items are addressed in this manual in the chapters that follow. It is suggested that you read the information pertaining to these items before you attempt your first emulation session.

SOFTWARE MATERIALS SUBSCRIPTION

Hewlett-Packard offers a Software Materials Subscription (SMS) to provide you with the most timely and comprehensive information concerning your HP 64000 Logic Development System. This service can maximize the productivity of your HP system by ensuring that you have the latest product enhancements, software revisions, and software reference manuals.

Consult with your local HP Field Representative for a complete list of available software update products (HP 64XXXAU), one-time product updates (HP 64XXXAX), and current prices.

By purchasing SMS, you will obtain the following:

- Software Updates
- Reference Manual Updates
- Software Problem Reporting
- Software Release Bulletins
- Software Status Bulletins
- General User Information

Software Updates

Software Updates may address specific anomalies in HP software or enhance the capability of the HP software in your system.

Reference Manual Updates

Reference manual updates assure that you always have the most recent documentation on a timely basis, and are aware of how to use any new features on the latest software releases.

Software Problem Reporting

Software problem reporting is provided so that you may inform HP of a discrepancy or problem found in the HP 64000 software or documentation.

Software Release Bulletins

Software Release Bulletins document all fixes and enhancements that are incorporated in the latest release of the HP 64000.

Software Status Bulletins

Software status bulletins contain timely information on the reported operational status of HP software and documentation. These bulletins also provide temporary corrections or ways to work around anomalies in HP software which have been located by HP personnel or HP 64000 users. You may reference these bulletins to see if a solution is already documented.

General User Information

General user information is documentation that contains operational tips, programming techniques, application notes, latest listings of software products and reference manuals, and other items of general interest to HP 64000 users.

NOTES

Chapter 2

INSTALLING YOUR EMULATION HARDWARE AND SOFTWARE

OVERVIEW

Chapter 2 will:

- Provide pre-installation inspection instructions.
- Provide a complete list of emulation system hardware, software, and manuals.
- Show you how to install an RFI ground bracket.
- Show you how to install the emulation system hardware.
- Show you how to install the emulation probe cable.
- Show you how to turn on the development station.
- Show you how to load the emulation system software.
- Explain how to customize a flexible disc for an HP 64000 station in stand alone configuration.
- Show you how to remove emulation system software modules.
- Provide reminders for stand alone operating configuration.

INTRODUCTION

If you are installing your HP 64000 Logic Development System, including peripherals, as a new installation, refer to the HP 64000 Installation and Configuration Manual for instructions concerning the installation of the system work stations, disc, and printer. Also, refer to "Pre-installation Inspection" given in this section. After you have done these, install the emulation system as instructed later in this section.

NOTE

All references to the 68000 microprocessor and the HP Model 64243 Emulator in this chapter are equally applicable to the 68008 microprocessor and the HP Model 64244 Emulator respectively, unless otherwise noted.

Emulator/Analyzer 68000/68008
Installing Your Emulation Hardware and Software

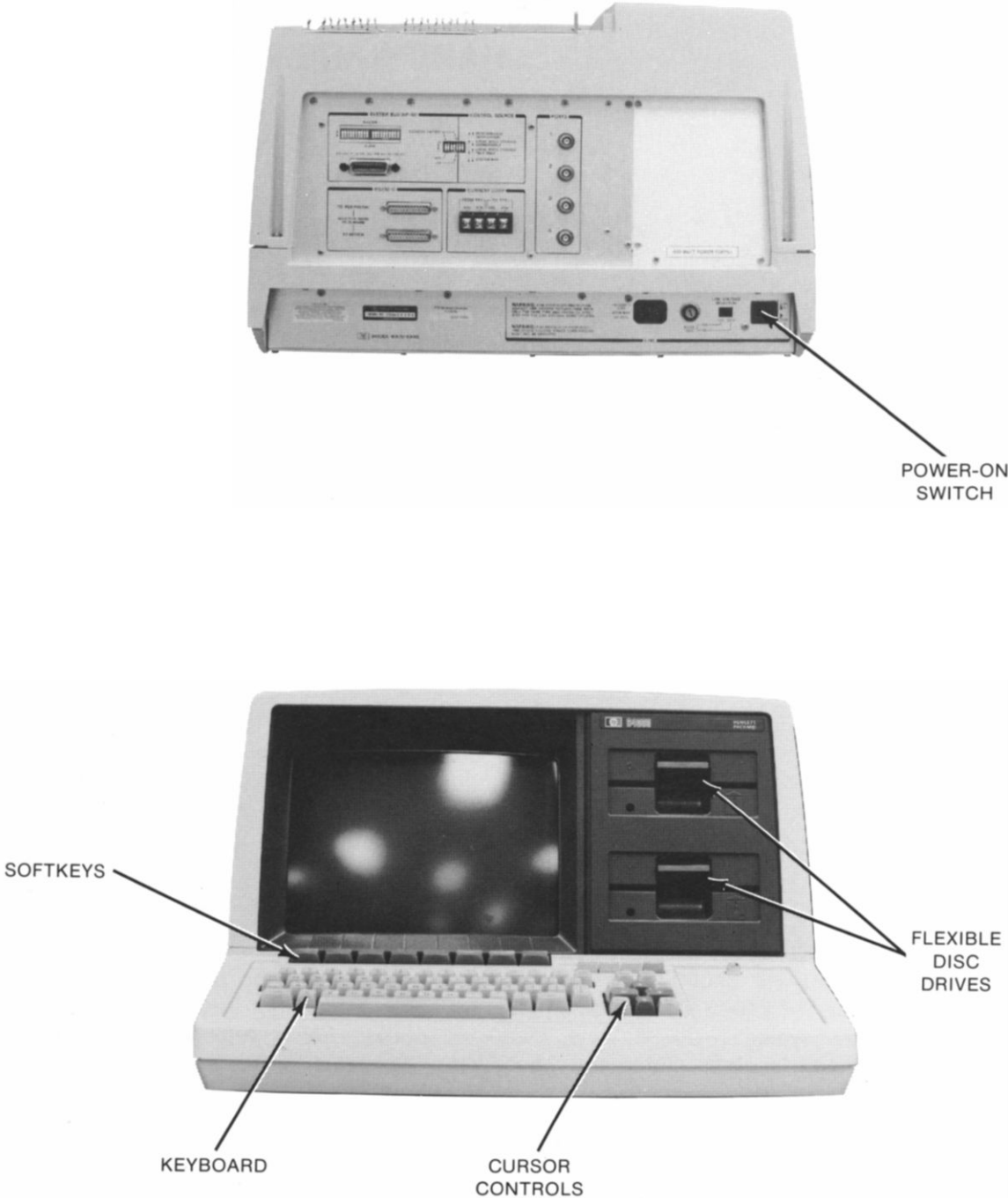


Figure 2-1. HP 64100 Key Features

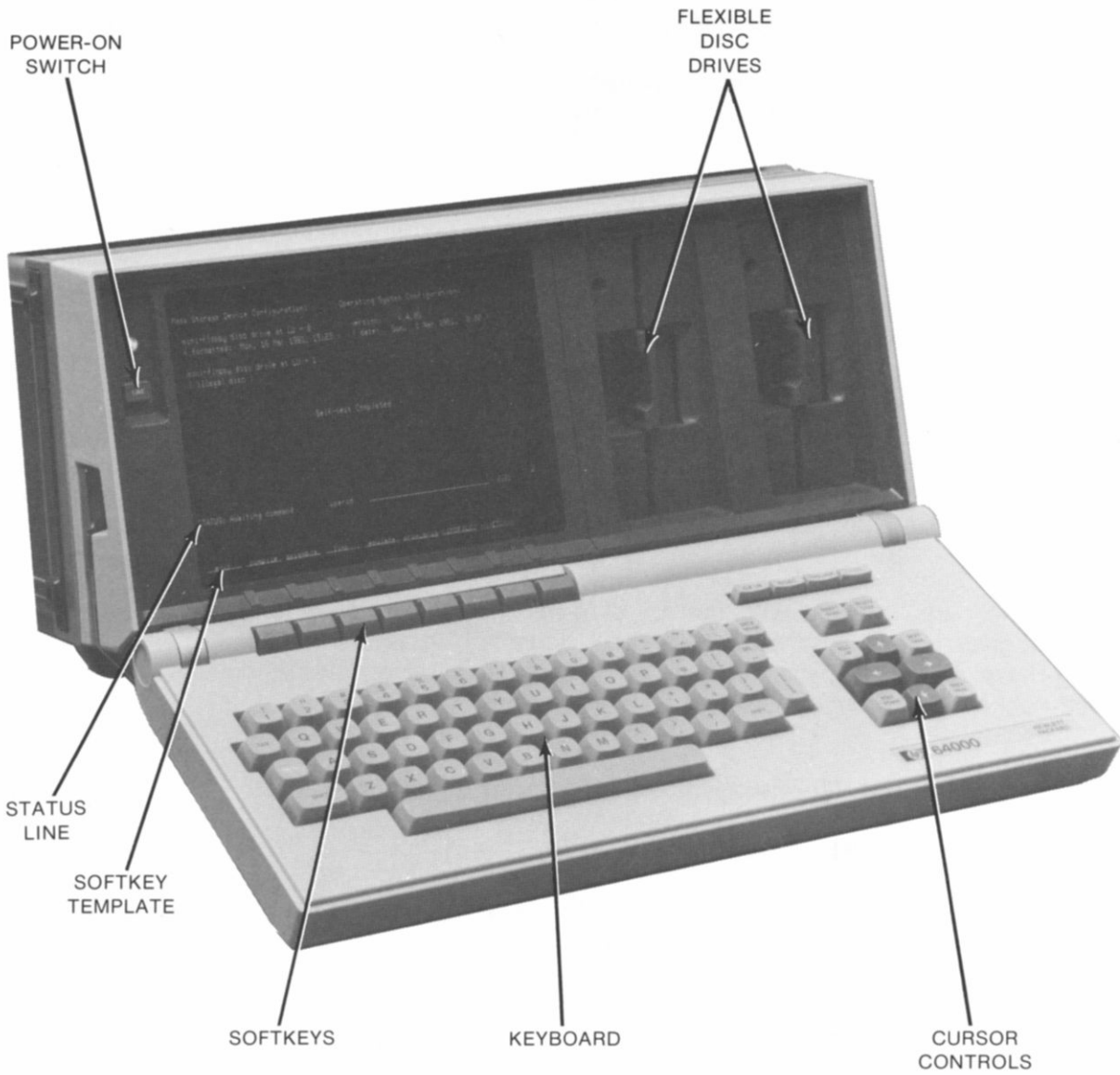


Figure 2-2. HP 64110 Key Features

If you have just received your 68000 Emulation system, check to see that you have all of the hardware, software, and manuals that are listed in this section. The hardware must be installed in the HP 64000 development station. You have a choice of loading the software to hard disc or using the software from a flexible disc. In either case, it is recommended that you generate back-up software as soon as your system is operational.

Figures 2-1 and 2-2 identify some key features of the HP 64100 and HP 64110 Stations, respectively. The identifying labels used in these figures are used throughout this manual. Note the location of the power switches. For more information on the hardware configuration, refer to the Installation and Configuration Manual.

If you are using the emulator for the first time, you should perform the checks listed in this section under the heading "Performing Operation Verification".

PRE-INSTALLATION INSPECTION

Unpack all of the emulation system circuit boards, cables, pod, and related equipment. Carefully inspect the equipment for damage that may have occurred during shipping. If any damage is found, please contact your nearest Hewlett-Packard Sales/Service Office as soon as possible.

Compare the parts received with the list of items that you ordered from the list of required equipment given below (illustrated in Chapter 1, figure 1-1) to assure that all of the items that you ordered have been shipped. If any equipment is missing, please contact your nearest Hewlett-Packard Sales/Service Office as soon as possible.

Required Equipment, Recommended Additional Equipment, and Additional Optional Equipment

The emulation system consists of four hardware modules plus the flexible disc emulation software package and technical manuals. The four hardware modules are:

- Emulation Subsystem
- Emulation Memory Subsystem
- Internal Analyzer
- Emulation/Memory/Analysis Bus Cable(s)

The emulation system hardware listed above consists of circuit boards which are installed in the development station card cage, the emulation probe (pod) which is connected by cabling to the development station, and bus cables to connect the boards. These parts, along with the software and technical manuals required, are shown in Chapter 1 (figure 1-1) and are listed below.

NOTE

You may already have some of the equipment listed if you have purchased other Hewlett-Packard equipment. For example, if you have previously purchased other emulation systems, you may already have an emulation memory subsystem, and possibly, the optional internal analysis board. In those cases, disregard the parts listed below that do not concern your situation, but do follow the installation instructions for the equipment that you have since the instructions are tailored to the 68000 emulation system.

Make sure you have the parts listed below that you have ordered and that no damage has occurred to any of the parts.

REQUIRED EQUIPMENT

Emulation Control Board (HP Part No. 64243-66501 or 64244-66501)

Emulation Probe (68000 Or 68008 Pod) with Microprocessor Connector (HP Model 64243 or 64244)

Emulation Memory Control Board (HP 64155A)

Emulation Memory Board (HP 64152/153/154A/B or HP 64161/162/163A: 1 required; 2-8 optional)

Emulation/Memory/Analysis Bus Cable (HP 64960A: 1 required for Memory Control Board)

Flexible Disc -68000 or 68008 Emulation System Software (HP Part No. 64243-12000 or 64244-12000)

Operating Manual with Service Appendix (HP Part No. 64243-90901)

RECOMMENDED ADDITIONAL EQUIPMENT

Wide Internal Analysis Board (64302A)

Emulation/Memory/Analysis Bus Cable (64960A: 2 required for Wide Internal Analysis Board)

Flexible Disc 68000/08/10 Assembler/Linker Software (64845AF)

ADDITIONAL OPTIONAL EQUIPMENT

Flexible Disc 68000/08/10 Pascal Compiler Software (64815AF)

Flexible Disc 68000/08/10 C Compiler Software (64819AF)

Software Performance Analyzer (64310A)

Logic State/Software Analyzer (64620) with Emulation Bus Preprocessor
(64304A)

68000 Dual In-Line Pin (DIP) Probe Cable (64243-61601)

68000 Pin Grid Array (PGA) Probe Cable (64243-61602)

INSTALLING EMULATION SYSTEM HARDWARE INTO AN HP 64100 LOGIC DEVELOPMENT STATION

WARNING

Any installation, servicing, adjustment, maintenance, or repair of this product must be performed only by qualified personnel. Make sure power is off prior to performing any of the installation instructions given below.

NOTE

The following installation steps assume the installation of a complete system (maximum memory and internal analysis). Disregard procedural steps for equipment you do not require or have.

Configuration of the Boards in the Station

While the emulation and analysis circuit boards may be installed in any card slot in the station chassis, mechanical considerations (i.e.; cabling) make the following card groupings most convenient:

For single module systems:

slot 9 Internal Analysis board
slot 8 Emulation Control board
slot 7 Memory Control board
slot 6 Memory board
slot 5 Memory board (if used)
slot 4 Memory board (if used)
slot 3 Memory board (if used)
slot 2 Memory board (if used)
slot 1 Memory board (if used)
slot 0 Memory board (if used)

For two emulation module systems:

slot 9 Internal Analysis board
slot 8 Emulation Control board
slot 7 Memory Control board
slot 6 Memory board
slot 5 Memory board (if used)
slot 4 Internal Analysis board
slot 3 Emulation Control board
slot 2 Memory Control board
slot 1 Memory board
slot 0 Memory board (if used)

When an emulator is used in a system with state or timing analyzers, either half of the above ordering may be used. As noted before, the boards may be installed in any order. It is important, however, that the boards be positioned so that the emulation control and intermodule bus cables do not cross.

Configuration of the Memory Boards

Each memory board has space for four rows of memory chips. The HP 6416XA memory boards may contain from 16K to 64K words (32K to 128K bytes) of random access memory (RAM). The HP 6415XA memory boards may contain from 4K to 16K words (8K to 32K bytes) of random access memory (RAM). The number of rows loaded on the board determines the amount of memory as follows;

NOTE

The HP 6415XA memory boards can no longer be purchased. However, they are mentioned here to cover the possibility that you may have purchased them previously to support earlier HP equipment purchases.

HP 64161A - 128K bytes - all four rows loaded
HP 64162A - 64K bytes - bottom two rows loaded
HP 64163A - 32K bytes - bottom row only loaded
HP 64152A - 32K bytes - all four rows loaded
HP 64153A - 16K bytes - bottom two rows loaded
HP 64154A - 8K bytes - bottom row only loaded

The number of bytes available on the board is also indicated on the ejector tab located on the top edge of the board.

6416XA Memory Boards

There are five 16-pin sockets located near the top edge (on the component side) of each memory board. Each half of the first four sockets is labeled, respectively, in 16K word ranges. Note that the numbers indicate words of memory (not bytes, as indicated on the ejector tab). The fifth socket contains jumpers which are used to prevent memory overlap by defining the memory partition of each memory board and to also allow mixing of Model HP 6416X and earlier Model HP 6415X series memory boards in the same emulation memory system. For further information regarding jumper configuration, refer to the Model HP 64161A/162A/163A Emulation Memory Service Manual. The factory-shipped position of the jumper in the fifth socket is as follows:

- HP 64161A - 8-pin jumper in the lower half of the fifth socket connecting R1 0-16K, R2 16-32K, R3 32-48K, and R4 48-64K.
- HP 64162A - Two 2-pin jumpers in the lower half of the fifth socket connecting R1 0-16K and R2 16-32K.
- HP 64163A - One 2-pin jumper in the lower half of the fifth socket connecting R1 0-16K.

If memory board types are not being mixed, the jumper in the fifth socket must be left in the factory-shipped position. If both HP 6416X and HP 6415X memory board types are to be used in your system, refer to the HP 64161A/162A/163A Emulation Memory Service Manual for configuration details.

6415XA Memory Boards

The HP 6415XA Emulation Memory boards are no longer available. If you have these boards available from a previous purchase, and wish to use them for supporting your 68000 emulation system you will need to refer to the HP 64152A/64153A/64154A Emulation Memory Service Manual for configuration details concerning these memory boards.

Installation Instructions

WARNING

Read the safety summary at the front of this manual before installation or removal of the Emulation Subsystem.

CAUTION

Power to the HP 64000 Development Station must be removed before installation or removal of option cards (emulation, etc.) to avoid damage to the option cards and the development station.

GENERAL. Installation of the circuit boards is accomplished by sliding each circuit board into the circuit board guide slots, with the component side of the board facing the front of the chassis. Align the connector at the bottom of the board with the motherboard connector at the bottom of the card cage, then apply a downward pressure until the board is seated in the motherboard connector. Be sure the ejector handles are in their full horizontal position when the board has reached its full downward travel.

Install the RFI Ground Bracket Assembly first. Because of cable restrictions, it is recommended that the internal analysis board be installed next, followed by the Emulation Control board, the Memory Controller, and then the memory boards.

To install the Emulation System and related equipment, proceed as follows:

- a. Turn OFF power to the HP 64100 Development Station. (See figure 2-1 for the location of the power switch.)

- b. Loosen the two hold-down screws on the top, rear of the card cage access cover, and remove the cover.
- c. Install the RFI Ground Bracket Assembly.

An RFI Ground Bracket must be installed in the HP 64100A development station before installing the Emulator Control Card and the Emulator Pod. The purpose of this ground bracket is to connect the emulator pod cable shield braid to earth ground. This effectively prevents the emission of radio frequency interference components from the pod cables.

The RFI ground bracket parts are included with the Emulator Pod. For convenience, a photograph depicting all the parts is reproduced here (see figure 2-3); Table 2-1 gives a listing of parts included in the ground bracket assembly (A4 for the 68000 Emulator Pod).

NOTE

If no ground bracket is installed in the development station, then the old U-shaped bracket on the rightmost side of the development station top cover (as viewed from the rear) must be removed. Loosen the two screws; remove the bracket; discard the bracket and the screws (they will not be needed for the RFI bracket).

Table 2-1. RFI Ground Bracket Assembly Parts List

REF DES	DESCRIPTION	PART NO.
A4	GROUND BRACKET ASSY	64100-62102
A4MP1	GROUND BAR CLAMP	1531-0273
A4MP2	SCREW 4-40 3/4"	2200-0151
A4MP3	SCREW 6-32 3/8"	2360-0117
A4MP4	SCREW 6-32 1"	2360-0129
A4MP5	NUT 6-32	2420-0001
A4MP6	WASHER	3050-0235
A4MP7	RFI GROUND BRACKET	64100-01205

Install the ground bracket on the HP 64100A Development Station, proceed as follows:

Place the RFI bracket on the top rear cover of the development station in the rightmost cable position (as viewed from the rear of the station) with the threaded stud on the bracket pointing upwards and the U-shaped portion of the bracket to the inside of the card cage. See figure 2-4 for details.

Thread two screws (A4MP3) into the RFI bracket from the outside of the development station rear panel sheet metal. Tighten the screws firmly.

To remove the bracket, reverse the installation procedure.

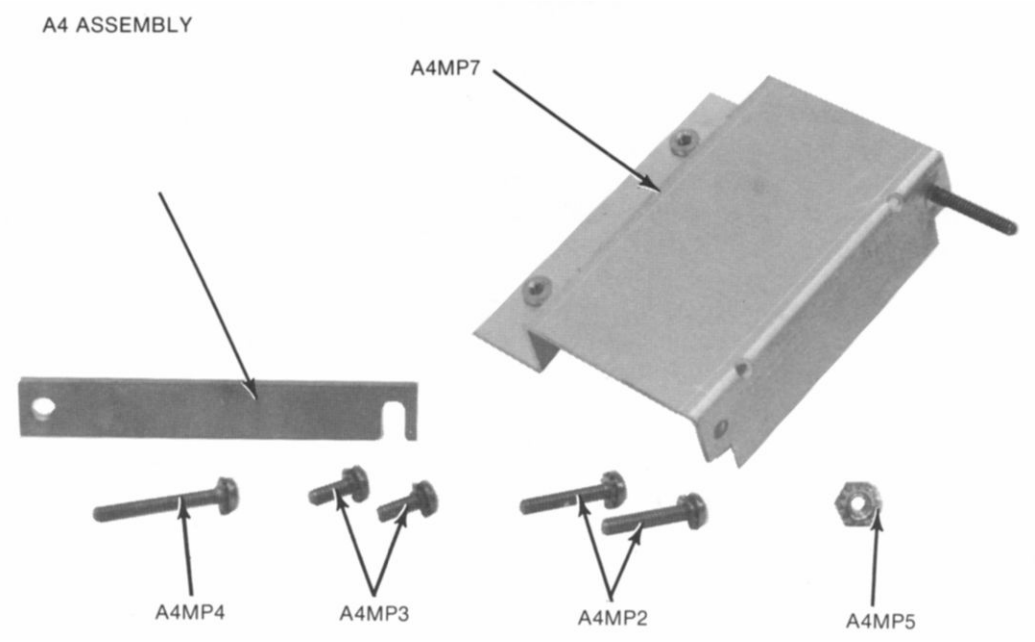


Figure G1-3. RFI Ground Bracket Assembly Parts
Figure 2-3. RFI Ground Bracket Assembly Parts

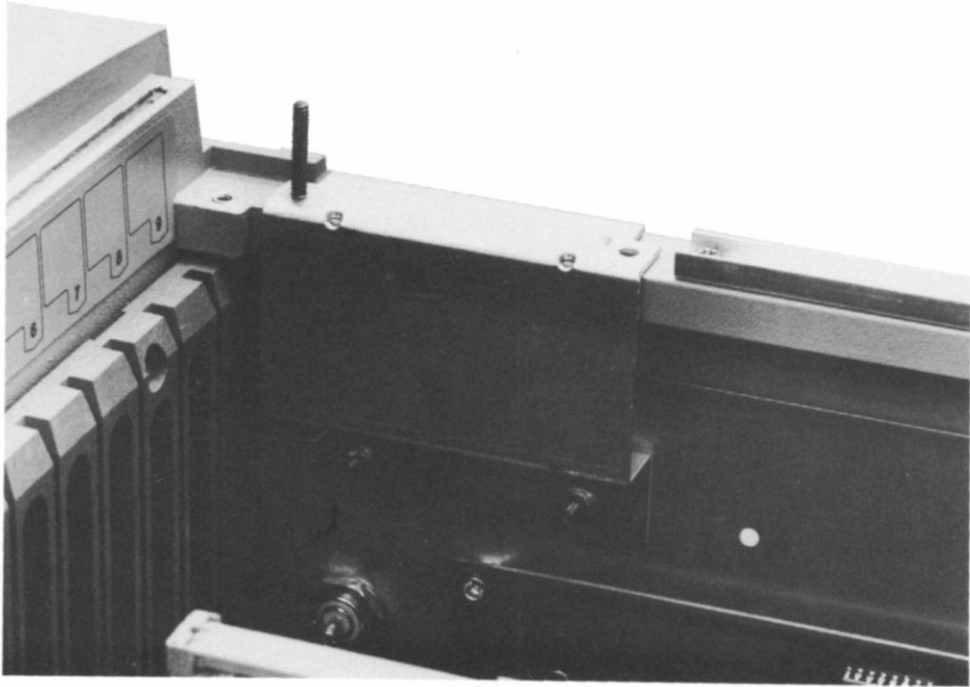


Figure 2-4. RFI Ground Bracket Installation (HP 64100A)

d. Install the Internal Analysis board

SINGLE MODULE SYSTEMS--- Install the Analysis board in the next rearmost slot (toward the rear of the HP 64000) from where the Emulation Control board will be installed. For example, if the Emulation Control board is to be installed in slot 8, the Analysis board should be installed in slot 9.

MULTIPLE MODULE SYSTEM--- Install the Internal Analysis boards between the locations where the Emulation Control boards will be installed.

e. Install the Memory Control board

SINGLE AND MULTIPLE MODULE SYSTEMS--- Install the Memory Control board in the next slot forward (one slot number less than) from the location where the Emulation Control board will be installed.

f. Install The Memory Boards

SINGLE AND MULTIPLE MODULE SYSTEMS--- Install the Memory board(s) in the slot(s) in front of the associated Memory Control board.

g. Install the Emulation Pod And Emulation Control board

Three multicolored ribbon cables are attached to the Emulation Probe (pod). These cables are used to connect the pod to the Emulation Control board (see figure 2-5). The cables have connectors which will only fit one way, described as follows: The brown and the red color-coded connectors on two of the cables plug in to the brown and red coded surface-mounted connectors located in the upper-right portion on the component side of the Emulation Control board. The yellow-coded connector on the other cable mates with the yellow-coded connector on the top edge of the Emulation Control board. Pin 1 on each cable connector is indicated by a triangle molded into the connector. Proper connection is also facilitated by keying of the connectors. Connect the pod to the Control Board by joining the connectors.

SINGLE MODULE SYSTEMS--- Install the Emulation Control board in slot 8 of the card cage to maximize the free cable length outside the development station chassis. Before fully seating the Emulation Control Board, connect the two Emulation Memory Control Split-bus cables from the Emulation Control Board to the Memory Control board located in the slot directly in front of the Emulation Control board.

Emulator/Analyzer 68000/68008
 Installing Your Emulation Hardware and Software

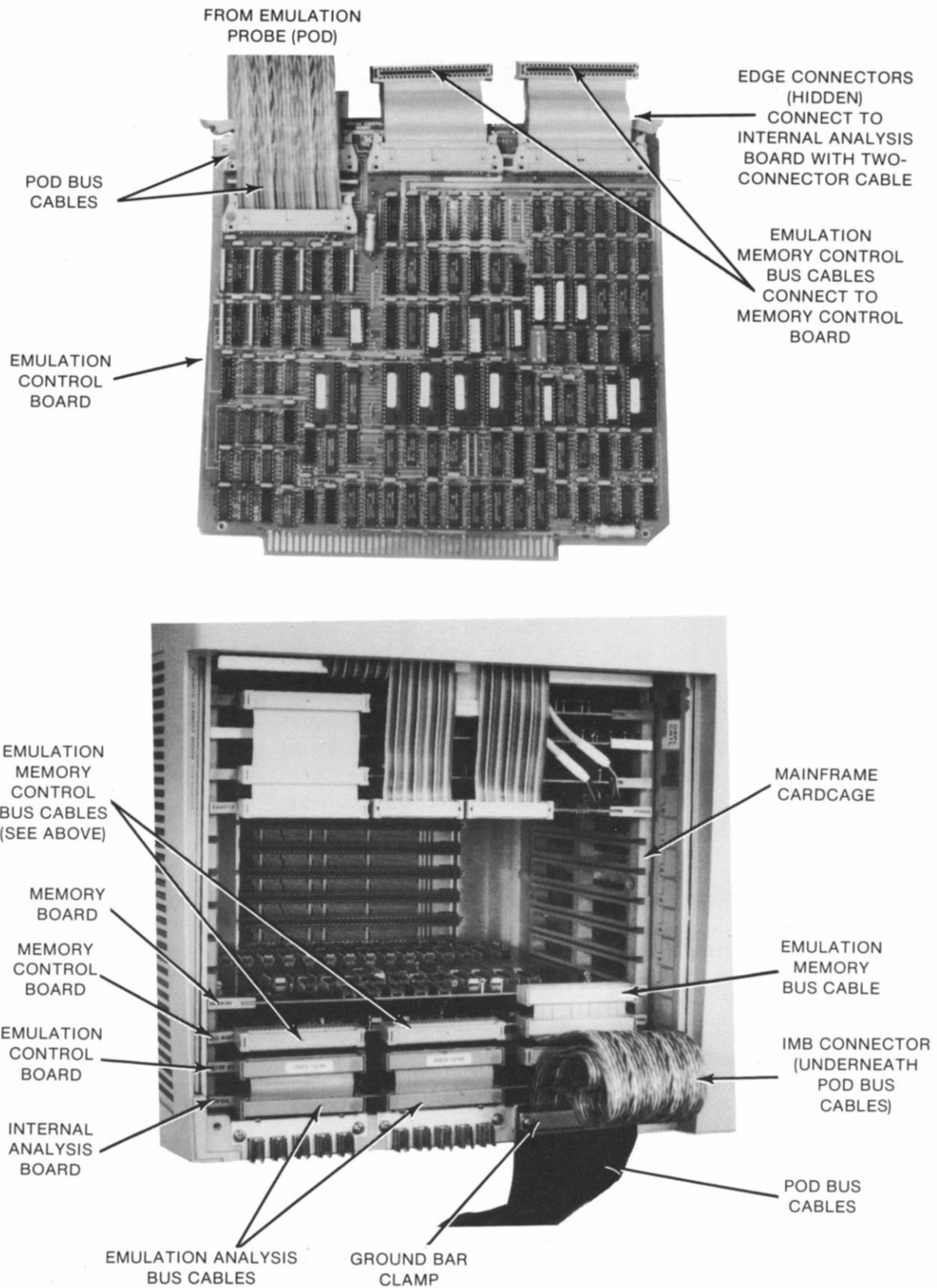


Figure 2-5. Emulation Control Board Connections

MULTIPLE MODULE SYSTEM--- Install the Emulation Control boards in slots 3 and 9 of the card cage so that problems connecting emulation and intermodule bus cables can be avoided. Before fully seating the Emulation Control Boards, connect the two Emulation Memory Control Split-bus cables from each Emulation Control Board to the Memory Control board located in the slot directly in front of each Emulation Control board.

- h. Install the ground bar clamp (see figure 2-6) across the emulator pod cable shielding to provide radio frequency interference (RFI) suppression, as follows:

Loosen both screws securing the ground bar clamp to the development station chassis. Swing the ground bar clamp back and insert the emulation pod cable between the two ground bar clamp screws. The shielded portion of the cable should be positioned between the two screws. There are notches in the cable to locate it in the correct position. You will need to fold the cable toward the front of the development station, then back to the rear, in order to position the cable properly.

Swing the ground bar clamp over the cable so that the slot in the ground bar clamp fits beneath the screw head. Tighten both screws securely, but take care not to damage the cable by over tightening the screws.

- i. Install the bus cables

After all the emulation, memory, and analysis circuit boards have been installed in the development station card cage, the emulation memory, emulation analysis bus cables, and the intermodule bus (IMB) cables must be installed across the top of the board set. See figure 2-7 for the cable configuration of a complete system.

The two cables (located at the top edge and on the right of the circuit board set as you face the front of the development station) connecting the Internal Analysis board and the Emulation Control board are the Emulation Analysis Bus cables. The connectors are keyed to facilitate correct installation, and each connector has a triangle indicator molded into the connector to indicate the location of the pin 1 side and end in the connector. When properly installed, the red stripe marker on the bus cable is on the left-hand side of the cable when viewed from the front and above the card cage.

The Emulation Memory Bus cable is on the left-hand side of the board set, as you face the front of the development station. The Memory Control board is joined to the Memory boards by this cable. The connectors are color coded and keyed to facilitate proper installation, with the color coding placed to the left end of the connector. Each connector has a triangle indicator molded into the connector to indicate the location of the pin 1 side and end in the connector. When properly installed, the red stripe marker will be on the left of the ribbon cable as you look down on the card cage from the front of the development station.

The intermodule bus (IMB) is not shown in figure 2-5. It is used when logic analyzers, other than the internal analyzer, are used with the emulation system. The IMB, when used with the emulation system, requires the internal analysis board to be installed in the development station to interface with other analysis boards. The analysis boards contain a 20-pin IMB connector on the top, left side of the board which is used to connect the boards for intermodule measurements. The intermodule bus between boards consists of a 20-conductor ribbon cable that is installed on the IMB connector of the appropriate board(s).

- j. Replace the card cage access cover and secure in place with the two attached hold-down screws.

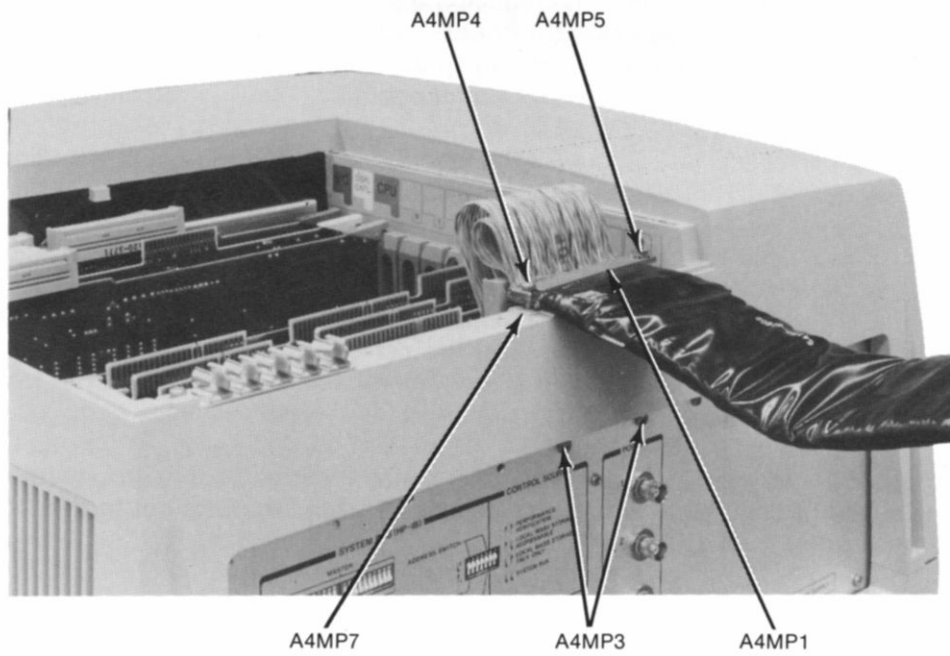


Figure 2-6. Ground Bar Clamp Installation (HP 64100A)

INSTALLING EMULATION SYSTEM HARDWARE INTO AN HP 64110 LOGIC DEVELOPMENT STATION

WARNING

Any installation, servicing, adjustment, maintenance, or repair of this product must be performed only by qualified personnel. Make sure power is off prior to performing any of the installation instructions given below.

NOTE

The following installation steps assume the installation of a complete system (maximum memory and internal analysis). Disregard procedure steps for equipment you do not require or have.

Configuration of the Boards in the Station

While the emulation and analysis circuit boards may be installed in any card slot in the station chassis, mechanical considerations (i.e.; cabling) make the following card groupings most convenient:

- slot 0 Internal Analysis board
- slot 1 Emulation Control board
- slot 2 Memory Control board
- slot 3 Memory board
- slot 4 Memory board (if used)

Configuration of the Memory Boards

Configuration of the memory boards is described previously in the installation instructions for the HP 64100 station.

Installation Instructions

GENERAL. Installation of the circuit boards is accomplished by sliding each circuit board into the circuit board guide slots, with the component side of the board facing the top of the chassis. Align the connector at the bottom of the board with the motherboard connector in the card cage, then apply an inward pressure until the board is seated in the motherboard connector. Be sure the ejector handles on the top of the boards are parallel to the top of the board when the board has reached its full inward travel.

To install the Emulation System and related equipment, proceed as follows:

WARNING

Read the safety summary at the front of this manual before installation or removal of the Emulation Subsystem.

- a. Turn OFF power to the HP 64110 Development Station. (See figure 2-2 for the location of the power switch.)

CAUTION

Power to the HP 64110 Development Station must be removed before installation or removal of option cards (emulation, etc.) to avoid damage to the option cards and the development station.

- b. Gain access to the card cage as follows:

- Loosen the two hold-down screws (see figure 2-7) on the rear of the card cage access cover, and remove the cover.
- Remove the two rear plate screws (figure 2-7), and remove the rear plate.

- c. Install the Internal Analysis board

Install the Analysis board in the next higher slot (toward the top of the HP 64110) above the slot where the Emulation Control board will be installed.

- d. Install the Memory Control board

Install the Memory Control board in the next vacant slot beneath the slot where the Emulation Control board will be located.

- e. Install the Memory board(s)

Install the Memory board(s) in the slot(s) beneath the associated Memory Control board.

Memory board configuration is described previously in the paragraph entitled "Configuration of the Memory Boards" for the HP 64100 Development Station installation.

- f. Install the Emulation Pod And Emulation Control board

Three multicolored ribbon cables are attached to the Emulation Probe (pod). These cables are used to connect the pod to the Emulation Control board (see figure 2-5). The cables have connectors which will only fit one way, described as follows. The connector on one of the

cables connects to the surface-mounted connector located on the upper-left portion of the component side of the Emulation Control board. The connector on the other cable mates with the connector on the top edge of the Emulation Control board. Pin 1 on the cable connectors is indicated by a triangle molded into each connector. Proper connection is facilitated by the color coding and keying of the connectors. Connect the pod to the control board by joining the connectors.

Install the Emulation Control board into the slot guides between the Internal Analysis board and the Memory Control board to maximize the free cable length outside the development station chassis. Before fully seating the Emulation Control Board, connect the two Emulation Memory Control Bus cables from the Emulation Control Board to the Memory Control board located in the slot directly below the Emulation Control Board.

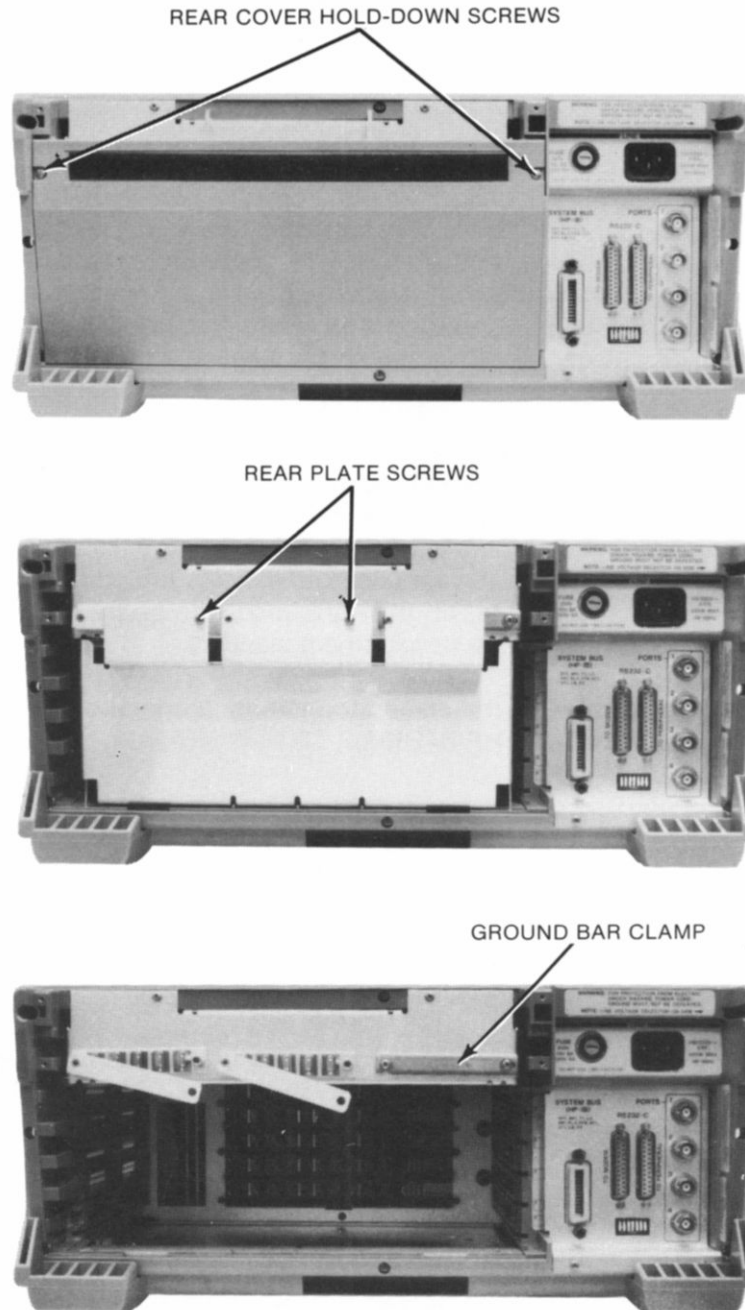


Figure 2-7. HP 64110 Installation Details

g. Install the ground bar clamp (see figure 2-8) across the emulator pod cable shielding to provide radio frequency interference (RFI) suppression, as follows:

Loosen both screws securing the ground bar clamp to the development station chassis. Swing the ground bar clamp back and insert the emulation pod cable between the two ground bar clamp screws. The shielded portion of the cable should be positioned between the two screws. There are notches in the cable to locate it in the correct position. You will need to fold the cable toward the bottom of the development station, then back toward the top, in order to position the cable properly.

Swing the ground bar clamp over the cable so that the slot in the ground bar clamp fits beneath the screw head. Tighten both screws securely, but take care not to damage the cable by over tightening the screws.

h. Install the bus cables

After all the emulation, memory, and analysis circuit boards have been installed in the development station card cage, the emulation memory, emulation analysis bus cables, and the intermodule bus (IMB) cables must be installed across the exposed edge of the board set. See figure 2-9 for the cable configuration for a complete system.

The three multicolored ribbon cables at the top right of the circuit board set (as viewed from the rear) that connect the Internal Analysis board and the Emulation Control Board are the Emulation Analysis Bus cables. These connectors are keyed and color coded. Each connector also has a triangle indicator molded into the connector to indicate the pin 1 location.

The Emulation Memory Split-bus is on the left-hand side of the board set as you face the rear of the development station. The Memory Control board is joined to the Memory boards by the two short Split-bus cables which attach to the center and left-hand edge connectors (as seen from the rear). The connectors are keyed to facilitate proper installation, and each connector has a triangle indicator molded into the connector to indicate the pin 1 location. When properly installed, the red stripe marker will be on the right-hand side of the ribbon cables as you look into the card cage from the rear of the development station.

The analysis system is attached to the center and left-hand edge connectors as seen from the rear) with the Emulation Bus Cable HP 64960A.

The intermodule bus (IMB) is not shown in figure 2-9. It is used when logic analyzers, other than the internal analyzer, are used with the emulation system. The IMB, when used with the emulation system, requires the internal analysis board to be installed in the development station to interface with other analysis boards. The analysis boards contain a 20-pin IMB connector on the top, left side of the board which is used to connect the boards for intermodule measurements. The intermodule bus between boards consists of a 20-conductor ribbon cable that is installed on the IMB connector of the appropriate board(s).

i. Reinstall the rear plate and rear access cover by reversing the procedures given in step b, above.

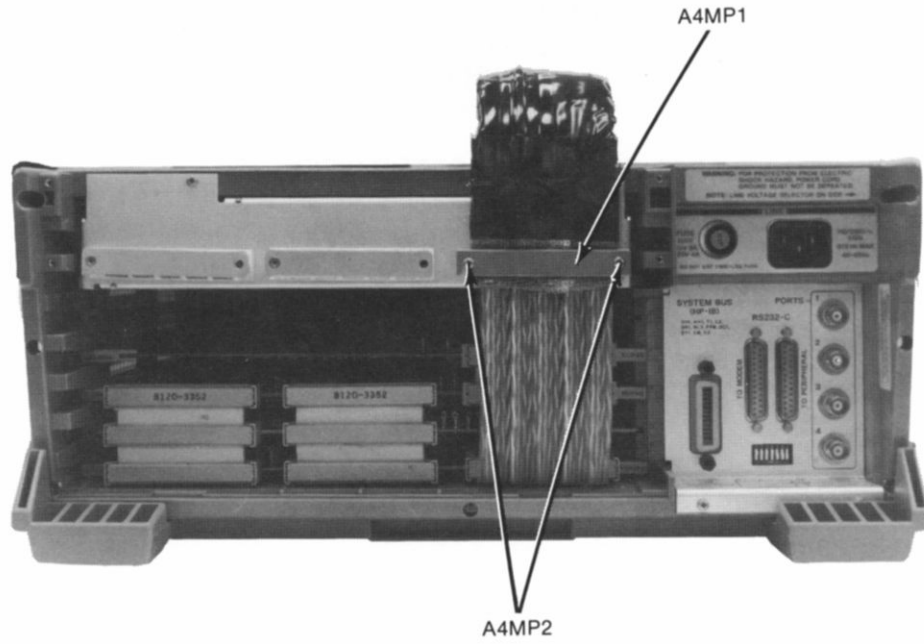


Figure 2-8. Ground Bar Clamp Installation (HP 64110A)

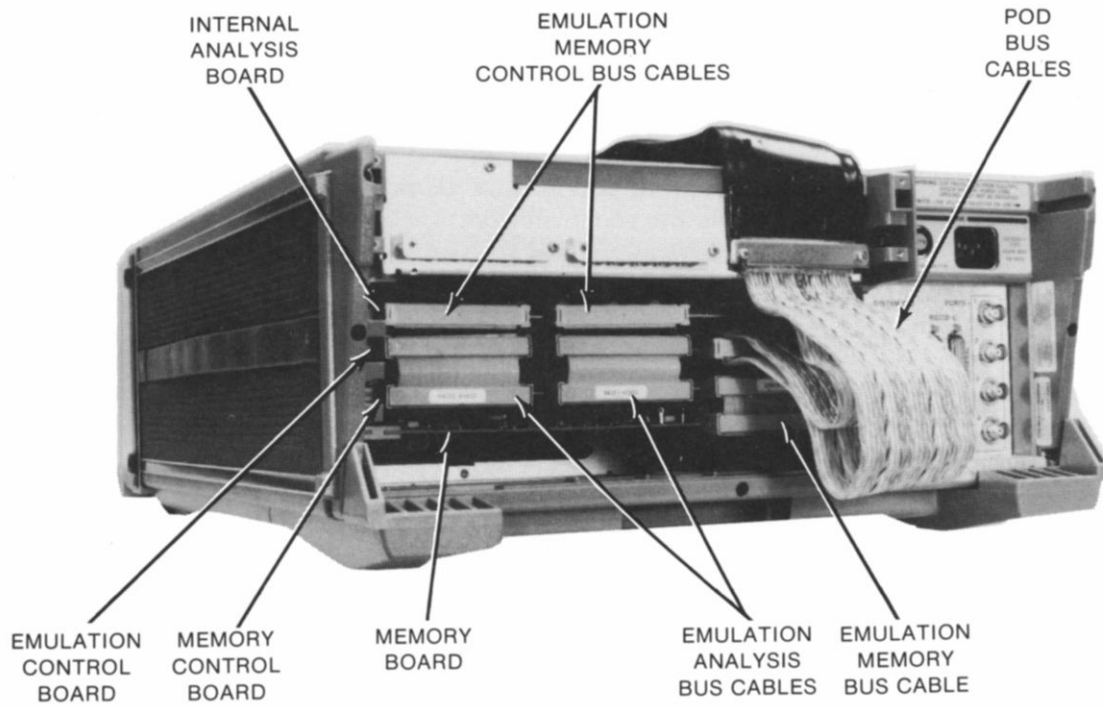


Figure 2-9. HP 64110 Memory, Emulation, and Intermodule Bus Cabling

INSTALLING THE EMULATION PROBE INTO THE TARGET SYSTEM



PROTECT AGAINST STATIC DISCHARGE

The emulation probe contains devices that are susceptible to damage by static discharge. Therefore, precautionary measures should be taken before handling the microprocessor connector attached to the end of the cable from the emulation probe to avoid damaging the internal components of the probe by static electricity.



Do not install the emulation probe into the processor socket with power applied to the target system. The pod may be damaged if power is not removed before installation.

The emulation probe is provided with a pin protector that prevents damage to the probe when connecting and removing the probe from the microprocessor socket. DO NOT use the probe without a pin protector installed. If the emulation probe is being installed on a densely loaded circuit board there may not be enough room to accommodate the plastic shoulders of the probe socket. If this occurs, another pin protector may be stacked onto the existing pin protector. The wire extending from the emulation pod may be connected to the target system signal ground.

Carefully remove the target processor from its socket, and place the processor into a protected area. Then install the emulation probe into the vacant socket.

Target System Microprocessor Connector Compatibility

To properly connect and operate the emulator, your target system microprocessor socket must be compatible with the microprocessor connector on the emulation system.

Installing Into a DIP Socket (See Figure 2-10)

To connect the microprocessor connector to a target system containing a dual in-line pin (DIP) socket for the host processor, proceed as follows: Remove the 68000 microprocessor from the target system microprocessor socket. Store the microprocessor in a protected environment. Note the location of pin 1 on both the microprocessor connector and the target system socket. Place the microprocessor connector attached to the end of the cable from the Emulation Probe into the target system microprocessor DIP socket.

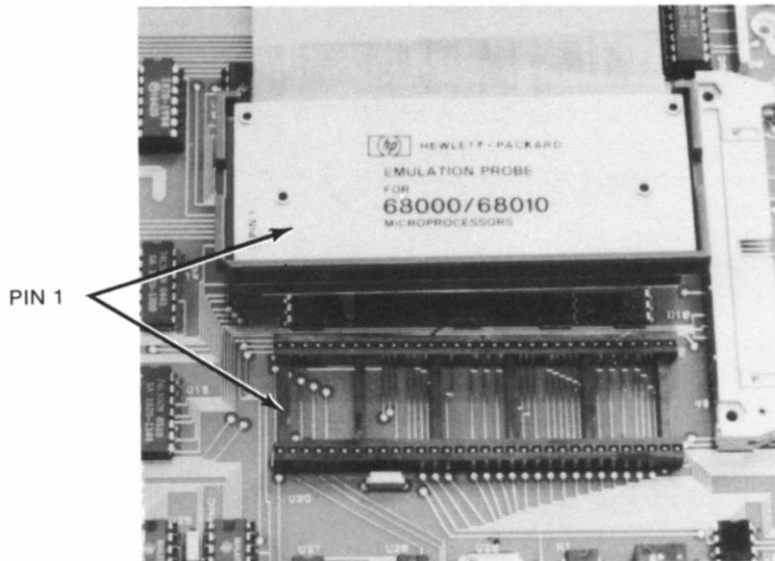


Figure 2-10. Installing the Emulation Probe into a DIP Socket

Installing into a PGA Socket (See Figure 2-11)

To connect the microprocessor connector to a target system containing a Pin Grid Array (PGA) socket, proceed as follows: Remove the 68000 microprocessor from the target system microprocessor PGA socket. Store the microprocessor in a protected environment. Note the location of pin 1 on both the microprocessor connector and the target system socket. Place the microprocessor connector attached to the end of the cable from the Emulation Probe into the target system microprocessor socket.

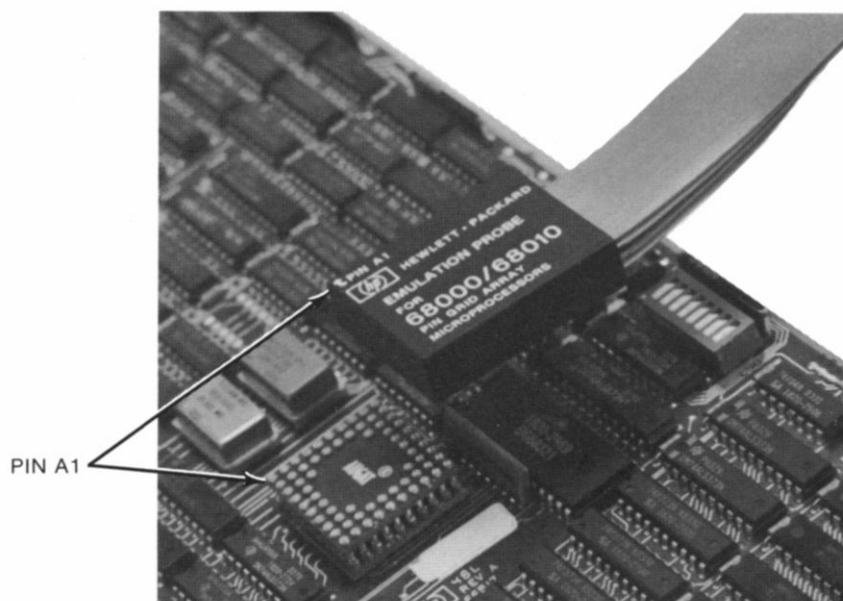


Figure 2-11. Installing the Emulation Probe into a PGA Socket

CAUTION

PROTECT PGA PINS FROM DAMAGE

To avoid damaging the PGA (Pin Grid Array) probe connector pins, an insertion/extraction tool (such as Augat P/N TX 8136-12) is recommended for removing the PGA probe connector.

TURNING ON THE DEVELOPMENT STATION

The power switches for the HP 64100 and HP 64110 development stations are identified in figure 2-1 and figure 2-2, respectively.

Turn the station power on.

You will hear several tones which indicate that power has been applied. Two additional tones indicate that a self-test has been performed. The initial display on the cathode ray tube (CRT) will be similar to the display in figure 2-12. The I/O bus configuration for your HP 64000 cluster system should list every station, hard disc and printer on the system bus.

I/O BUS CONFIGURATION

```
ADRS  DEVICE
  0  13037 DISC CONTROLLER
      UNIT  0  7906 DISC MEMORY  LU=0
  1  PRINTER (t2)
  3  64000
  4  THIS 64000
  5  64000
```

```
STATUS: Awaiting command      userid _____ 0:00
```

```
edit  compile  assemble  link  emulate  _____  run  ---ETC---
```

Figure 2-12. I/O Bus Configuration Display

LOADING EMULATION SYSTEM SOFTWARE

Backing Up Your Software

As soon as you receive your emulation system software you should make a back-up copy for archival purposes. Keep the flexible discs which you received from Hewlett-Packard as an archive copy.

To duplicate your emulation system software:

1. PRESS *-BACKUP-- floppy utilities* **(RETURN)**.
2. Insert a new flexible disc into disc drive 1.
3. PRESS *format 1* **(RETURN)**.
4. When the disc format is complete insert Emulation System Software disc 1 into disc drive 0.
5. PRESS *duplicate 0* **(RETURN)**.
6. Repeat the above steps for Emulation System Software disc 2.
7. PRESS *end* **(RETURN)**.

Store the master copy of the Emulation System Software that you received from Hewlett-Packard in a safe place.

Loading Software in Clustered Stations Configuration

Your 68000 and/or 68008 Emulation System includes two flexible discs containing all of the emulation software. Follow the instructions given below to load this software onto a hard disc in a cluster configuration.

1. Insert disc 1 of the Emulation System Software into disc drive 0.
2. PRESS *-BACKUP-- floppy sys_gen* **(RETURN)**.
3. PRESS *copy all from local 0 to bus_disc 0* **(RETURN)**.

NOTE

On the left side of the display you see module name EMULATION_S68000. When the copy is complete, EMULATION_S68000 will be added to the list on the right side of the display. Press the **(NEXT PAGE)** key on the cursor control panel until you see EMULATION_S68000 on the "System modules on bus disc" list. For the 68008 processor, substitute "S68008" in place of "S68000" in all the software backup and loading procedures in this section.

4. Remove disc 1 of the Emulation System Software from disc drive 0.
5. Insert disc 2 of the Emulation System Software into disc drive 0.
6. Press **RETURN**.
7. PRESS *end* **RETURN** to exit from sys_gen.
8. Remove disc 2 of the Emulation System Software from disc drive 0 and store both discs for future use.

Configuring A Flexible Disc For Stand-Alone Operation

Follow these guidelines to configure flexible discs for stand-alone operation (operation from 5 1/4 inch flexible discs).

- Format a new flexible disc.
- Define the configurations of software you will need on each flexible disc.
- Copy (use floppy sys_gen) the operating system modules and emulation modules you need onto the newly formatted disc(s).

FORMAT A NEW FLEXIBLE DISC. You will need to format three discs for the suggested configuration given below. To format a disc, proceed as follows:

1. Insert Operating System disc 2 into disc drive 0.
2. Insert a new disc into disc drive 1.
3. PRESS *floppy utilities* **RETURN**.
4. PRESS *format 1* **RETURN**.
5. When the formatting is complete the STATUS line will read:

Disc 1 formatting completed
6. To exit floppy utilities, press *end* **RETURN**.

DEFINE DISC SOFTWARE CONFIGURATIONS. The following configuration is a suggestion for configuring your discs to assemble, link, edit your files, and run emulation and analysis. Disc 1 is required in all situations. Install it in disc drive 0. Use disc 2 with disc 1 for assembling, linking, or editing. Use disc 3 with disc 1 for emulation and analysis.

Disc 1	Disc 2	Disc 3
FLOPPY_OP_SYS	ASSEMBLER	SYS_GEN
FLOPPY_UTILITIES	LINKER	MEAS_SYS
MONITOR_S68000	COPY	EMULATION_S68000
YOUR SOURCE FILE	DIRECTORY	ANLY_302_S68000
YOUR ABSOLUTE FILE	EDITOR	
	ASMB_LINK_68000	

COPY THE MODULES ONTO NEW DISCS. To copy the Operating System and Emulation Software modules onto the suggested new discs, proceed as follows:

NOTE

The following procedure is based on the 2.01 version of the Operating System Software. If you have any other version, the software modules referred to in the procedures may not be on the disc specified. To find out where each of the modules reside, perform steps 1 through 5. Note the modules on disc 2 for future reference during the procedure. Remove Operating System disc 2 from disc drive 0 and insert Operating System disc 1 in its place. Press **(RETURN)** and note the modules on this disc. Repeat the procedure for Operating System disc 3. Now that you know where all of the modules reside, substitute the disc number in the remaining portion of the procedure for the disc number that contains the module that you want to transfer.

1. Insert Operating System disc 2 into disc drive 0 to boot the system up. Label one of the new formatted discs DISC #1, and insert it into disc drive 1.
2. Remove Operating System disc 2 from disc drive 0 and insert Operating System disc 3 in its place. This disc contains the SYS_GEN software which allows you to show and copy modules.
3. PRESS *floppy sys_gen* **(RETURN)**.
4. Remove Operating System disc 3 from disc drive 0 and insert Operating System disc 2 in its place.
5. PRESS *show local 0* **(RETURN)**. The display will show the modules on Operating System disc 2. You will copy some of these to Disc #1 in disc drive 1.
6. PRESS *copy* (Number of FLOPPY_OP_SYS) *from local 0 to local 1* **(RETURN)**.

7. PRESS *copy* (Number of FLOPPY_UTILITIES) *from local 0 to local 1* **(RETURN)**.
8. Remove Operating System disc 2 from disc drive 0 and insert Emulation System Software disc 2.
9. PRESS *show local 0* **(RETURN)**.
10. PRESS *copy* (Number of MONITOR_S68000) *from local 0 to local 1* **(RETURN)**.
11. Remove DISC#1 from disc drive 1. Label one of the new formatted discs DISC#2, and insert it into disc drive 1.
12. Remove Operating System disc 2 from disc drive 0 and insert the 68000 Assembler/Linker disc in its place.
13. PRESS *copy all from local 0 to local 1* **(RETURN)**.
14. Remove 68000 Assembler/Linker disc from disc drive 0 and insert Operating System disc 2 in its place.
15. PRESS *show local 0* **(RETURN)**. The display will show the modules on Operating System disc 2.
16. PRESS *copy* (Number of ASSEMBLER) *from local 0 to local 1* **(RETURN)**.
17. PRESS *copy* (Number of LINKER) *from local 0 to local 1* **(RETURN)**.
18. PRESS *copy* (Number of COPY) *from local 0 to local 1* **(RETURN)**.
19. PRESS *copy* (Number of DIRECTORY) *from local 0 to local 1* **(RETURN)**.

Emulator/Analyzer 68000/68008
Installing Your Emulation Hardware and Software

20. Remove Operating System disc 2 from disc drive 0 and insert Operating System disc 1 in its place.
21. PRESS *show local* 0 **(RETURN)**. The display will show the modules on Operating System disc 1.
22. PRESS *copy* (Number of EDITOR) *from local 0 to local 1*.
23. Remove DISC#2 from disc drive 1. Label the last of the new formatted discs DISC#3, and insert it into disc drive 1.
24. Remove Operating System disc 1 from disc drive 0 and insert Operating System disc 3 in its place.
25. PRESS *show local* 0 **(RETURN)**. The display will show the modules on Operating System disc 3.
26. PRESS *copy* (Number of MEAS_SYS) *from local 0 to local 1* **(RETURN)**.
27. PRESS *copy* (Number of SYS_GEN) *from local 0 to local 1* **(RETURN)**.
28. Remove Operating System disc 3 from disc drive 0 and insert Emulation System Software disc 1 in its place.
29. PRESS *show local* 0 **(RETURN)**. The display will show the modules on Operating System disc 1.
30. PRESS *copy* (Number of EMULATION_S68000) *from local 0 to local 1* **(RETURN)**.
31. Remove Emulation System Software disc 1 from disc drive 0 and insert Emulation System Software disc 2 in its place.
32. PRESS *show local* 0 **(RETURN)**. The display will show the modules on Emulation System Software disc 2.
33. PRESS *copy* (Number of ANLY_302_S68000) *from local 0 to local 1* **(RETURN)**.
34. Remove DISC#3 from disc drive 1. Cover the notch on the side of discs #2 and #3 that you have just loaded to protect your new stand-alone configuration operating system discs. Leave disc #1 unprotected so that you can write your program to it.

REMOVING EMULATION SOFTWARE FROM THE OPERATING SYSTEM

System software, such as Emulation software, cannot be purged from the system and it cannot be removed file-by-file. System software must be "removed" from the Operating System via floppy `sys_gen`. The following instructions will illustrate this process:

1. PRESS *-BACKUP-- floppy sys_gen* **(RETURN)**.
2. PRESS *show bus_disc 0* **(RETURN)**.

NOTE

PRESS **(NEXT PAGE)** key on cursor control panel until you locate the module you want to remove. You can remove or copy modules by their list number or by the module name.

3. PRESS *remove <MODULE> or <NUMBER> from bus_disc <DISC #>* **(RETURN)**.
4. PRESS *end* **(RETURN)**.

PERFORMING OPERATION VERIFICATION

To perform operation verification:

1. Make sure the board configuration is correct.
2. Unplug the emulator from your target system.
3. Remove the conductive pin protector from the emulator plug-in.
4. PRESS *---ETC---* until `opt_test` appears as a softkey option.
5. PRESS *opt_test* **(RETURN)**.
6. PRESS *<SLOT #>* (of your emulator) **(RETURN)**. The display shown in figure 2-13 will appear on the screen.
7. PRESS *cycle*. The inverse video bar will cycle down through each of the lines denoting the different tests being performed.
8. If # Fail column indicates that any of the tests have failed, refer to Section G2 of this manual.
9. To exit the `opt_test` routine press *end*.
10. To return to the system monitor press *end*.

68000 Performance Verification

```
MC68000L12 Emulator   in slot #8           Wide Emul. Analysis in slot #9
Wide Memory Ctl. in slot #7

      Test                                     #Fail #Test
1 - 3 Pod Register Tests (Static)           0      0

4 - 12 Pod Functionality Tests              0      0

13 - 16 Analysis Tests                      0      0

STATUS: Awaiting option_test command _____ 14:18

end      cycle      next_test      disp_test      _____      print
```

Figure 2-13. Option Test Display.

OPERATING THE EMULATION SYSTEM IN STAND-ALONE CONFIGURATION

Remember, when you are operating in stand-alone mode:

- a. If your operating system in disc drive 0 is write-protected, the files you create, edit, assemble, and emulate must be assigned to disc drive 1 by adding :1 to the file name--i.e., FILENAME:1.
- b. When you are writing to your files you will see an error message on the status line "Disc 0 write protected". You can ignore this message as long as you are correctly assigning your files.

REMOVING THE EMULATOR CONTROL BOARD

To remove the 68000 Emulator Control board, use the following procedures:

WARNING

Read the safety summary at the front of this manual before installation or removal of the Emulation Subsystem.

- a. Turn OFF power to the HP 64000 Development Station.

CAUTION

Power to the HP 64000 Development Station must be removed before installation or removal of option cards (emulation, etc.) to avoid damage to the option cards and the development station.

- b. Remove the card cage access cover. (Refer to installation procedures given earlier in this section for instructions to remove this access cover.)
- c. Remove the emulation bus cables.
- d. Remove the RFI ground bar clamp.
- e. Pull up on the board extractor levers and pull the card clear of the development station card cage.

REPACKAGING FOR SHIPMENT

Original Packaging Materials

Containers and packing materials identical to those used in factory packaging are available through Hewlett-Packard offices.

Other Packaging Materials

The following general instructions should be used for re-packing with commercially available materials:

- a. Wrap the emulator component in heavy paper or plastic.
- b. Use a strong shipping container. A double-wall carton made of 350-pound test material is adequate.
- c. Use a layer of shock-absorbing material 70 to 100 mm (3 to 4 inches) thick around all sides of the components to provide firm cushioning and prevent movement inside the container.
- d. Seal shipping container securely.
- e. Mark shipping container FRAGILE to ensure careful handling.
- f. In any correspondence, refer to instrument by model number and full serial number.

Chapter 3

GETTING STARTED

OVERVIEW

In this chapter you will:

- Learn to assign your own userid.
- Learn the difference between single-module and multi-module systems.
- Become familiar with the monitor level and emulation level softkeys.
- Create a sample program consisting of a short loop that decrements a register.
- Learn to set up command files for assembling, linking, and configuring program modules.
- Assemble your program module and the emulation monitor program module.
- Link your program module with the emulation monitor program module.
- Learn how to gain access to the emulation system from the monitor level softkeys.
- Answer the emulation configuration questions and map memory.
- Run an emulation session (trace a program sequence and examine registers).

GENERAL

In this chapter, whenever 680XX is mentioned (even as part of a filename) both the 68000 and the 68008 are assumed unless there are differences between the two processors. In that case, the differences will be noted. When the name 680XX appears in any program listings, just insert the appropriate name of your processor.

INTRODUCTION

The purpose of this chapter is to give you an operational overview of the emulation process. This will be done by leading you step-by-step through an emulation session. None of the emulation features will be explained in depth in this chapter; your goal should be to familiarize yourself with the emulation procedure and the softkey choices available. You should, however, read the entire chapter and go through all of the exercises in the order presented so that you will gain an understanding of the fundamentals of the emulation system operation.

There are three steps to the emulation process: preparing the software, preparing the emulator, and using the emulator.

If you are ready to use the emulator you should have written one or more software modules. These modules (and the emulation monitor program) must be assembled and linked together before they can be run by the emulator. A short example program is provided in this chapter.

Next you need to configure the emulator for your application. You do this by answering several configuration questions. These questions define which resources will be provided by your target system and which resources will be provided by the emulation system. For example, the clock can be provided by either your target system or by the emulation system. Finally, you use the emulator by loading the absolute code provided by the linker (the linked programs) and by using the emulation features to observe your program as it is running, or by examining the registers and memory locations. In this chapter we will load a sample program, run it, perform a trace, and display the processor registers.

UNDERSTANDING THE EXAMPLES

The examples provided throughout this manual use the following structure:

PRESS *edit* MODULE **RETURN**.

PRESS or press-- means you should enter a command by selecting the softkeys and/or typing in any file names or other variables which are not provided in the softkey selections.

edit -- softkeys will appear in italics. Usually you will not be prompted to use the ---ETC--- softkey to search for the appropriate softkey template.

MODULE -- this is the name of a file which you must type in. Softkeys are not provided for this type of selection since it is variable. However, a softkey prompt such as, <FILE> will appear as a softkey selection.

RETURN -- this indicates that the RETURN key, located on the keyboard, should be pressed.

ASSIGNING YOUR USERID

Assigning your own userid (code name) to each one of your projects allows you to keep all of your files for each project separate from other user files and/or files you have created for different projects. Before you begin a project it is a good idea to assign a code name to the project and then use that name to identify your project to the HP 64000 Development Station whenever you sign on. In this way, whenever you sign on to the station using that userid, all of the files that you have created and will create, will be available to you (without having to search through everyone else's files or files you have created for another one of your own projects).

There are a few simple rules you must follow when you assign your userid. The userid must start with an upper case alphabetic character and is limited to six characters, or less. If more than one character is used, then after the first character, the following characters, up to five, may be alphanumeric. Upper and lower case alphabetic characters and the underscore character are accepted after the first upper case alphabetic character.

To assign the userid you have selected for your project perform the following steps:

Press the ---ETC--- softkey until the *userid* softkey appears as the left-most softkey on the softkey selection line.

Press the *userid* softkey and type in the code name that you have selected (we will be using the userid YOURID throughout the manual).

Press the **RETURN** key on the keyboard.

SINGLE-MODULE SYSTEMS -VS- MULTI-MODULE SYSTEMS

In this chapter, and throughout this manual, references will be made to single-module systems and multi-module systems. It is important that you know what these refer to, since the system that you have will determine what softkey labels will appear on the softkey label line of the display, and therefore, what options are available in each case.

A single-module system is defined as a system that contains only one hardware system (in this case, the emulation system; which includes the internal analyzer) in the development station, and where the HP High Level Software Analyzer module for your particular emulator is not resident on the system disc.

A multi-module system is defined as a system that contains more than one hardware system (e.g.; two emulation systems, an emulation system and a state analyzer system, etc.) in the development station and/or a system where the HP High Level Software Analyzer module for your particular emulation system is resident on the system disc.

NOTE

You can have a configuration where the development station contains only one system (e.g.; the emulation system), and the system disc contains the HP High Level Software Analyzer module for your particular emulation system. Even though there is only one hardware system installed in the development station, this configuration is considered to be a multi-module configuration because the software for the high level software analyzer is resident on the system disc.

In the above configuration, if the HP High Level Software Analyzer software module resident on the system disc is not for use with your particular emulation system, the configuration you have is considered to be a single module system.

If you have a single-module system installed in your development station, one of the monitor level softkey selections displayed across the bottom of the display will be different than if you have a multi-module system installed in your development station. The *emulate* softkey label will appear for the single-module system. The *meas_sys* softkey label will appear if you have a multi-module system installed in your development station.

The software you received with your emulation system is designed to distinguish between single-module systems and multi-module systems, therefore all softkey differences are automatic.

For the purposes of the examples given in this chapter, the emulation system should include the components listed below (note that the internal analysis feature is present) to take full advantage of the *emulate* softkey label examples found in single-module systems and the *meas_sys* softkey label examples found in the multi-module systems.

Internal Analysis Board
Emulator Control Board
Memory Control Board

Memory (Static RAM) Board
680XX Emulation Probe (Pod)

SOFTKEYS

Softkey templates at the HP 64000 system monitor level provide softkeys which, among numerous other functions, are used to access your emulation system. Softkey templates at the emulation system level provide softkeys used to operate your emulation system. The softkey template sets are discussed below.

Monitor Level Softkeys

Three softkey templates are available at the HP 64000 system monitor level. The three templates are shown below.

```
edit   compile  assemble  link   *emulate  prom_prog  run   ---ETC---  
directory  purge  rename  copy   library   recover   log   ---ETC---  
userid  date&time  opt_test  terminal  <CMDFILE>  gen_db  -BACKUP-  ---ETC---
```

* Multi-module systems will have the *meas_sys* softkey in place of the *emulate* softkey.

Emulation System Softkeys

The emulation system displays two softkey templates when your development station contains only your emulation system hardware (with the exception shown below). It displays three templates when your development station contains more than one hardware system (e.g.; your emulation system plus a state analyzer system, a timing analyzer system, etc). The templates available in each case are shown below.

** DEVELOPMENT STATIONS CONTAINING ONLY YOUR EMULATION SYSTEM HARDWARE:

```
run   trace  step  display  modify  break   end   ---ETC---  
load  store  list  stop_trc  reset  wait   <CMDFILE>  ---ETC---
```

** If your system disc contains the software for the high level software analyzer, you may, with some versions of the Operating System software, have three software templates, as shown below. The *specify* and *execute* softkeys are not used with the high level software analyzer. These softkeys will not appear as an option in single emulation systems on the latest version of the Operating System software.

DEVELOPMENT STATIONS CONTAINING YOUR EMULATION SYSTEM HARDWARE AND OTHER HARDWARE SYSTEMS:

```
run   trace  step  display  modify  break   end   ---ETC---  
load  store  list  stop_trc  reset  wait   <CMDFILE>  ---ETC---  
specify  execute  _____  _____  _____  _____  _____  ---ETC---
```

YOUR PROGRAM MODULE

All of the sample programs that are listed in this manual are provided for you on the software package disc for the 680XX emulation system. The program listings are provided for your convenience. The Module named DEMO680XX is described below:

DEMO680XX this module contains all of the sample programs used in the getting started section (MODULE) , using the linker section, and the Simulated I/O chapters.

FILES YOU WILL CREATE

You will create several files in this chapter which are used throughout this manual. The names of these files were chosen for the purposes of this manual and are not defined by either the HP 64000 system or the emulation system software. These file names are:

DEMO_CMD: a system command source file made in this chapter which is an edited version of a "logged" system command file. It is used to assemble and link the MODULE and Mon_680XX emulation monitor programs, to gain access to the emulator, and to load the linker absolute file into emulation/user memory.

DEMO: a system command source file made in this chapter which is an edited (simplified) version of the DEMO_CMD file. It does not contain the assemble and link commands. It is used throughout this manual to gain access to the emulation system, and to load the linker absolute file into emulation/user memory.

RUN: an example source command file made in this chapter to illustrate the use of command files during an emulation session.

You will note as you progress through this chapter that you will assign the name DEMO to other files you will create as a result of linking and answering the configuration questions. This is a unique feature of the emulator. It simplifies the process of keeping track of your files by using one file name for all functions required to gain access to the emulator. You use the same file name to link your file (DEMO:link_com), answer the emulation configuration questions (DEMO:emul_com), and load emulation/user memory (DEMO:absolute). There are also two more DEMO files created during the course of this chapter. These are the DEMO:link_sym file, which gives you the linking information, and the DEMO:listing file, which gives you the address information for the linked programs.

Note also that when you assemble your programs each program will produce two more files with the same name as the source program. These are the "reloc" and "asmb_sym" files which tell you the location of your relocated file after assembling and the assembler symbols in your program.

For a more detailed discussion concerning the number of files and their identity, refer to chapter 4 in the paragraph entitled "Why So Many Files Have The Same Name".

ORGANIZING YOUR PROGRAM MODULES

In this section you will create a small loop program, and copy the emulation monitor program to your userid. As you learned in chapter 1, the emulation monitor program allows you to observe what is going on while your program is being executed. The emulation monitor program is supplied on the same flexible disc as the Emulator System Software. It must be copied to your userid, assembled, linked to your program modules, and mapped to emulation memory. The mapping procedure will be accomplished as part of the procedures in this chapter, but will be explained in detail in chapter 5.

Copying the 680XX Programs to Your Userid

Press *copy* Mon_680XX:HP *to* Mon_680XX:YOURID **(RETURN)**.

Press *copy* MODULE:HP24*a *to* MODULE:YOURID **(RETURN)**.

* = 3 for the 68000, * = 4 for the 68008.

NOTE

At this time, we will make a modification to the monitor program that is necessary to allow us to run the demonstration program. This change will allow you to use the exception vector lookup table that is provided with the monitor program. Later, when you are developing your own code, you will probably want to write your own lookup table. When you write your own table, you can reverse the process that follows. This change is necessary because the table provides the monitor with the various addresses that it needs to operate.

Press *edit* Mon_680XX **(RETURN)** Use the **(NEXT PAGE)**, **(PREV PAGE)**, **(ROLL UP)**, and the **(ROLL DOWN)** keys to find:

```
*****
*   THE FOLLOWING TABLE DEFINES THE FIRST 11 EXCEPTION VECTORS
*   FOR THE CONVENIENCE OF THE USER.  IF THE USER WISHES TO
*   DEFINE ANY OR ALL OF THESE VECTORS, THEY MAY BE CHANGED
*   TO OR FROM COMMENTS.
*****
*   ORG 0      ---RESET---
*   DC.L SP_TEMP
*   DC.L RESET_ENTRY
*   ORG 8      ---BUS ERROR---
*   DC.L BE_ENTRY
*   ORG 0CH    ---ADDRESS ERROR---
*   DC.L AE_ENTRY
*   ORG 10H    ---ILLEGAL INSTRUCTION---
*   DC.L II_ENTRY
*   ORG 14H    ---ZERO DIVIDE---
*   DC.L ZD_ENTRY
*   ORG 18H    ---CHK INSTRUCTION---
*   DC.L CI_ENTRY
*   ORG 1CH    ---TRAPV INSTRUCTION---
*   DC.L TI_ENTRY
*   ORG 20H    ---PRIVILEGE VIOLATION---
*   DC.L PV_ENTRY
*   ORG 24H    ---TRACE---
*   DC.L T_ENTRY
*   ORG 28H    ---1010 EMULATOR---
*   DC.L EA_ENTRY
*   ORG 2CH    ---1111 EMULATOR---
*   DC.L FE_ENTRY
*
*****
```

Figure 3-1. Exception Vector Table (Commented)

Now, using the *revise* softkey, remove the "*" from the start of each line of code to make your code look as follows:

Emulator/Analyzer 68000/68008 Getting Started

```
*****
*   THE FOLLOWING TABLE DEFINES THE FIRST 11 EXCEPTION VECTORS
*   FOR THE CONVENIENCE OF THE USER.  IF THE USER WISHES TO
*   DEFINE ANY OR ALL OF THESE VECTORS, THEY MAY BE CHANGED
*   TO OR FROM COMMENTS.
*****
ORG 0      ---RESET---
    DC.L SP_TEMP
    DC.L RESET_ENTRY
ORG 8      ---BUS ERROR---
    DC.L BE_ENTRY
ORG 0CH    ---ADDRESS ERROR---
    DC.L AE_ENTRY
ORG 10H    ---ILLEGAL INSTRUCTION---
    DC.L II_ENTRY
ORG 14H    ---ZERO DIVIDE---
    DC.L ZD_ENTRY
ORG 18H    ---CHK INSTRUCTION---
    DC.L CI_ENTRY
ORG 1CH    ---TRAPV INSTRUCTION---
    DC.L TI_ENTRY
ORG 20H    ---PRIVILEGE VIOLATION---
    DC.L PV_ENTRY
ORG 24H    ---TRACE---
    DC.L T_ENTRY
ORG 28H    ---1010 EMULATOR---
    DC.L EA_ENTRY
ORG 2CH    ---1111 EMULATOR---
    DC.L FE_ENTRY

*****
```

Figure 3-2. Exception Vector Table (Un-commented)

Press *end* (**RETURN**) to save your changes.

By removing all of the "*"s from this section of the monitor, you have now "un-commented" the table which makes it usable. This table provides all of the addresses that the monitor needs to operate.

Looking at Your Sample Program

```
'680XX'  
  
          GLB          START, LOOP_1  
  
          PROG  
START    CLR.B        D0  
LOOP_1   SUBQ.B       #1,D0  
          BNE         LOOP_1  
          JMP         START  
          END
```

INITIATING A SYSTEM "LOG" COMMAND FILE

A command file is a file that lists a series of commands that you have predetermined must be performed to accomplish a particular function. An easy way to make a command file of this type is to use the "log" commands. The *log_commands to* command will initiate a system command file by logging your softkey commands to a file which you name. The command file will be ended when the *log_commands off* command is given. In this case, that will be after the emulation memory has been loaded. At that time the command file will be edited and its function will be explained.

The reason we are using the "log" commands at this time is because the commands we will be using are ideal for setting up for, and accessing the emulation system. Once the file is created, all you have to do is edit the file. You then have a permanent way to enter your emulation session. Since this chapter is for the purpose of getting you started, you will need to go through all of the steps required to enter a session anyway. The "log" commands, as you will see, allow you to do this the easy way, once you have gone through the sequence one time.

PRESS *log to* DEMO_CMD (RETURN).

ASSEMBLING AND LINKING THE PROGRAM MODULES

Assembling Modules

PRESS *assemble* MODULE (RETURN).
PRESS *assemble* Mon_680XX (RETURN).

PRESS the *directory* softkey and (RETURN). Note that both of the assembled modules now each have three file types listed for each file name: 1) :source, 2) :reloc, and 3) :asmb_sym.

Linking Modules

PRESS *link listfile* DEMO **(RETURN)**.

<u>Linker Question</u>	<u>Your answer</u>
1) Object files? (Name of Relocatable files)	MODULE (RETURN)
2) Library files? (Files which have been libraried previously--in this case none)	(RETURN)
3) Load addresses:PROG,DATA,COMN,A5=0002000H,0000000H,0000000H,0000000H (note assignment of 002000H to PROG address This instruction places modules of the program at specific addresses. ORG statements in program override any answer here.)	(RETURN)
4) More files?	yes (RETURN)
5) Object files?	Mon_680XX (RETURN)
6) Library files?	(RETURN)
7) Load addresses:PROG,DATA,COMN,A5=0007000H,0000000H,0000000H,0000000H (Note assignment of 007000H to "PROG=")	(RETURN)
8) More files?	no (RETURN)
9) list,xref,overlap_check,comp_db= on off on off	(RETURN)
10) Absolute file name=	DEMO (RETURN)

If you have a printer on your HP 64000 system, print out the listing file which you requested when you initiated the linker (*link listfile* DEMO), as follows:

PRESS *copy* DEMO:*listing to printer* **(RETURN)**.

If you do not have a printer, display the listing to the screen, as follows:

PRESS *edit* DEMO:*listing* **(RETURN)**.

The information shown in figure 3-3 will appear on the screen. You will use the information contained in this listing file when you map emulation memory in the next section; copy the address range information. The address range for MODULE is 002000 thru 002010 and that the address range for Mon_680XX is 0007000 thru 00075A0. This the information needed in the next section.

HP 64000 Linker

FILE/PROG NAME	PROGRAM	DATA	COMMON	ABSOLUTE	DATE

MODULE:Y68000	0002000				Tue, 19 Mar 1985
next address	000200C				
Mon_68000:Y68000	0007000		0000000-0000007		Tue, 19 Mar 1985
				0000008-000000B	
				000000C-000000F	
				0000010-0000013	
				0000014-0000017	
				0000018-000001B	
				000001C-000001F	
				0000020-0000023	
				0000024-0000027	
				0000028-000002B	
				000002C-000002F	
next address	00075A0				

XFER address= 0000000 Defined by DEFAULT
absolute & link_com file name=DEMO:Y68000
Total# of bytes loaded= 00005DC

Figure 3-3. Linker Output Listing

PRESS *end* **RETURN**

NOTE

The displays shown in this manual are 68000 displays. Displays for the 68008 may be slightly different.

GAINING ACCESS TO THE EMULATION SUBSYSTEM

To gain access to the emulation system once you have assembled and linked your files, you need to leave the monitor level softkeys and answer the emulation configuration questions. This section will show you how to arrive at the configuration questions. The section that follows will show you how to answer the configuration questions. When the emulation configuration questions are answered, you will have the emulation softkey selections available to load your absolute file (created by the linker), and start your emulation session.

The first softkey label line at the system monitor level will contain a softkey label of either *emulate* or *meas_sys*. If your system is a single-module system, *emulate* will appear on the softkey label line. If your system is a multi-module system, *meas_sys* will appear on the softkey label line.

If the *emulate* softkey is present, press the *emulate* softkey and press **(RETURN)**. The system software will immediately access the emulation system beginning with the configuration questions.

Once the emulation configuration questions are answered (as explained later in this chapter), you are at the emulation softkey selection level. To begin using the emulator you must then load the absolute file you created when you linked your programs. This will also be discussed later in this chapter.

If the *meas_sys* softkey label is present, press the *meas_sys* softkey and press **(RETURN)**. The system software will access the measurement system level softkeys, and the *eM680XX* softkey label will appear on the screen, along with the names of any other analysis modules in the development station. This level of softkeys allows access to any one of the emulation or analysis modules installed in the development station. Press the *eM680XX* softkey and press the **(RETURN)** key. The system software will immediately access the emulation system beginning with the configuration questions. Once the emulation configuration questions are answered (as explained later in this chapter), you are at the emulation softkey selection level. To begin using the emulator you must then load the absolute file you created when you linked your programs. This will also be discussed later in this chapter.

NOTE

If the emulation system is not the only system in the development station, the *eM680XX* softkey label will appear followed by a number (e.g.; *eM680XX_S*). The number denotes the slot in the development station that contains the emulator control board.

Figure 3-4 shows the syntax for gaining access to the emulator from the *emulate* softkey. Figure 3-5 shows the syntax for gaining access to the emulator from the *meas_sys* softkey.

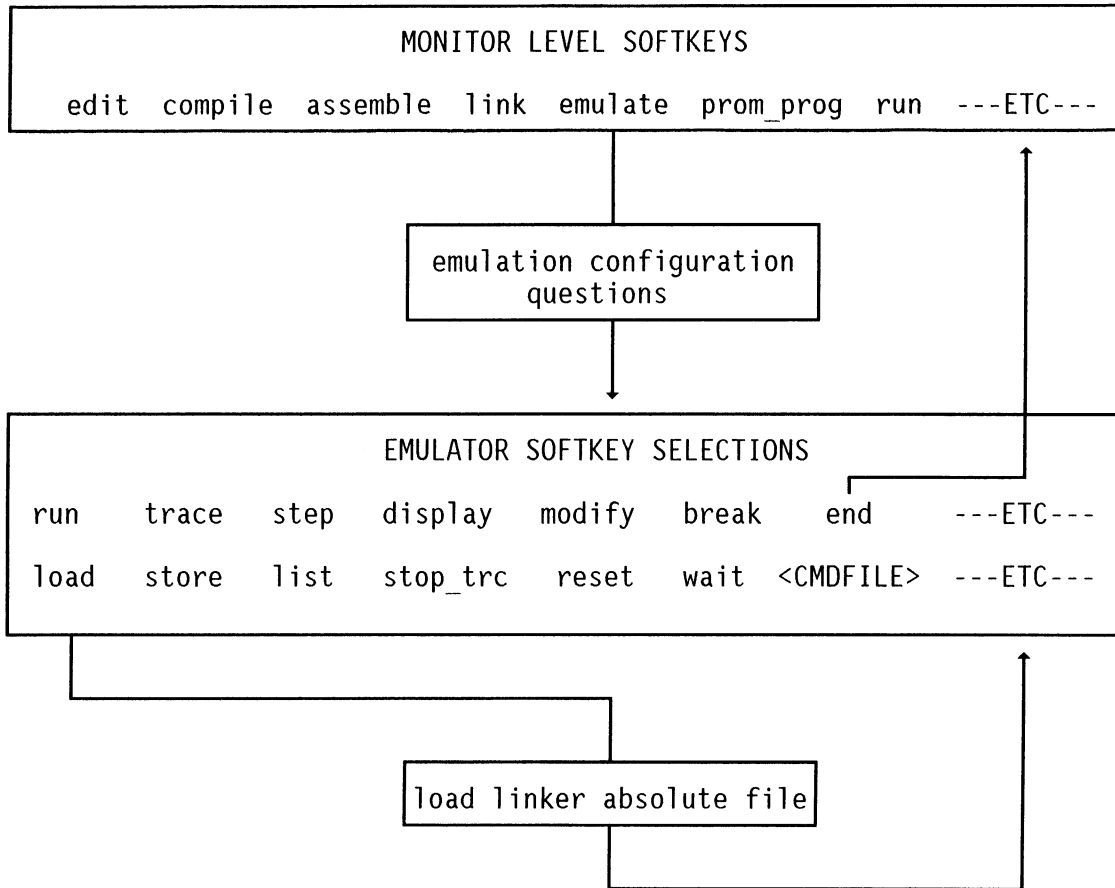


Figure 3-4. Utility Keys Used For Transportation (from "emulate")

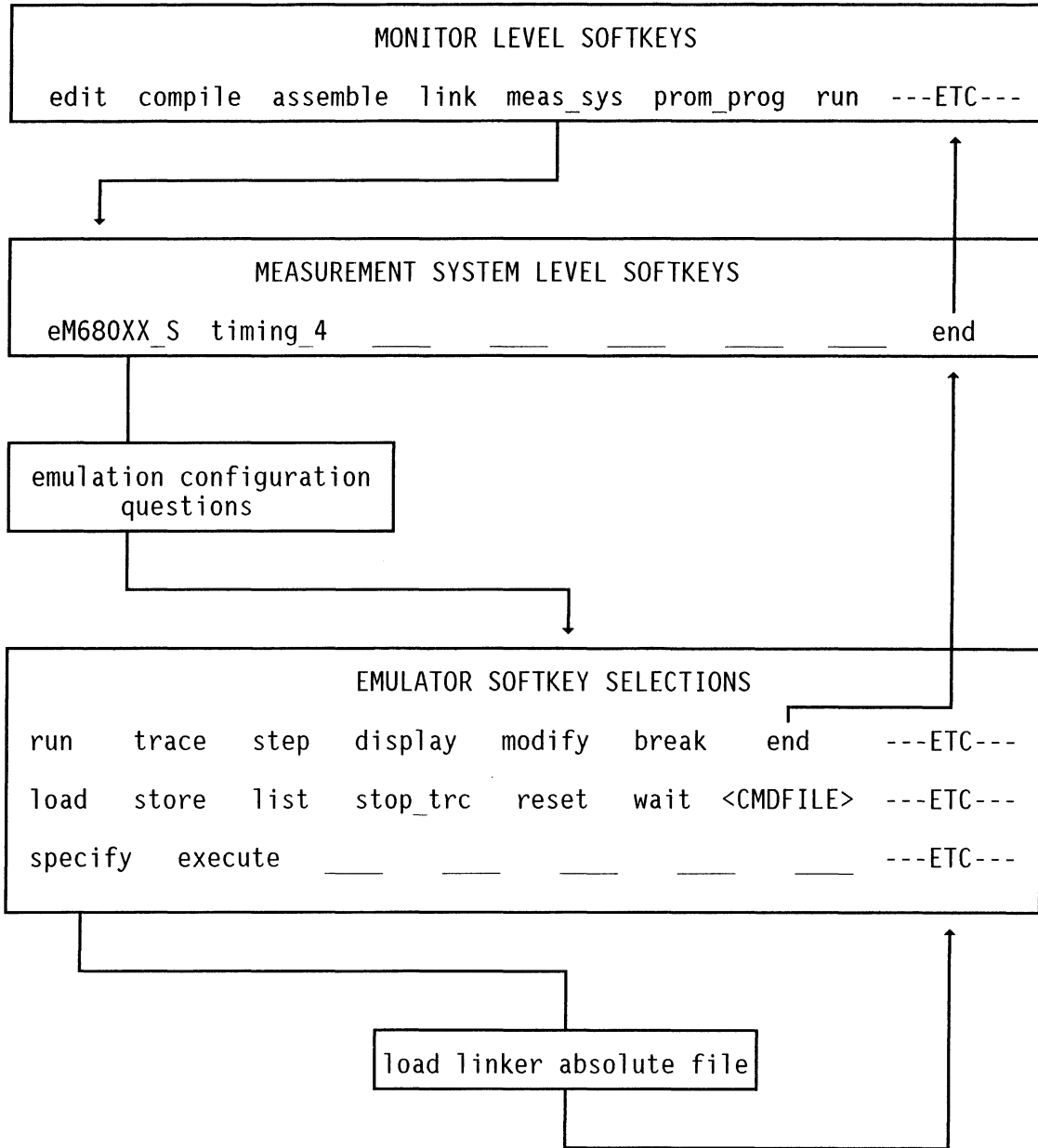


Figure 3-5. Utility Keys Used For Transportation (from "meas_sys")

ANSWERING THE EMULATION CONFIGURATION QUESTIONS

At this point you should be at the first emulation configuration question. If not, return to the previous section entitled "Gaining Access to the Emulation Subsystem" and access the emulator as described therein. Remember, however, if you do return, the "log" command file you initiated earlier in this chapter is still being updated. This only means that the logged file will contain more information after the log file is ended than is shown later in this chapter. When you have gained access to the emulator, answer the emulation configuration questions as shown below.

<u>Emulation Configuration Question</u>	<u>Your Answer</u>
1) Slot number of analysis card? 4	RETURN
2) Micro-processor clock source? internal	RETURN
3) Restrict to real-time runs? no (Display of memory or registers is not allowed during real-time -- unless a break condition is entered)	RETURN
4) Number of significant address bits? 24 (NOTE: for the 68008, the question is 20)	RETURN
5) Break processor on write to ROM? yes	RETURN
6) (Enter the following softkey information. Type in the addresses)	
0H thru 0FFFH emulation rom	RETURN
2000H thru 2FFFH emulation rom	RETURN
7000H thru 7FFFH emulation ram	RETURN
default guarded	RETURN
end	RETURN
7) Modify simulated I/O? no	RETURN
8) Re-configure emulator pod? no	RETURN
9) Modify interactive measurement specification? no	RETURN
10) Command file name?	DEMO RETURN

If you watch the STATUS line you will see that the emulator is loading the memory mapper. When loading is complete the STATUS line will indicate that the emulation processor is Reset. The emulator is ready to be used.

LOADING EMULATION MEMORY

Once you have answered the emulation questions and assigned a command file name to that configuration of question/answers, the emulator softkey selections are present at the bottom of your CRT display. You are now at the beginning of an emulation session. Before you can begin to perform emulation, you must load emulation memory with the absolute file created when you linked your program modules together. To load emulation memory, press the *load* softkey, type in the name of your absolute file (DEMO in this case), and press the **(RETURN)** key. You are now ready to begin an emulation session, as discussed later in this chapter.

Now it is time to edit the commands which have been logged to the command file DEMO_CMD.

PRESS *end* **(RETURN)**. This will return you to the monitor level softkeys if you started with the *emulate* softkey (see figure 3-3). It will return you to the measurement system level softkeys if you started with the *meas_sys* softkey (see figure 3-4). PRESS *end* **(RETURN)** again to return to the monitor level softkeys from the measurement system level softkeys.

ENDING THE "LOG" COMMAND FILE

The log command file you initiated prior to assembling and linking your modules is now complete enough so that, with some editing, it will accomplish the purpose for which it was initiated. That is, to initiate an emulation session by assembling and linking your modules, answering the emulation questions, and loading memory with you absolute file. End the "log" session as follows:

PRESS *log off* **(RETURN)**

The paragraph entitled "Logging Commands to a Command File" in the next section will show you how to edit the "log" command file.

BUILDING SYSTEM COMMAND FILES

Earlier, when you linked the program modules and configured the emulator, you set up a link command file (DEMO:link_com) and an emulation configuration command file (DEMO:emul_com) so that the next time you need to perform one of those processes all you have to do is to call up the DEMO command file.

For example, if you have a single-module system:

PRESS *emulate* DEMO *load* DEMO **(RETURN)**.

If you have a multi-module system:

PRESS *meas_sys* **(RETURN)**.

PRESS *eM680XX_S* DEMO **(RETURN)**. (Where S indicates the slot)

PRESS *load* DEMO **(RETURN)**.

Watch the status and command lines as your emulator is configured and loaded. The emulation command file is answering all of the configuration questions for you. The linker command file you created earlier, in turn created an absolute file (also called DEMO), which is now used to load emulation memory. Press *end* **(RETURN)** to return to the monitor level software for single-module systems or *end* **(RETURN)**, *end* **(RETURN)** for multi-module systems.

If you edit any of your program modules you will need to reassemble the edited module(s) and relink all over again. You can build a system command file which will do all of these tasks for you. You build a system command file in two ways: type the commands in an edit file or log the commands to a file.

When you begin using the emulator you may find that you have a series of commands which you use repeatedly. You can build a system command file for this series and call it while in emulation.

Logging Commands to a Command File

You initiated a log command file, and logged commands to that file when you assembled and linked your program modules, and configured the emulator, in the previous sections. The command file was initiated when you entered the command *log to* and the command file was ended when you entered the command *log off*. To edit the command file:

PRESS *edit* DEMO_CMD **(RETURN)**.

The edit file illustrated below is the result of the operations you performed between the time you entered the *log to* command and the time you entered the *log off* command. The files shown below are the result of following the instructions exactly as they were given earlier in this chapter. If your file does not look exactly as shown, it can be edited as described in the next discussion.

"emulate" Softkey Systems

```
START
  1 assemble MODULE
  2 assemble Mon_680XX
  3 link listfile DEMO
  4 copy DEMO:listing to printer
  5 edit DEMO:listing
  6 end
  7 emulate load DEMO
  8 end
END
```

"meas_sys" Softkey Systems

```
START
  1 assemble MODULE
  2 assemble Mon_680XX
  3 link listfile DEMO
  4 copy DEMO:listing to printer
  5 edit DEMO:listing
  6 end
  7 measurement_system
  8 eM680XX_S
  9 load DEMO
 10 end
 11 end
END
```

Revise the edit file to look like that shown below (remember to insert the link and emulate command file names):

"emulate" Softkey Systems

```
START
  1 assemble MODULE
  2 assemble Mon_680XX
  3 link DEMO listfile DEMO
  4 emulate DEMO load DEMO
END
```

"meas_sys" Softkey Systems

```
START
  1 assemble MODULE
  2 assemble Mon_680XX
  3 link DEMO listfile DEMO
  4 measurement_system
  5 eM680XX_S DEMO
  6 load DEMO
END
```

PRESS *end* **(RETURN)**.

The command file is now ready to be used; just type in your command file name (DEMO_CMD) and press **(RETURN)**.

Notice the command and status lines as your command file is being executed. You will note that the modules are being assembled and linked, the emulator is being configured, and memory is being loaded.

Press *end* **(RETURN)** to return to the monitor level software for single-module systems or *end* **(RETURN)**, *end* **(RETURN)** for multi-module systems.

Typing The Commands Into a Command File

It is not necessary to log commands to a command file. You may simply open a new edit file and type in the command sequence. You must, however, observe the following rules:

- Type all commands exactly as they appear on the command line when the softkey is pressed. This is not necessarily the same as the softkey label.
- All commands are lower case (e.g.; emulate)
- All parameters are upper case (e.g.; your program name, MODULE).
- End each line by pressing **(RETURN)**.

A system command file is one which is activated from the system monitor level of softkeys. It is used to get you into the emulation session quickly and simply. It can also contain commands to start your emulation session at a certain predetermined point (e. g.,; run from <ADDRESS>, trace after <G_L_SYM>) once you have entered your emulation session. For purposes of this discussion we will treat these two operations separately. Keep in mind, however, that these two command files can be combined into one file which would be initiated at the system monitor level of softkeys. We will call the command file used to initiate the emulation session the "system command file". We will call the command file used inside of the emulation session the "emulation command file".

TYPING A SYSTEM COMMAND FILE. The file you created above, in the paragraph entitled "Logging Commands to a Command File" by using the "log_commands" method, would be one you would use after you have modified one or more of your modules. It would not be practical to reassemble and relink each time you wished to enter an emulation session. All that is necessary to enter a session after your emulation and linker command files are built is to build a system command file that uses these two files. Do this as follows:

For "emulate" Softkey Systems

PRESS *edit* (RETURN).

Type in the following line:

```
emulate DEMO load DEMO
```

PRESS *end*, type in DEMO, and press (RETURN).

For "meas_sys" Softkey Systems

PRESS *edit* (RETURN).

Type in the following lines:

```
measurement_system  
eM680XX_S DEMO ( S indicates the slot in the HP64000 )  
load DEMO
```

PRESS *end*, type in DEMO, and press (RETURN).

You now have a system command file, named DEMO, which allows you to enter an emulation session from the monitor level softkeys with a minimum of time and effort. All you need to do is type in DEMO, and the command file automatically answers the emulation configuration questions and loads your linked program into memory. You are then ready to run your emulator.

You will note, at this time, that when you press the *directory* softkey and (RETURN) that there are six file types for the file named DEMO. They are 1) :source, 2) :emul_com, 3) :link_com, 4) :absolute, 5) :link_sym, and 6) :listing. Refer to chapter 4 in the paragraph entitled "Why Do So Many Files Have The Same Name?" for further information concerning file names and types.

TYPING AN EMULATION COMMAND FILE. Emulation command files are used to start your emulation session at a predetermined point once you have entered your emulation session, or to restart your session from that point. Suppose, for instance, you found it convenient to start your session by viewing a trace after the global symbol START, then after a 15 second interval, viewing memory mnemonics after START. Build the emulation command file as follows:

PRESS *edit* **(RETURN)**.

Type in the following lines:

```
run
trace after START
run from START
display trace
wait 15
display memory START mnemonic
```

PRESS *end*, type in RUN, and press **(RETURN)**.

You now have an emulation command file, named RUN, which will start your emulation session at the point described above once you enter your emulation session and type in RUN.

USING THE EMULATOR

(If you are beginning from the system monitor level of softkeys)

Type in DEMO from the keyboard.

To See Real-Time Tracing of Processor Activity

Tracing is a feature which is available when an internal analyzer is included with the emulation system.

When the STATUS line reads "M680XX--Reset" :

PRESS *trace after* START **(RETURN)**.

PRESS *run from* 2000H **(RETURN)**.

The information shown in figure 3-6 will appear on screen. Use the **(ROLL UP)**, **(ROLL DOWN)**, **(NEXT PAGE)**, and/or **(PREV PAGE)** cursor control keys to scroll through the entire trace.

```

Trace: bus cycle data                break: none                count:
line#  address  opc/data  mnemonic opcode or status                time, relative
-----
after  002000   4200   CLR.B  D0
+001   002002   5300   SUBQ.B #1,D0                        <1.  uS
+002   002004   66FC   BNE.B  0002002H                      1.  uS
+003   002006   4EF9   JMP    *****
+004   002002   5300   SUBQ.B #1,D0                        1.  uS
+005   002004   66FC   BNE.B  0002002H                      <1.  uS
+006   002006   4EF9   JMP    *****
+007   002002   5300   SUBQ.B #1,D0                        1.  uS
+008   002004   66FC   BNE.B  0002002H                      <1.  uS
+009   002006   4EF9   JMP    *****
+010   002002   5300   SUBQ.B #1,D0                        1.  uS
+011   002004   66FC   BNE.B  0002002H                      1.  uS
+012   002006   4EF9   JMP    *****
+013   002002   5300   SUBQ.B #1,D0                        1.  uS
+014   002004   66FC   BNE.B  0002002H                      1.  uS
+015   002006   4EF9   JMP    *****

```

STATUS: M680XX--Running Trace complete 10:48

trace after 002000H

run trace step display modify break end ---ETC---

Figure 3-6. Trace After 002000H

To Break Into Emulation Monitor and See Registers

PRESS *display registers* (RETURN).

Information similar to that shown below will be present on the screen.

```

M680XX  Registers
-----
Next PC  002004  STATUS 2708 < s n >          USPT 0000761A  SSPT 0000761A
D0-D7  00000077  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
A0-A7  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  0000761A

```

Stepping

The step <N> command normally causes one instruction or N instructions to be executed.

Note that in some single word instructions the step function will 'double step', that is, it will only show the second of two instructions that were executed. This can be seen in the example below. Note that the Program Counter skipped from 2002H to 2004H. The instruction at 2002H was executed, but does not show up in the trace.

PRESS *step from* 2000H **RETURN**

Information similar to that shown below will be added to the screen.

```
Next PC 002002 STATUS 2708 < s n > USPT 0000761A SSPT 0000761A
D0-D7 00000089 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-A7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000761A
```

```
PC 002000 Opcode 4280 CLR.B D0
Next PC 002004 STATUS 2719 < sxn c> USPT 0000761A SSPT 0000761A
D0-D7 000000FF 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-A7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000761A
```

PRESS *step* **RETURN**

Information similar to that shown below will be added to the screen.

```
PC 002004 Opcode 66FC BNE.B 0002002H
Next PC 002002 STATUS 2719 < sxn c> USPT 0000761A SSPT 0000761A
D0-D7 000000FF 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-A7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000761A
```

Press *end* **RETURN** to return to the monitor level software for single-module systems or *end* **RETURN**, *end* **RETURN** for multi-module systems.

USING THE COMMAND FILES

This section will show you the results of the system command file and the emulation command file which you built previously in the section entitled "BUILDING SYSTEM COMMAND FILES". Assuming that you are at the system monitor level of softkeys, proceed as follows:

Type in DEMO from the keyboard.

When the STATUS line reads M680XX--Reset, type in RUN from the keyboard.

Your screen will show a trace like that shown in figure 3-7 except take note that the following line appears beneath the STATUS line in inverse video.

Waiting for any keystroke or delaying for 15 seconds.

This line is the result of the "wait 15" entry in your "RUN" emulation command file which you built earlier in this chapter. It causes a 15 second wait before the next command in the RUN command file is executed. The time will count down to 0 on the display. When the 15 seconds counts down to zero, the display information will change to that shown in figure 3-8.

```

Trace: bus cycle data                break: none                count:
line#  address opc/data mnemonic opcode or status                time, relative
-----
  after 002000 4200 CLR.B D0
+001 002002 5300 SUBQ.B #1,D0                <1.  us
+002 002004 66FC BNE.B 0002002H            1.  us
+003 002006 4EF9 JMP *****                1.  us
+004 002002 5300 SUBQ.B #1,D0                1.  us
+005 002004 66FC BNE.B 0002002H            <1.  us
+006 002006 4EF9 JMP *****                1.  us
+007 002002 5300 SUBQ.B #1,D0                1.  us
+008 002004 66FC BNE.B 0002002H            <1.  us
+009 002006 4EF9 JMP *****                1.  us
+010 002002 5300 SUBQ.B #1,D0                1.  us
+011 002004 66FC BNE.B 0002002H            1.  us
+012 002006 4EF9 JMP *****                <1.  us
+013 002002 5300 SUBQ.B #1,D0                1.  us
+014 002004 66FC BNE.B 0002002H            1.  us
+015 002006 4EF9 JMP *****                <1.  us

STATUS: M680XX--Running                Trace complete                11:05
  Waiting for any keystroke or delaying for 15 seconds
wait 15

```

Figure 3-7. RUN Emulation Command File Trace 1

Emulator/Analyzer 68000/68008
Getting Started

```
Memory :mnemonic
address
-----
002000    CLR.B    D0
002002    SUBQ.B   #1,D0
002004    BNE.B    0002000H
002006    JMP      0002000H
00200C    ORI.W    #00207H,[A3]+
002010    MOVE.L   D1,D0TE PTR 06H[D1],DH
002012    ORI.B    #045H,[A7]+
002016    ORI.     Illegal Operand
002018    ANDI.B   #044H,D1H
00201C    ORI.     Illegal Operand
00201E    ORI.L    #0004F0061H,0000BH[A5]
002026    EORI.B   #08EH,00049H[A1]
00202C    ORI.B    #05DH,A2
002034    ANDI.W   #00003H,0002250H[PC],
00203A    ORI.B    #039H,A7

STATUS: M680XX--Running          Trace complete          ____ 11:15

display memory START mnemonic

run      trace      step      display      modify      break      end      ---ETC---
```

Figure 3-8. RUN Emulation Command File Trace 2

This completes your introduction to the 680XX emulation system. You have created and assembled a sample program, then linked your program to the monitor program. In the process of these operations, you have built and run command files, and used a few of the basic features of the emulation system. For more specific and detailed operational data, refer to the information contained in the following chapters.

Chapter 4

PREPARING YOUR PROGRAM MODULES FOR EMULATION

OVERVIEW

This chapter will:

- Review the assembling and linking processes.
- Discuss file names and their similarities.
- Explain where the monitor program should be located in your absolute program.

GENERAL

In this chapter, whenever 680XX is mentioned (even as part of a filename) both the 68000 and the 68008 are assumed unless there are differences between the two processors. In that case, the differences will be noted. When the name 680XX appears in any program listings, just insert the appropriate name of your processor.

INTRODUCTION

This chapter reviews the software development steps which must be completed before the 680XX emulation system can run your program. As you learned in chapter 3, your program modules must be assembled and then linked together with the 680XX emulation monitor program.

In this chapter two example programs will be created that demonstrate how programs that call one another, i.e. a subroutine, must be mapped. It is likely that you will have several modules in your program and they will need to be linked in a similar manner. Also the emulation monitor program provided with your emulation system software will be assembled and linked. The development station displays and output listings for each step of the process will be provided.

No emulator is needed to perform the assembling and linking, however, in order to run the emulation system, the 680XX monitor software (which is on the flexible disc containing the emulator system software) must be assembled and linked prior to initiating the emulation process.

To perform the processes in this chapter you need the 68000 Family Assembler Software.

USING THE ASSEMBLER

The assembler generates object code (machine code) from a source file (the program you have typed into a file). The object code is relocatable. Each module is referenced to its own beginning, each starting at address 0, provided that the ORG statement is not part of your source program. The ORG statement takes precedence over anything assigned when you answer the linker questions.

The Relationship of Linker Mapping and Emulation Memory Mapping

The linker map and emulation configuration map work together. The linker map specifies which addresses in memory will be occupied by code from the sample program. The emulation configuration map specifies whether these addresses are located in the hardware of the emulation system or the target system, and whether these hardware addresses are ROM or RAM memory, or they are guarded (nonexistent) memory.

The following two sample programs will be used to demonstrate how the linker map and the emulation configuration map work together in the software development process. These programs are included in the software that you received when you bought your HP 680XX emulation system.

Sample Program Number 1

```
1 "68000"
2
3          GLB          START
4
5          EXTERNAL    ADDSUB
6
7          PROG
8 START   MOVEA        #03FFCH,A7
9         MOVE.L      DATA1,COMN1
10        JSR         ADDSUB
11        MOVE.L      COMN2,DATA2
12        JMP         START
13
14        DATA
15 DATA1   DC.L      0001H
16 DATA2   DC.L      0000H
17
18        COMN
19 COMN1    DC.L      0FFFFH
20 COMN2    DC.L      0FFFFH
21
```



```
Sample Program Number 2
1 "680XX"
2
3          GLB          ADDSUB
4
5          PROG
6 ADDSUB          MOVE.L          COMN1,D0
7          ADD.L          DATA1,D0
8          MOVE.L          D0,COMN2
9          RTS
10
11         DATA
12 DATA1          DC.L          0002H
13
14         COMN
15 COMN1          DC.L          0FFFFH
16 COMN2          DC.L          0FFFFH
```

Figure 4-1. Example Link Programs

Retrieve The Two Sample Programs as Follows:

PRESS *copy* PROGRAM1:HP24*a to PROGRAM1:YOURID **(RETURN)**.

PRESS *copy* PROGRAM2:HP24*a to PROGRAM2:YOURID **(RETURN)**.

* = 3 for the 68000, * = 4 for the 68008.

Now there are two assembly language source programs in memory. One program is designed to call the other. Each of them has an area identified as "program code", another area identified as "data code", and another area identified as "common code". Now assemble these two programs as follows:

"*assemble* PROGRAM1 **(RETURN)**"

"*assemble* PROGRAM2 **(RETURN)**"

List a directory to the screen as follows: "*directory* **(RETURN)**".

Note there are three files for PROGRAM1, and three for PROGRAM2. The "*source*" file is the file you typed. The "*reloc*" file is the relocatable file of machine code that was produced by the assembler when you assembled these files. The "*asmb_sym*" file lists each of the symbols that were recognized in the program code during assembly, along with their relocatable addresses. See the section in this Chapter entitled "Why Do So Many Files Have The Same Name?" for more information.

USING THE 680XX EMULATION MONITOR PROGRAM

A standard emulation monitor source file is supplied with each emulation system. This file must be assembled and linked by you before the emulation monitor can be loaded and executed. The program supplied will enable all of the emulation system functions to operate properly after the file is assembled, linked and loaded.

Emulation Monitor Memory Requirements (68000).

NOTE

If you are using the 68008 skip to the section entitled Emulation Monitor Memory Requirements For The 68008.

All of the code for the test program used in this demonstration will be loaded into memory space in the emulation hardware. The memory available in the emulation hardware is divided into 4K blocks of address space by the emulation system. Each 4K block begins on an even address multiple of 1000H.

The relocatable program area of the emulation monitor requires one 4K byte block of memory. This is because if you look at the :reloc file (figure 4-3) you will see under Prog length = 0764H that the program will take up 764 Hexadecimal locations. Because the value 764H is much less than 4000H, the entire 764H locations, can be mapped into one 4K byte block.

These memory requirements assume that the blocks each start on a 4K byte boundary and that the standard emulation monitor is being loaded. To check the memory requirements for the emulation monitor being used, record #1 of the emulation monitor relocatable file should be checked. An example of record #1 of the Mon_680XX relocatable file is shown in figure 4-2A.

Page # 1

```
File = Mon_680XX:YOURID:reloc  Fri, 29 Mar 1985,  9:41

Record #  1    size =  94
Name record:
Source = Mon_680XX:YOURID
Prog length = 0764H   Data length = 0000H   Comn length = 0000H
Number of externals is  0
Linker is l68000:HP
Wed, 27 Mar 1985,  8:10
Checksum = 1EB3H
```

Figure 4-2A. Mon_680XX Relocatable File - Record #1

Emulation Monitor Memory Requirements (68008)

NOTE

Use this section for configuring the 68008 only. Refer to the section entitled Emulation Memory Requirements (68000) when using the 68000.

The relocatable program area of the emulation monitor requires six 256 byte blocks of memory.

These memory requirements assume that the areas each start on a 256 byte boundary and that the standard emulation monitor is being loaded. To check the memory requirements for the emulation monitor being used, record #1 of the emulation monitor relocatable file should be checked. Look at Prog length for this value. As you can see from the listing, the program occupies 578H locations. If you convert this number to decimal, you obtain a value of 1400. Then you must divide this number by the block size of 256. This operation then gives the value of 5.5. Therefore, six blocks are required. You, of course, could convert the 256 block size to hexadecimal and divide, thereby saving a step. The results are the same either way.

An example of record #1 of the Mon_68008 relocatable file is shown in figure 4-2B.

Page # 1

```
File = RELOC:TEMP:reloc      Fri, 17 May 1985,  8:20

Record #  1    size =  90
Name record:
Source = Mon_68008:TEMP
Prog length = 0578H  Data length = 0000H  Comn length = 0000H
Number of externals is  0
Linker is l68008:HP
Fri, 17 May 1985,  7:43
Checksum = 9385H
```

Figure 4-2B. Mon_68008 Relocatable File - Record #1

Use This Information For Both Processors

The monitor program must reside in emulation memory and be mapped as RAM. This is necessary because the emulation monitor performs both reads from and writes to this area. It is recommended that in order to protect the monitor from being written over by a lost target system program, the first block above and below the data area should be mapped as guarded memory. Doing this will cause a break to be generated before the emulation monitor area has been accessed. Refer to the section in Chapter 6 "Filling in the Emulation Map" for more information on mapping these areas as guarded.

In order to run PROGRAM1 and PROGRAM2 in emulation, you will have to link them together with the MONITOR program. Press the following keys:

"*directory* **RETURN**"

See if the monitor program source file "Mon_680XX" is in your directory. If it is in the directory, enter this command:

```
"assemble Mon_680XX (RETURN)"
```

If the monitor program is not in your USERID, then make the following entry to call it into your directory:

```
"copy Mon_680XX:HP :source to Mon_680XX:<your userid> (RETURN)"
```

```
"assemble Mon_680XX (RETURN)"
```

The memory map in figure 4-3 is the plan for loading the two sample programs used in this demonstration. The absolute code will be placed as follows:

1. Program code will start at address 2000H. The code for program 2 will start at address 2500H. There is an open space between the end of the code for program 1 and the beginning of the code for program 2.
2. The common space is used to store code that is present and identical in both modules. When PROGRAM1 is loaded, the COMN code from PROGRAM1 will fill its assigned space. Then when PROGRAM2 is loaded, the COMN code from PROGRAM2 will overwrite the COMN code from PROGRAM1. Only the version of common code from PROGRAM2 will remain. This overwriting of address space will cause the emulator to show an error message. We won't worry about this message; it will be expected.
3. The data code space is contiguous code space. The first byte of data2 will immediately follow the last byte of data1.
4. In this example, the monitor program is loaded beginning at address 10000H. The monitor program is required in order to allow you to actually run the program in emulation. It must always reside in emulation RAM space.

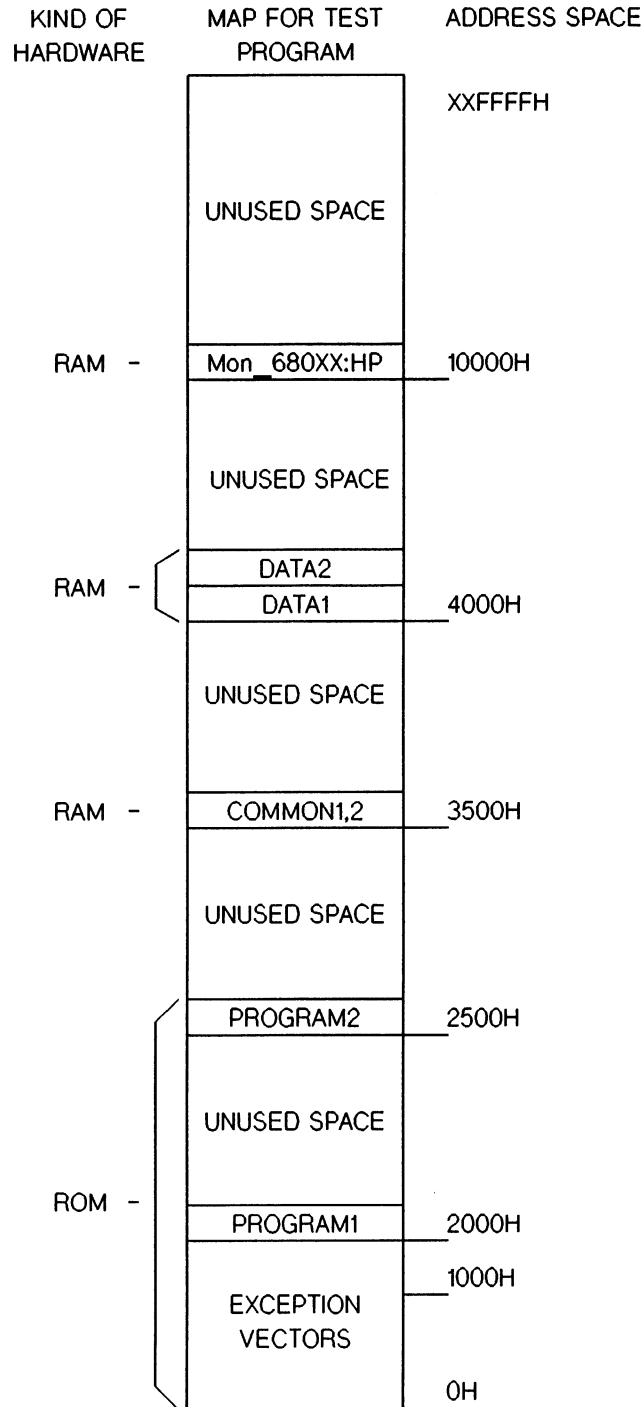


Figure 4-3. Memory Map For Test

Note that in the map the addresses 0H thru 0FFFH are designated as ROM. This is done to protect the exception vector lookup table that the monitor program will place there. (The monitor will only place the table in memory if the table is not in 'comments' within the monitor. Refer to the section entitled "Describing The Mon_680XX Program" in Chapter 8 for more details on the EXCEPTION VECTOR table). Now it is time to link the two relocatable (machine code) files and the monitor file together into one absolute file. In the link process, you make a linker map. In the linker map, you specify absolute addresses in memory where you want portions of the code from your linked modules to reside. In the example map, you see the program code assigned to specific addresses. You also see that some of the address space is assigned to emulate ROM hardware and some is assigned to emulate RAM hardware. Some of the address space can be located in the emulation hardware, and some can be located in the target system hardware. The linker map only specifies the values of the absolute addresses where each of the code segments will reside. Whether that space is in the target hardware or the emulation hardware, and whether it emulates ROM or RAM hardware will be decided when you make the emulation configuration map. Refer to chapter 5 for more information on the emulation configuration map.

Now for the link. Press the following keys:

"link **RETURN**"

"Object files? PROGRAM1 **RETURN**"

"library files? **RETURN**"

"Load addresses: PROG,DATA,COMN,A5=0002000H,0004000H,0003500H,0000000H **RETURN**"

"More files? yes **RETURN**"

"Object files? PROGRAM2 **RETURN**"

"library files? **RETURN**"

"Load addresses: PROG,DATA,COMN,A5=0002500H,CONT ,0003500H,0000000H **RETURN**"

In large programs where memory space must be conserved, you may want to load program parts as close to one another as possible. The "CONT" (continuous) entry allows you to do this. In the line above, the CONT in the DATA address space causes the first byte of DATA code for PROGRAM2 to be loaded immediately following the last byte of DATA code from PROGRAM1. Therefore, if the last byte of DATA code from PROGRAM1 was loaded into address 4007H, the first byte of DATA code from PROGRAM2 will be loaded into address 4008H. The linker will keep track of this for you.

"More files? yes **RETURN**"

"Object files? Mon_680XX **RETURN**"

"library files? **RETURN**"

"Load addresses: PROG,DATA,COMN,A5=0010000H,0000000H,0000000H,0000000H **RETURN**"

"More files? no **RETURN**"

You have specified the addresses where the code in your absolute file will be located. There are still some linker questions to be answered.

```
"list,xref,overlap_check,comp_db= on off on off (RETURN)"
```

These are the default selections for this question.

"list" will create a list of all symbols and their locations in the absolute file.

"xref" will list all symbols and their locations in the absolute file.

"overlap_check" will allow the display to give us a warning message if any of our code space gets overwritten during the load of the program. We should get one message when the COMN code from PROGRAM2 overwrites the COMN code from PROGRAM1.

"comp_db" is created when you answer "yes" to this question. This file is a data base containing information from all of the comp_sym files associated with the relocatables in the absolute file.

The last question in the linker configuration questions asks for an absolute file name.

```
"Absolute file name= PROGRAM (RETURN)"
```

When you execute this command, the development station links the three modules together, and you get the single error message on screen warning of the memory overwrite that occurred when the COMN code from program 2 overwrote the COMN code from program 1.

```
*****ERROR: Memory Overlap, files= PROGRAM2 0003500-0003507 PROGRAM1 00035
```

Naming the absolute file here can be a little misleading. You may think you named only one file. Actually you named three: an absolute file, a link_symbols file, and a linker_command file.

(1). The absolute file contains the machine-language code from all three program modules linked together.

(2). The link_symbols file contains the load plan for the absolute file.

(3). The linker_command file contains the command sequence you just went through. If you ever need to link these three program modules again in just the same way, you can enter "link PROGRAM (RETURN)" at the monitor softkey level, and the development station will enter all the above commands automatically.

If you would like to look at these commands one-by-one, press the following keys:

```
"link PROGRAM options edit (RETURN)"
```

Now press (RETURN) to step through the commands you entered.

This completes the link. The link load plan will create an absolute file and load all the code at the addresses you specified in the memory map at the beginning of this sequence. You still haven't told the machine whether those addresses are located in the emulation hardware or the target system hardware, and whether they are ROM or RAM. Don't worry, you'll enter those

Emulator/Analyzer 68000/68008
Preparing Your Program Modules

specifications later when you go through the emulation configuration questions in the next Chapter.

Before going into the emulation configuration part of this demonstration, it will be interesting to take a look at the link_symbols PROGRAM file you just created. Type in the following:

```
"copy PROGRAM :link_sym to LOADPLAN :listing RETURN"
```

```
"edit LOADPLAN RETURN"
```

This is the load plan you created using the linker.

```
START
1                               Page # 1
2
3 File = PROGRAM:YOURID:link_sym   Fri, 25 Jan 1985, 13:16
4
5 Record #   1   size =   26
6 Name record:
7 Linker is I68000:HP
8 Double word addresses
9 Checksum = EEFBH
10
11 Record #   2   size =   92
12 Global record:
13 MONITOR_MESSAGE 000105DCH          MONITOR_ENTRY. 00010000H
14 SPECIAL_ENTRY   000100C0H          MONITOR_CONTROL 00010658H
15 MONITOR_REGISTE 0001057EH          JSR_ENTRY       000102F0H
16 START           00002000H          RESET_ENTRY     00010320H
17 XFR_BUF         00010664H          MONITOR_6464XS 00010000H
18 ADDSUB          00002500H
19 Checksum = 1C74H
20
21 Record #   3   size =   44
22 Program record
23 Program is PROGRAM1:YOURID
24 Program load address is 2000H
25 Data load address is 4000H
26 Common load address is 3500H
27 Program is PROGRAM2:YOURID
28 Program load address is 2500H
29 Data load address is 4008H
30 Common load address is 3500H
31 Program is Mon_680XX:YOURID
32 Program load address is 00010000H
33 Data load address is 0000H
34 Common load address is 0000H
29 Checksum = E832H
36
37 Record #   4   size =  128
38 Module range record
39 0000 0000 0033 0000 0008 884D 6F6E 5F36      3   Mon_6
```



```

40 3830 3130 4850 382D 3135 1000 0080 0000 8010HP8-15
41 00BF 0000 0008 884D 6F6E 5F36 3830 3130      Mon_680XX
42 4850 382D 3135 1000 2000 0000 201B 0000  HP8-15
43 0001 9850 524F 4752 414D 3120 524F 4745      PROGRAM1 YOUR
44 5243 0000 2500 0000 2513 0000 0005 9850  ID % %      P
45 524F 4752 414D 3220 524F 4745 5243 0000  PROGRAM2 YOURID
46 3500 0000 3507 0000 0007 9850 524F 4752  5  5      PROGR
47 414D 3220 524F 4745 5243 0000 3500 0000  AM2 YOURID  5
48 3507 0000 0003 9850 524F 4752 414D 3120  5      PROGRAM1
49 524F 4745 5243 0000 4000 0000 4007 0000  YOURID @ @
50 0002 9850 524F 4752 414D 3120 524F 4745      PROGRAM1 YOUR '
51 5243 0000 4008 0000 4008 0000 0006 9850  ID @ @      P
52 524F 4745 414D 3220 524F 4745 5243 0000  PROGRAM2 YOURID
53 0000 0001 0763 0001 0009 884D 6F6E 5F36      c      Mon_6
54 3830 3130 4850 382D 3135 1000 51A9      80XXHP8-15 q
55
56 End of file after record #  4
END

```

Figure 4-4. Example Link_Sym Listing

Press:

"end **RETURN**"

You should now be at the command line level.

The LOADPLAN listing contains four records. Record #1 names the microprocessor on which this will run, and specifies double-word addressing. Record #2 shows all of the symbols in the set of linked modules (the monitor program plus the two program modules you linked together, along with their assigned addresses). Record #3 shows the starting addresses of the program, data, and common areas in each of the modules. Record #4 shows the machine code of the entire program.

WHY SO MANY FILES HAVE THE SAME NAME

Several files are created each time you use the assembler/linker or configure the emulator. You assign the names of your program module, listfiles, and assembler, linker, and emulator output. However, the assembler and linker will have several files with the same name (they are uniquely identified by file type designation i.e., FILENAME:filetype--FILENAME:absolute, FILENAME:source)

You input:

Program Module Source File (:source)

You get from the assembler:

Relocatable Object Code File (:reloc)
Symbol Cross-reference List (:asmb_sym)
List file (:listing)

You get from the linker:

Symbol Cross-reference List (:link_sym)
Load Map Listing (:listing)
Link Command File (:link_com)
Absolute code (:absolute)

You get from answering the emulation configuration questions:

Emulator Command File (:emul_com)

For more information on the assembler/linker and assembler and linker output files, refer to the Assembler/Linker Reference Manual and the 680XX Assembler/Linker Supplement Manual.

WHERE TO LOCATE THE MONITOR

An example of an emulation memory map is shown in figure 4-5. The target system is assumed to have 4K bytes of RAM starting at 3000H. It will be assumed that the target system RAM is usable. NOTE: This example is for the 68000; however, the method used is the same for the 68008, just change the 4K values to 256 bytes.

Emulation memory blocks: available= 11 mapped= 5 size= 4K bytes

entry	range	type	blocks	entry	range	type	blocks
1	0-	FFF RAM/EM/ALL	000-000				
2	2000-	2FFF RAM/EM/ALL	001-001				
3	3000-	3FFF RAM/TR/ALL	-				
4	4000-	4FFF RAM/EM/ALL	004-003				
5	10000-	10FFF RAM/EM/ALL	004-004				

Figure 4-5. Example Memory Map

It appears that any block between 5000H and 9FFFH would be a good place to load the emulation monitor. The monitor in this example begins at 10000H. If this is not possible due to memory images or other hardware considerations, then blocks of the user RAM or EPROM will have to be allocated for use by the monitor. Let us assume there are no hardware restrictions in this example. The starting location for the emulation monitor program area is chosen as 10000H (from figure 4-4). Examining record #1 of the emulation monitor relocatable file (see figure 4-2) will show that 764H bytes are required for the program area.

Since memory is allocated in 4K byte blocks, this number, 764H, should be rounded up to the next multiple of 4K minus one (FFFH). Thus, giving an address range of 10000H to 10FFFH for the monitor.

The emulation memory map shown in figure 4-5 was generated using the above considerations. Entry one protects the exception lookup table (Note that this may not be necessary in your system if you have developed your own exception table vector table). Entry two is allocated as emulation ROM and is where your programs will reside in memory. Entry three is the COMN area of your program and therefore is designated as RAM. Entry four is the DATA area of the program and will reside in target system memory; the map, therefore, shows target system RAM. The final entry, number 5, shows the allocated space for the monitor program. This is shown as emulation RAM.

NOTE

Notice the break in the map from 4FFFH to 9FFFH. This un-mapped area bounds the monitor and is mapped as guarded. Although it does not specify guarded on the map, it is recommended, but not necessary, that you specify all un-mapped memory as guarded (non-existent). These guarded areas protect the monitor program from being written over by a run-away program. **Don't forget that the monitor must be mapped in emulation RAM**, this is why the protection is needed.

NOTE

Another method of protecting your monitor program would involve changing the monitor program such that you could map the program portion of the monitor to emulation ROM, and the data area of the program to emulation RAM.

NOTES

Chapter 5

ANSWERING EMULATION CONFIGURATION QUESTIONS

OVERVIEW

This chapter will:

- Explain each of the emulation configuration questions.
- Describe how to configure the emulator for your 680XX target system.
- Explain how to map the memory of your 680XX system to a combination of emulation and target system memory sources.

GENERAL

In this chapter, whenever 680XX is mentioned (even as part of a filename) both the 68000 and the 68008 are assumed unless there are differences between the two processors. In that case, the differences will be noted. When the name 680XX appears in any program listings, just insert the appropriate name of your processor.

INTRODUCTION

The purpose of the emulation configuration questions is to define your 680XX target system to the emulator. The emulator needs to know what resources are available on your target system (clock and memory). Since the emulation memory boards provide memory to be used if your 680XX system memory is not yet available, you must define the mapping of memory resources. In addition you must define the mode of operation (real-time or nonreal-time) for the emulator and whether or not you want to be notified of attempts to write to ROM.

The emulation configuration questions must be answered even if you are running out-of-circuit emulation (developing/debugging software without your target system).

To answer the configuration questions you need to know the answers to the following questions:

- Is a clock available on your target system?
- What memory is available in your target system?



When the emulator detects a guarded memory access or other illegal condition, or when you request an access to memory which causes the emulator to break into the monitor, the emulator stops executing your code and enters the monitor. Thus, if you have circuitry that can be damaged because the emulator is not executing code, you should exercise special caution. For example, you should configure the emulator to restrict to real-time runs, and you should not break into the monitor.

In order to become familiar with the emulation and analysis user interface and feature set, it is recommended that a short program be written and executed with the emulation probe disconnected from your target system, or "out of circuit". A simple program that adds the contents of processor registers or memory locations (i.e.; the one used in chapter 4) will provide a good example.

ADDRESS CONVENTIONS

You will need to understand the address conventions for the 680XX processor in order to map the emulation/user memory later in this chapter. The physical address is used during configuration setup for specifying the memory map. The physical address takes the form XXXXXB, where "B" is the number base (H = hexadecimal, B = binary, Q or O = octal, D = decimal). Leading zeros are only required where the leading character is a non-numeric hexadecimal character. Physical addresses can be 32 bits long; however, the 68000 only uses the least significant 24 bits on the address bus. The 68008 only uses the least significant 20 bits.

ACCESSING THE CONFIGURATION QUESTIONS

To begin emulation using your example programs (refer to chapter 4), enter the "measurement_system" command, for multi-module systems, or the "emulate" command, for single-module systems. The syntax for each form is described in more detail in the following paragraph. The command initiates a series of questions that you will answer to configure the emulator for your particular application. Each question is provided with a default answer that you can enter "as is" by pressing the **(RETURN)** key, or that you can modify by using the softkeys or keyboard entries. The meaning of these questions and answers is described in detail later in this chapter.

Measurement System/Emulate Command Syntax

You can access the measurement system level of softkeys through either one of two ways, depending on your mainframe/system disc configuration. If more than one module is present in the card cage, or if the Model 64330 High Level Software Analyzer software is resident on the system disc, the command "meas_sys" appears at the first level of softkeys. If your emulator is the only module present, then "emulate" is present at the first level of softkeys. The "measurement_system" command syntax and the "emulate" command syntax are given on the following pages. The measurement_system command syntax is given twice; once for reentering an emulation session that you previously set up and is running (options continue), and once for beginning a new session for which no measurement parameters have been set up, but you have an emulation configuration command file already made. If you enter "measurement_system" or "emulate" without *options continue*, and you do not enter an emulation configuration command file (<CMD_FILE>) at the em680XX_S level, you must answer the configuration questions.

Configuring/Reconfiguring the Emulation Pod

You configure or re-configure the emulation pod for an emulation session by answering the emulator pod configuration questions. The emulator pod configuration questions are given later in this chapter. In addition, if the emulator is to be used with your target system, you must install the microprocessor connector in your target system (refer to Appendix G2 for instructions on installing the microprocessor connector into your target system).

NOTE

Before we begin the emulation configuration questions, it is important that you are aware of one of the questions you will be asked to answer.

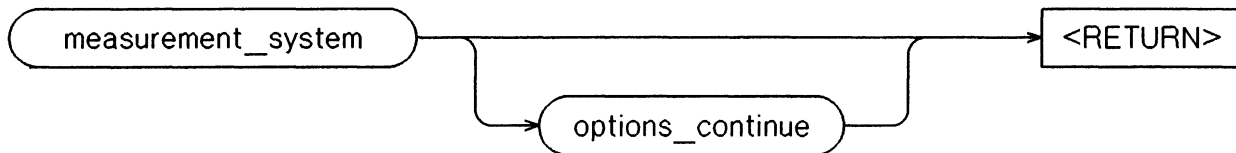
The question is "Trap number for software break (0..0FH) ?"

This emulator function is not available unless the monitor program is prepared using the procedure described in Chapter 8 in the section "Modifying the Monitor to use Software Breakpoints." This is a feature that will be used during software debugging and you probably will not need to use it now. Note that it is not necessary to make this modification in order to use your emulator, the software breakpoint function will just not be available to you.

— measurement_system —

For multi-module systems (using options continue):

SYNTAX



FUNCTION

The command "measurement_system" with "options continue" causes system operation to enter the measurement system monitor. The measurement system monitor coordinates and displays the interaction between the modules present and, in multi-module systems, controls entry to and exit from the individual modules of the system. Once in the monitor program, the emulator can be accessed by issuing the command "em680XX_S" , where "S" is the slot number of the emulation control board. The choice is made through the softkeys.

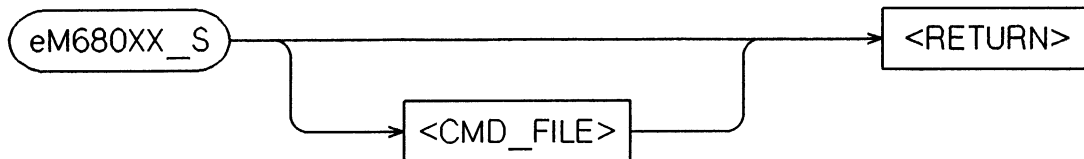
The "continue" option allows reentry to a previous session without disrupting a measurement in progress. If "continue" is not specified, all measurement system modules will be reset to their default configuration and any activity stopped. A "continue" is not possible under any of the following conditions:

- a. Power has been cycled or the station reset by shift/reset.
- b. Performance Verification (option_test) has been initiated.
- c. The last session was exited by reset/reset.
- d. The measurement system configuration file is not present.

measurement_system

For multi-module systems (without using options continue):

SYNTAX



where "S" is the slot number of the emulator control board, and "<CMD_FILE>" is an optional emulation command file.

Default Value

[<CMD_FILE>] The last specified command file.

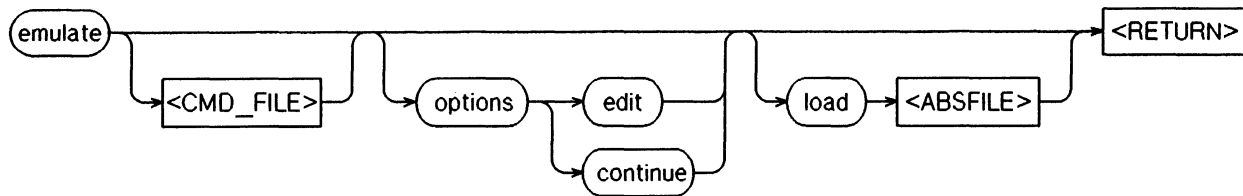
FUNCTION

The emulate command, when issued from the measurement system monitor program, transfers action to the monitor program for the specified emulator. When no command file exists, or there is a conflict between the specified command file and the previous configuration, the emulation configuration questions are initiated and either a new command file is generated, or the specified file is edited.

emulate

For single module systems:

SYNTAX



Examples:

```
emulate  
emulate LOOP  
emulate LOOP load MUCH
```

FUNCTION

When no options are selected, emulation configuration is initiated and a new command file is constructed. If <CMD_FILE> is specified, an emulation session is initiated using the configuration specified by the command file. When a command file is specified, it is possible to continue a previous session. Or, if an altered configuration is needed, the edit option can be selected, allowing a new configuration by editing the previous one. Another option is specifying an absolute file to be loaded into emulation memory upon entry to the session.

ANSWERING THE CONFIGURATION QUESTIONS

The emulation configuration questions are used to prepare the emulation hardware and software for a specific application. Each question is displayed along with a default response, with additional options here shown in parentheses. Selecting the default responses will set up the emulation configuration that is easiest to use in most applications. The default response can be selected by pressing the **(RETURN)** key; or another response can be selected by the appropriate soft key, or by typing in a suitable response. If an emulation command file, which you previously made, is used to configure the emulator, the default responses will be the responses already in the command file.

Using the example programs given in chapter 4, all of the default answers will be sufficient except during memory mapping. When the memory map is displayed, some portion of memory must be assigned. For this example the commands "2000H thru 2FFFH emulation ROM, 3500H thru 35FFFH emulation RAM, 4000H thru 4FFFH emulation RAM, and 10000H emulation RAM" are required because of the way the linker map was made (refer to figure 5-1) , i.e. the example program and the emulation monitor program have been loaded in this area of memory. The mapping section of configuration is exited with the "end" command. The last question asked during configuration is "Command file name?". If a name is given, a file of type "emul_com" will be created. This file is similar in function to the link_com file and is described later in this chapter. For the example above a blank answer is sufficient but a file name may be entered.

The configuration questions deal with the following items:

- a. Selecting the card slot
- b. Selecting the clock
- c. Selecting real-time/nonreal-time runs
- d. Modifying memory configuration
- e. Mapping memory
- f. Configuring simulated I/O
- g. Configuring the emulator pod
- h. Configuring for interactive measurements
- i. Naming your emulation configuration command file

Selecting the Card Slot

For single-module systems, the card slot questions will not be asked. The first question asked in that case will concern the clock selection. For multi-module systems, however, it will be necessary to identify the slots of the memory controller and internal analysis cards associated with the emulator being used. The following questions will appear:

Slot number of memory controller card? 0..9

The default answer will be the slot number of one of the memory controllers, or the slot number specified in a command file.

Slot number of analysis card? 0..9 (none)

The default answer will be the slot number of one of the analysis cards, or the slot number specified in a command file. It is possible to emulate without the benefit of an analysis card by selecting "none". None of the functions, however, that require an analysis card will be usable. The functions requiring an analysis card are "run until", "step", and "trace".

Selecting the Clock

The 680XX emulation processor can be driven by the internal 10 MHz clock or by an external clock from your target system.

Microprocessor clock source? internal (external)

internal - The internal clock is normally used when the emulator is to be run out-of-circuit, i.e., with no target system.

When the internal clock is selected, it is not necessary to disable your target system clock source.

external - An external clock is normally used when the emulator is connected to your target system. Your target system clock must meet the specifications for the clock input to the microprocessor in order to be reliably used with the 680XX emulator.

Maximum rate is 12.5 MHz. for the 68000.

Maximum rate is 10.0 MHz. for the 68008.

Microprocessor clock rate greater than 10MHz? no(yes) ***68000 only***

This question will only be asked if you answer "external" for the question "Microprocessor clock source? internal (external)."

no An answer of no means that no wait states are required.

yes If you answer yes, (68000 only) then one wait state is required by the emulation system.

Selecting Real-Time/Nonreal-Time Runs

The question listed below provides an opportunity to restrict the emulator to real-time program execution. "Real-time" in this case is not based on whether wait states are inserted or not, since none are needed by the 680XX emulator. Instead, real-time refers to the continuous execution of the your 680XX program without interference from the development system except as instructed by the operator.

Restrict to real-time runs? no (yes)

no - If runs are not restricted to real time, the emulation software will perform all commands upon request, and detect entry to the emulation monitor at any time. To allow this to happen, the emulation system must assert DMA controls and breaks (INT7) to the 680XX processor at any time. The DMA controls are used to allow the emulation software to access the 680XX memory mapped as emulation memory. Emulation memory is accessed when a command to display, list, modify, load, or store emulation memory is processed, or when the emulation software is communicating with the emulation monitor program via a control block in the emulation monitor data area. The DMA controls to the 680XX last approximately 34 to 40 usec and occur at 500 msec intervals.

In the nonreal-time mode, the emulation system will force the emulation monitor program to be entered whenever a command that requires access to the 680XX registers, target system I/O, or target system memory is processed. If your code was executing at the time the request was processed, the emulation system will generate a break to force entry to the emulation monitor. After the necessary information is obtained through the monitor, your code resumes execution.

Both the assertion of the 680XX DMA controls and breaks (NMI) extend the execution time of your code. If your system is dependent on execution time, it may be necessary to restrict to real-time.

yes - Restricting to real-time will inhibit the emulation system from extending the execution time of your code. While your code is executing, all emulation commands that require 680XX DMA controls or conditional breaks are not allowed. (A conditional break is a special type of break that is generated by the emulation software that momentarily diverts execution to the emulation monitor and then resumes execution of your code.) These restricted commands are listed below:

```
display memory      list memory        modify memory
display registers   list registers     modify software_breakpoints

load memory
store memory
```

Following reset, the emulator will remain in the real-time mode until a break from one of five sources is detected by the emulation software. The five sources are: a memory break due to a write to ROM or an access to guarded memory, an analysis break from a "trace" command that included a "break_on" specification or a "run_until" command, a "break" command, a "run from" command, or a "step" command. Once a break is detected, the emulation system will generate DMA controls to the 680XX processor to allow communication with the emulation monitor. Once the emulation monitor is detected, the above commands will no longer be restricted. The emulation system will return to the real-time mode when execution is returned to your code with a "run" command.

Selecting/Modifying a Memory Configuration

When you begin your initial emulation session you must configure (map) the memory space you will be using. Your decisions as to where to map your memory are based on the length and features of your target system program(s). As you progress with your program development, your memory map requirements will probably change. Rather than start a configuration session from the beginning, you have the option of modifying your present configuration. You can then either keep the same configuration file name (writing over the current file) or assign the new configuration a new file name. If you assign a new name to the file and you are using a command file to enter your emulation session, remember to modify your command file to change the name of your emulation configuration file (refer to chapter 4 for more information concerning command files).

The following discussion deals with modifying a current configuration first (one question that is in addition to starting a new configuration), then deals with starting a new configuration.

MODIFYING A CURRENT CONFIGURATION. If the emulator has already been configured (configuration questions have been answered) and the emulation session has been entered, you can modify the configuration. You do this by pressing the *modify config* softkeys while in the emulation session. You will get the questions "Microprocessor clock source" and "Restrict to real-time runs" discussed above, and then you will be asked the following question:

Modify memory configuration? no (yes)

This question provides the opportunity to review and modify the memory configuration stored in the emulation command file.

If the emulation configuration question/answer sequence was entered without specifying an emulation command file, no memory map for the current session exists and the response to this question is assumed to be "yes". The memory configuration questions are then presented.

- yes - Allows the parameters for memory usage to be altered. Parameters include number of significant address bits, break on a write to ROM, and memory mapping. The processor will be reset, and the configuration questions will be presented one at a time along with their current default values. Each default response can be entered as listed by pressing **RETURN** or modified as necessary to apply to the current emulation session and then entered using the **RETURN** key. The questions asked are given and explained in the next paragraph, "Selecting the Memory Configuration".

- no - Allows you to skip over memory usage questions if it is not desired to change memory usage. If "no" is specified, the next question will concern simulated I/O configuration. A "no" response causes the memory to be configured as specified by the current emulation command file.

SELECTING THE MEMORY CONFIGURATION. The following questions will now be asked whether you are entering a session for the first time or you are modifying a configuration.

Number of significant address bits? 24 (<#BITS>) *68000

Number of significant address bits? 20 (<#BITS>) *68008

The number of significant address bits determines the maximum number of bits that the mapper will respond to. All address bits above the specified value will be treated as "don't cares". For the 68000, a value of 24 allows all of the processor address lines to be used with the mapper.

For the 68008, a value of 20 allows all of the processor address lines to be used with the mapper.

Break processor on write to ROM? yes (no)

When "yes" is entered, a break will occur when the processor attempts to do a write to an address location mapped as ROM. If an illegal write is attempted with emulation memory, the contents of that memory location will not be altered. Illegal references to target memory may, or may not, alter memory contents, depending on your target system hardware.

Mapping Memory

In order to perform emulation, the memory mapper must be properly programmed to correspond to emulation memory and target system memory resources. The memory mapper allows you to divide the processor's address space into a number of blocks that can be individually assigned any one of five descriptors: emulation RAM, emulation ROM, target RAM, target ROM, or guarded memory. During emulation, the mapper monitors the address bus and provides the descriptor for the address present at any given time. This information is used by the emulator hardware to control the flow of data and code between the emulation processor and the memory resources.

The responses to the memory configuration questions are used to configure the memory mapper.

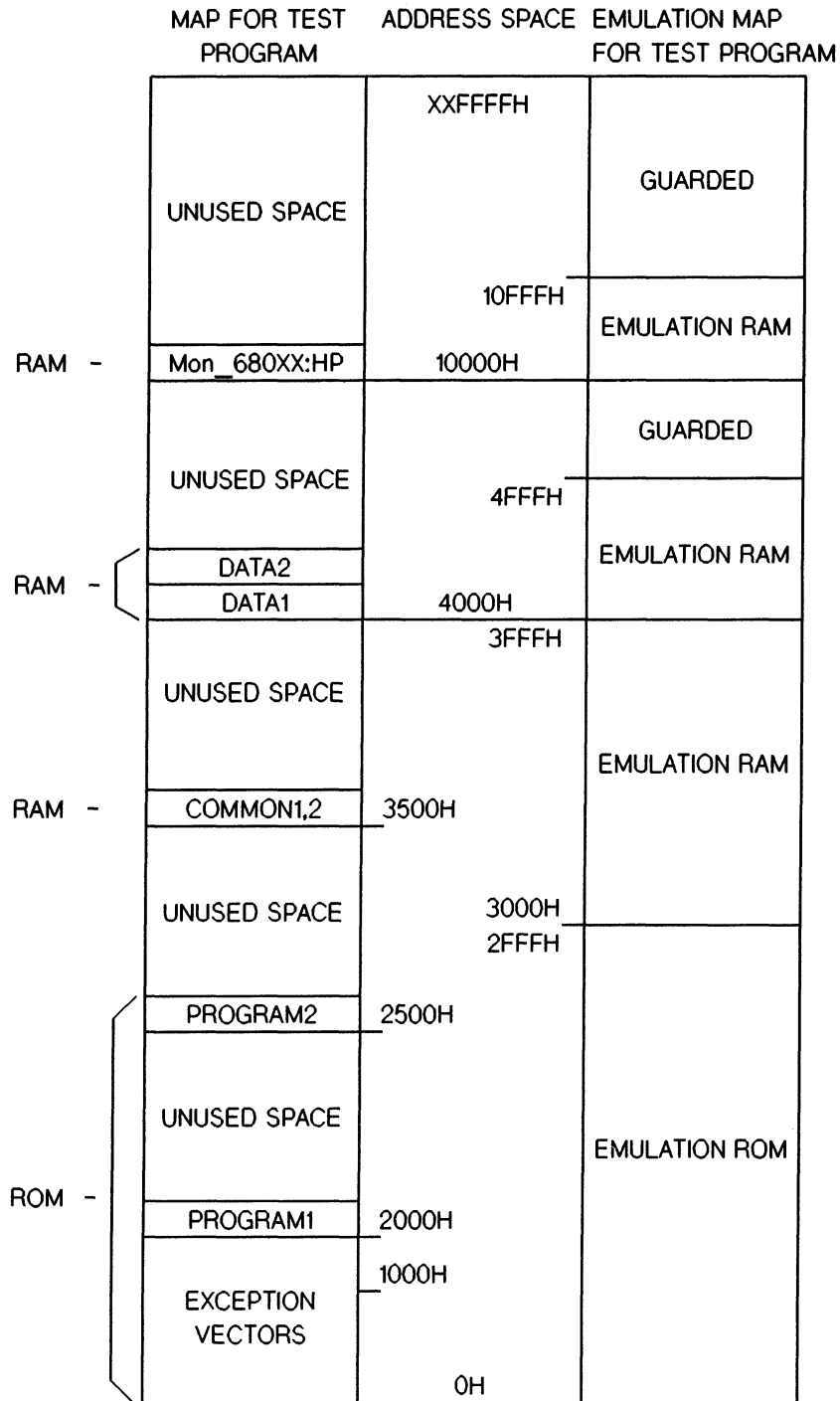


Figure 5-1. Emulation Map For Test Program

MEMORY MAP DEFINITION. The memory map partitions the processor address range into one of the five possible configurations. The map is composed of from 1 to 32 address range definition entries and a memory default. Each entry defines a particular address range as one of the possible memory types. Any address range not defined by an entry is mapped to the memory default.

Entries need not be an integral multiple of the block size but once the mapper software processes the inputs, the bounds will be rounded to integral multiples of the block size. The final boundaries will include all of the memory space specified, plus the remainder of any blocks which were partially specified. The remaining parts of the address range, not covered by an entry, are mapped to the memory default.

When target memory is specified for a given address range, all memory cycles using that address range will access the target system and will function as RAM. All memory load and display operations for your target system are done via the emulation monitor program.

Emulation memory can be specified as either ROM or RAM. As with target memory ROM, write attempts to emulation ROM can generate a break, if desired. Additionally, any write attempt to emulation ROM will not change the contents of that memory location. All emulation memory is displayed and loaded directly by the emulation software via the memory port assigned to the mainframe processor.

Guarded memory is memory that you cannot access. One reason may be that there is a memory shadow from another memory block (due to partial address decoding in your target system memory) in the same address space, or the memory in that section is either not developed or not available to your system. The block of memory may not even exist.

OVERLAY ADDRESSES When making a memory map, you can enter "overlay" addresses in emulation memory hardware blocks. That means, you can have a single block behave as if it were two different blocks, each responding to its own set of addresses.

For example, you can have block 001 responding to addresses 1000H through 1FFFH, and 8000H through 8FFFH. The memory mapper will do this for you. Then as the processor places address 101AH on the address bus, it will fetch the same word as when it places 801AH on the address bus. In this way, you save hardware memory during development. You may want to have duplicate code in these two address areas of your target system. During development, the emulator will allow the system to emulate having duplicate code at two address blocks simply by allowing the one block to respond to two sets of addresses. Overlay addressing is discussed in further detail later in this chapter.

EMULATION MONITOR PROGRAM MEMORY REQUIREMENTS. You will need to know certain information about the emulation monitor program (delivered as part of your software package) prior to linking the emulation monitor program. Such as; the monitor program must be mapped to emulation ram. Chapter 8 contains a detailed description of the monitor itself, including the memory requirements for the program. Refer to the paragraph entitled "Emulation Monitor Memory Requirements" in chapter 8 for a full description of the emulation monitor memory requirements.

DISPLAY ORGANIZATION. The memory map display is shown in figure 5-2. The top line of the display indicates the number of emulation memory blocks available for mapping, the number of blocks currently mapped, and the size of the blocks. As mapper entries are made, the available and mapped block numbers will be updated to reflect the current values.

The number of available blocks depends upon the amount of emulation memory in the card cage. Table 5-1 provides a quick reference of the number of blocks available as a function of the total available memory.

Each emulation memory card can contain up to four, 32K byte rows of high speed RAM and up to eight fully loaded memory cards can be installed in the card cage.

To map the sample program according to the figure 5-1 load map, make the following keyboard entries:

```
"0000H thru 0FFFH emulation rom (RETURN)"  
"2000H thru 2FFFH emulation rom (RETURN)"  
"3500H thru 35FFFH emulation ram (RETURN)"  
"4000H thru 4FFFH emulation ram (RETURN)"  
"10000H emulation ram (RETURN)"  
"default guarded (RETURN)"
```

The display should show:

```
Emulation memory blocks: available= XX   mapped=  X   size=  XX bytes  
entry  range      type   blocks entry  range      type   blocks  
1      0-    FFF ROM/EMUL  000-000 :  
2     2000- 2FFF ROM/EMUL  001-001 :  
3     3000- 3FFF RAM/EMUL  002-002 :  
4     4000- 4FFF RAM/EMUL  003-003 :  
5    10000- 10FFF RAM/EMUL  005-004 :  
  
STATUS: Mapping emulation memory, default blocks: guarded_____11:11  
  
default  guarded  
  
<ADDRESS> _default_ _delete_ _____ _print_ __end__
```

Figure 5-2. Memory Map Display

The center portion of the display consists of up to 32 entries arranged in two columns of 16 each. If the emulation session is being entered using an emulation command file, the entries which were previously defined will be displayed.

However, if emulation is being entered without a command file designation, the map will be blank. In this case, a new map must be set up before the mapping session is terminated. Any attempt to end the session with a blank memory map will cause an error message to be displayed.

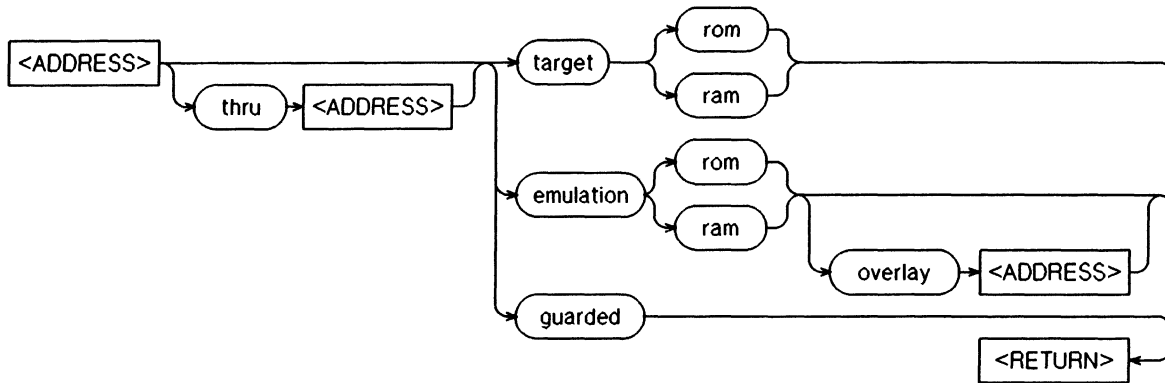
The softkey labels on the mapping display identify the options available during the mapping session. You can specify individual map blocks, define the default state, delete any or all of the currently defined blocks, copy the current map display to a printer, or end the map definition session. The implementation of these options is described on the following pages.

Table 5-1. Number of Mapper Blocks vs. Available Memory

Memory board model number	# of bytes per card	#of 4K byte blocks per card	#of 256 byte blocks per card
		68000	68008
HP 64161A	128K	32	512
HP 64162A	64K	16	256
HP 64163A	32K	8	128
HP 64152A	32K	8	128
HP 64153A	16K	4	64
HP 64154A	8K	2	32

ENTERING MAPPER BLOCKS. All mapper entries consist of an address or address range and a descriptor which assigns the type of memory accessed by the specified addresses. Once the desired address or address range has been typed from the keyboard, the list of available descriptors will appear as softkeys.

One of five entry potential descriptors must be selected for the memory area defined by the address designation. The descriptors are target ROM, target RAM, emulation ROM, emulation RAM, and guarded space. In addition, if an entry is mapped to emulation space, it is possible to overlay the newly defined area over a previously defined emulation memory area. The overlay procedure is discussed in detail in the next section.



where:

- target designates memory supplied by your target system. Mapping an address range to target space requires no emulation memory. Therefore, the number of available memory blocks listed at the top of the screen will not change when target space is specified.
- emulation designates memory supplied by the emulation system. When an entry specifying emulation memory is made, the number of available blocks of emulation memory will be decreased by the appropriate number of blocks.
- guarded designates an address range which is not expected to be accessed. Any processor access to a location within such a range will result in a Break of the program execution. No emulation memory is used when an address range is specified as "guarded".
- rom designates memory which can detect the occurrence of write cycles and cannot be modified by the processor. Emulation memory that is actually RAM but is mapped as ROM will perform as ROM during emulation.
- ram designates memory which can be read from or written to without restriction.
- <ADDRESS> defines a bit pattern of up to 20 bits which specifies a particular location in memory. That bit pattern can be represented by a binary, octal, hexadecimal, or decimal number.

The first <ADDRESS> of a range specification should be the starting address of a block boundary. If an address within the interior of a memory block area is entered, the system will convert this address to the starting address of the block prior to its mapping. Leading zeros may be deleted as long as the most significant digit listed is numeric.

If a single address is specified rather than a range of addresses, the entry will map only the block containing that address. In this case, the "thru <ADDRESS>" portion of the syntax need not be entered; only a single address and its descriptor are entered.

EMULATION MEMORY OVERLAYS. When making a memory map, you can enter "overlay" addresses in emulation memory hardware blocks. That means, you can have a single block behave as if it were two different blocks, each responding to its own set of addresses.

For example, you can have block 000 responding to addresses 1000H through 1FFFH, and 8000H through 8FFFH. The memory mapper will do this for you. Then as the processor places address 101AH on the address bus, it will fetch the same word as when it places 801AH on the address bus. In this way, you save hardware memory during development. You may want to have duplicate code in these two address areas of your target system. During development, the emulator will allow the system to emulate having duplicate code at two address blocks simply by allowing the one block to respond to two sets of addresses. Perhaps an example will make this concept more clear.

To see how these entries are made, type in the following on the map:

1000H *thru* 1FFFH *emulation ram* (RETURN)

8000H *thru* 8FFFH *emulation ram overlay* 1000H (RETURN)

```
Emulation memory blocks: available= 15    mapped= 1    size= 4K bytes
entry  range      type      blocks entry  range      type      blocks
1     1000- 1FFF RAM/EMUL  000-000 :
2     8000- 8FFF RAM/EMUL  000-000 :
```

STATUS: Mapping emulation memory, default blocks: guarded_____ 11:11

8000H thru 8FFFH emulation ram overlay 1000H

<ADDRESS> _default_ _delete_ _____ _print_ _end_

Figure 5-3. Sample Overlay Mapping #1

The way your map shows this overlay is in the "blocks" column. You'll see that block 000-000 is assigned to serve both address entries.

Your target system may use ROM space in the address range 1000H through 1FFFH, and RAM space in the address range 8000H through 8FFFH; the emulator can handle that. Simply specify ROM when entering the lower address and RAM when entering the upper address block. The

emulator will only allow reads when the processor is in the lower address range and will allow both reads and writes when in the upper address range.

Of course, there is a risk involved with this arrangement. If your program writes some new information when in the RAM address range, that new information will be read as ROM information when in the ROM address range.

```
Emulation memory blocks: available= 9   mapped= 7   size= 4K bytes
entry  range      type      blocks  entry  range      type      blocks
1      0-    FFF RAM/EMUL  000-000 :
2     1000- 1FFF ROM/EMUL  001-001 :
3     2000- 2FFF RAM/EMUL  002-002 :
4     3000- 3FFF  GUARDED    -      :
5     4000- 4FFF RAM/EMUL  001-001 : <-- The same block is used
6     5000- 5FFF RAM/EMUL  002-002 : for two address ranges.
7     6000- 9FFF RAM/EMUL  003-006 : When it responds to
8    10000- 13FFF RAM/EMUL  003-006 : addresses 1000H-1FFFH,
9    20000- 20FFF ROM/EMUL  001-001 : it will emulate ROM space.
                                     When it responds to
                                     addresses 4000-4FFFH,
                                     it will emulate RAM space.
```

```
STATUS: Mapping emulation memory, default blocks: target/ram_____11:11
```

```
20000H thru 20FFFH emulation rom overlay 1000H
```

```
<ADDRESS> _default_ _delete_ _____ _print_ _end_
```

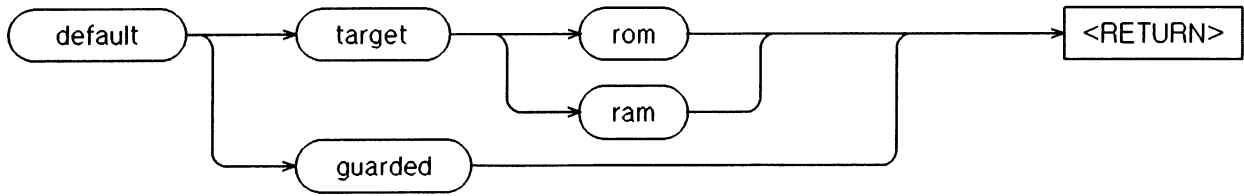
Figure 5-4. Sample Overlay Mapping #2

In the above display, "overlay" addressing was used. Block 001 will respond to three different ranges of addresses. Block 002 will respond to two different ranges of addresses. Blocks 003 through 006 will respond to two different ranges of addresses.

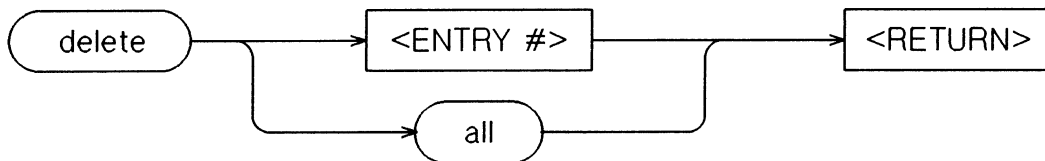
Addresses 3000H through 3FFFH were assigned to act as guarded memory space. No hardware block of addresses is required for a guarded memory assignment. This is memory that does not exist. The value in specifying guarded memory is that the emulator will report an error if your program ever tries to write to or read from any space designated "guarded". If an error, such as an attempt to overlay onto an unidentified range or a range not mapped to emulation memory, is made while implementing the overlay function, an error message will notify the operator of the error.

SETTING THE DEFAULT. Any address ranges which are unmapped when the mapping session is terminated are assigned the memory descriptor specified as the default. The default descriptor can be set up to be target RAM, target ROM, or guarded by using the default command.

The syntax for the default command is shown below:



DELETING BLOCKS. Any one or all of the memory map entries can be removed by using the delete command. The syntax for the delete command is listed below:



ENDING THE MAPPING SESSION. The memory map configuration session is exited by pressing the *end* softkey followed by **RETURN**. If an attempt is made to end the mapping session with a blank memory map, an error message will be displayed.

Configuring Simulated I/O

Simulated I/O is not available in real-time mode. If this condition is true, the simulated I/O question will not be asked. Available memory for simulated I/O is determined by the number of measurement system modules present. If the maximum number of measurement system modules (4) is present, then simulated I/O memory is not available and the simulated I/O configuration is not presented. If three or less modules are present, then the memory available is as follows:

one measurement system module, available memory is 768 words.

two measurement system modules, available memory is 512 words.

three measurement system modules, available memory is 256 words.

Available memory is allocated during the actual emulation when an open command is requested for simulated I/O devices. Some devices do not require additional memory. The simulated I/O devices that require memory are: display, printer, RS232, and disc files.

Each device, except RS232, requires a minimum of 145 words of memory space. RS232 requires 128 words of memory space for the read buffer, and 128 words of memory space for the write buffer.

A maximum of five devices, not including RS232, may be open at one time for a single module measurement system or 768 words available. With RS232 read and write buffer operation, another three devices may be opened.

Emulator/Analyzer 68000/68008
Emulation Configuration

A maximum of three devices, not including RS232, may be open at one time for a dual module measurement system or 512 words available. With RS232 read and write buffer operations, only one other device may be opened.

A maximum of one device, not including RS232, may be open at one time for a triple module measurement system or 256 words available. With RS232 operation, only one read buffer and one write buffer may be open, but no other devices may be opened.

Available memory is deallocated during actual emulation when a close command is requested for the simulated I/O device. Deallocated memory can then be allocated to some other simulated I/O device.

If simulated I/O devices try to allocate more memory than is available, an error return of 9 (request not allowed) is returned to the simulated I/O device control address.

When there is available memory for simulated I/O, the command line displays the following question and answer:

Modify simulated I/O? no (yes)

The status line shows:

STATUS: Simulated I/O assignment

Answering yes to "modify simulated I/O?" will allow modification to all available simulated I/O devices. The simulated I/O devices are: display, printer, RS232, keyboard, and up to six disc files.

Questions for a control address for each device are then asked. If a reply of blank is made, then that device is not used. The control address may be specified for a maximum width of 32 bits. As each question is answered the results are displayed.

The simulated I/O questions are:

- a. **Display control address?**
- b. **Printer control address?**
- c. **RS232 control address?**
- d. **Keyboard control address?**

Each unit is identified with a physical address.

Next the command line displays:

Modify simulated disc files? no (yes)

Answering "no" bypasses any modification to simulated disc files I/O. Answering "yes" allows modification to simulated disc files.

The disc file simulated I/O questions are:

File 1 name?
File 1 control address?

File 2 name?
File 2 control address?

File 3 name?
File 3 control address?

File 4 name?
File 4 control address?

File 5 name?
File 5 control address?

File 6 name?
File 6 control address?

A blank file name disables simulated I/O for the specified file number.

Configuring the Emulator Pod

The following questions are required to be answered in order to configure (or re-configure) the emulator pod.

Reconfigure emulator pod? no (yes)

The emulator pod reconfiguration can be skipped if the default conditions listed below are appropriate for your target system.

Interlock emulation memory DTACK with user DTACK	- disabled
Enable Bus Error on emulation memory accesses	- disabled
Enable emulator processor interrupts	- enabled
Enable emulator DMA transfers	- enabled
Enable DMA transfers to emulation memory	- disabled
Enable tracing of DMA memory transfers	- disabled
Enable tracing of DMA tags	- disabled
Trap number for software breakpoint	- 0FH
Enable emulator use of INT7	- enabled
Enable INT7 sharing	- enabled
Enable tristate delay on halts	- disabled

Interlock emulation memory DTACK with user DTACK? no (yes)

If you answer no to this question, the processor will use the target DTACK input when accessing target memory, but use the emulator generated DTACK exclusively when accessing emulation memory. If you answer yes, the processor uses the target DTACK input exclusively when accessing target memory, but waits for both the target DTACK and the emulator generated DTACK

when accessing emulation memory. This feature can be used to keep the emulator synchronized to the target system even while accessing emulation memory.

NOTE

Please note that if interlocking target and emulation memory DTACK is desired, the DTACK switch on the pod should NOT be closed and the plug-in lead should NOT be connected to the target system hardware. Interconnecting the signals will OR them, and Interlocking the signals will AND them.

Enable Bus Error on emulation memory accesses ? no (yes)

If the answer to this question is no, the processor will ignore the BERR while accessing emulation memory. It will, however, respond normally while target memory is accessed. If the answer is yes, the processor will respond to the BERR input even while accessing emulation memory.

Enable emulator processor interrupts ? yes (no)

If you answer this question yes, the processor will respond normally to interrupts. If the answer to the question is no, the processor interrupts (all levels) will be masked off by hardware in the emulation pod.

Enable emulator DMA transfers ? yes (no)

If you answer this emulation configuration question yes, the processor will respond normally to the assertion of the BR (Bus Request) and BGACK (68000 Only) inputs. If the answer to this question is no, the processor will ignore the BR and BGACK inputs and will not respond with BG (Bus Grant). If the answer is yes, the next question will be "Trap number for software break (0..0FH)?".

Enable DMA transfers to emulation memory ? no (yes)

An answer of no will not allow DMA transfers to emulation memory while an answer of yes will allow DMA devices to read/write to emulation memory provided that the timing is the same as the 680XX processor and that the DMA generated signals, i.e. LDS, UDS, AS, and RW are present at the pins of the emulator probe. Analysis of DMA transfers is also provided. The target system will not have access to DMA controls during a Host access.

The next question will be "Trap number for software break?"

Enable tracing of DMA memory transfers ? no (yes)

If you answer no to this question, no DMA activity will be recorded in the trace data; However, if the answer is yes, as described above in the question "Enable DMA transfers to emulation memory?", if a DMA device generates processor compatible data strobes at the pins of the emulation probe, analysis of DMA transfers is provided. DMA access to emulation memory is not allowed.

Enable tracing of DMA tags ? no (yes)

Answering no to all of the DMA configuration questions selects the "normal" 680XX response of the emulator to DMA activity, i.e. going inactive, no analysis, etc. If the answer is yes, a state will be recorded in the analysis unit whenever the DMA handshake occurs.

Trap number for software break (0..0FH) ? 0000FH

The software breakpoint number specifies which of the sixteen trap vectors of the 680XX will be used for the software breakpoint instruction.

NOTE

Without modifying the vector look-up table in the monitor, this feature of the emulator is not available.

If you desire to use the software breakpoint feature in developing your software, you must perform the steps outlined in Chapter 8 in the section "Modifying the Monitor to use Software Breakpoints" to give the vector address of the appropriate trap number to the monitor.

Enable emulator use of INT7 ? yes (no)

If you answer yes, the emulator will be allowed to use the INT7 for breaking the processor into the monitor. The target system is not prevented from using INT7. If, however, the answer to this question is no, the emulator will NOT be allowed to break. The target system may use INT7. Note that if the emulator cannot break, you will give up the stepping, analysis, breakpoints, illegal memory access and write to ROM detection, and "break" ing from the keyboard. NOTE: Caution should be used when answering no to this question. This will not allow the "break" function to be used, thus, only allowing the emulator to run in "real-time" with no method of looking at your registers, memory or any other features that are available when "Running in monitor." This feature is very similar to "restrict to real time runs" but will lock out the break key function. The question on sharing interrupt 7 will be skipped.

Enable Interrupt level 7 sharing ? yes (no)

If you answer this question yes, the "break" function will be transparent to your target system. Enabling this function will keep the interrupt acknowledge cycle i.e. AS, UDS&LDS,(DS on 68008) and RW from being passed to your system when an emulation system break (INT7) occurs. VMA (68000 Only) will also be held high. If, however, you answer no to this question and a break from the emulation system occurs at the same time as a target system INT7, there is a possibility that the target system interrupt acknowledge may be ignored.

Enable tristate delay on halts ? no (yes)

When using the HALT pin on the microprocessor to halt the operation of the 680XX, it is important that you know whether the halt request will occur synchronous or asynchronous to the activity on the bus. If you are positive that when halt is asserted the processor will definitely halt after completion of the current bus cycle, then an answer of no to this question will be sufficient. If, however, you are asserting HALT asynchronously, there is the possibility of pulling HALT active late in the current bus cycle and one more cycle will execute before the processor halts. In this case a delay is desirable to allow the processor to halt before any activity occurs on the bus.

Configuring for Interactive Measurements

It is possible to coordinate measurements between the modules of a multi-module system by selecting various options possible under this category. Options selected for interaction will be displayed by the measurement system monitor in multi-module systems.

If you decide to modify the default interactive measurements, then you cannot continue any function involving the analysis card, but it does not affect the rest of the system. If there is any conflict between the interaction specified by a command file and the interaction specified by the measurement system monitor, and it cannot be resolved, then you will have to modify your specification to resolve the conflict.

The internal analysis unit can interact with other measurement equipment during emulation through either or both of the BNC output ports located on the back of the development station. The analysis unit can also interact with other card cage analysis modules through the IMB connector located at the top of the analysis card. The following questions appear during configuration.

Modify interactive measurement specifications? no (yes)

If interaction is desired or if a previously defined interactive specification is to be modified, a "yes" answer to this question allows the analysis interaction specification format to be reviewed and modified as necessary. If no modification is desired, the "no" response should be selected. The Interactive Measurement questions will then be skipped, leaving the responses in their default or previously defined states.

If this question is answered "yes", the following series of questions will be presented in sequence.

(a) PORT 1? off (drive)

The "drive" option causes the internal analysis unit to output a pulse to Port 1 when the analysis trigger is encountered. This function is useful for arming or triggering an external measurement instrument such as a scope or logic analyzer.

If "off" is selected, PORT 1 has no function.

(b) PORT 2? off (drive)

The "drive" option causes the internal analysis unit to output a pulse to Port 2 when the analysis measurement is complete. This function is useful for arming or triggering an external measurement instrument such as a scope or logic analyzer.

If "off" is selected, Port 2 has no function.

(c) Active edge? rising (falling)

This question is only encountered if either Port 1 or Port 2 is configured to operate in the "drive" mode. The response specifies the polarity of the drive pulse which will be generated at the active ports.

"Rising" specifies a positive going output pulse whereas "falling" specifies a negative going output pulse. The polarity specification applies to both ports if both are active.

The following questions refer to the lines available through the IMB connector on the internal analysis board, and on other interacting modules.

(d) Trigger enable? off (drive) (receive)

1. No IMB Interaction over the trigger enable line.

If the "off" option is selected, internal analysis will not interact with the trigger enable line.

2. Drive IMB Trigger Enable

Selection of the "drive" option causes internal analysis to drive the IMB trigger enable line when analysis finds the internal trigger point or receives an external trigger.

3. Receive IMB Trigger Enable

Selection of the "receive" option prevents internal analysis from finding its internal trigger point until some other module has driven the trigger enable line.

In addition to the IMB trigger enable options, the following additional options are available with the 48 channel (HP 64302A) analysis board:

For 48 channel analysis there is one function that is always used whenever any other interaction is desired. This is the function of receiving the IMB Master Enable line in order to allow synchronous initiation of the multi-module systems. Internal analysis will select the correct option for this function depending on the options chosen for the other functions.

(e) External trigger? off (drive) (receive) (drive and receive)

1. No interaction over IMB trigger line.

When "off" is selected, internal analysis will not participate in any interaction over the IMB trigger line.

2. Drive IMB trigger

Selection of the "drive" option causes internal analysis to drive the trigger line when it finds its internal trigger point.

3. Receive IMB trigger

Selection of the "receive" option allows internal analysis to trigger either on finding its internal trigger point or when another module drives the IMB trigger line.

4. Drive and receive IMB trigger

Internal analysis will search until it finds its internal trigger or until another module drives the trigger line. Regardless of the source of the trigger, once internal analysis has triggered, it begins to drive the IMB trigger line.

(f) Internal trigger? on (off)

1. Enable internal trigger

If the "on" option is selected the internal triggering mechanism is enabled. This means that triggers specified via a "trace" or "specify trace" command will cause internal analysis to trigger if they are enabled (see trigger enable option above).

2. Disable internal trigger

If the "off" option is selected, then the internal triggering mechanism is disabled and will not cause a trigger. Thus triggers specified by "trace" or "specify trace" command will be ignored and internal analysis will only trigger when it is receiving an external trigger.

(g) Delay clock? off (drive)

1. No interaction on delay clock line

If the "off" option is selected then internal analysis will not interact over the delay clock line.

2. Drive delay clock line

Selecting the "drive" option causes internal analysis to drive the delay clock line once it has triggered, whether by an internal trigger or a received external trigger.

Naming Your Emulation Configuration Command File

Once you have finished answering, or defaulting, the configuration questions, the next question allows you to establish a command file containing all of the information pertaining to the questions just answered for emulation configuration. The command file is stored on disc and can then be called up for use during any future emulation session.

All that is required to create the command file is to type in a file name in answer to the following question. If no file name is entered, the configuration information will not be stored, and the questions will be required to be answered for each new emulation session.

Command file name? <file name> Enter PROGRAM for this example.

Configuration questions and answers will be stored in a command file of the name specified. Default is the current command file. If no command file exists, a new file will be created under the name provided. Specifying a command file avoids having to answer the configuration questions each time an emulation session is begun. There must be a command file specified for each module in a multi-module emulation session.

Emulation can be started with the same configuration by specifying the emul_com file name along with the "emulate" command. The answers to the questions may be changed by specifying "options" "edit" with the "emulate" command. When emulation is ended using the "end" command, the current state of the processor is stored in the emul_com file. An additional file of type "trace" is created containing the current analysis specification. This information allows emulation to be re-entered without resetting the processor and analysis hardware. This is done by specifying "options" "continue" in addition to the emul_com file name with the "emulate"

command. When entering an emulation session through "measurement_system" and "em680XX_S", an emulation command file is the only available option. An emulation session within measurement_system will always be continued, if possible. Editing of an emul_com file will be allowed only if there is a conflict, between the configuration file and the hardware, that must be resolved before entering the emulation session.

CONFIGURATION SWITCHES

There are three switches located on the top of the emulation pod, shown in figure 5-5, that allow you to select some infrequently changed hardware options. These switches are described in the following paragraphs.

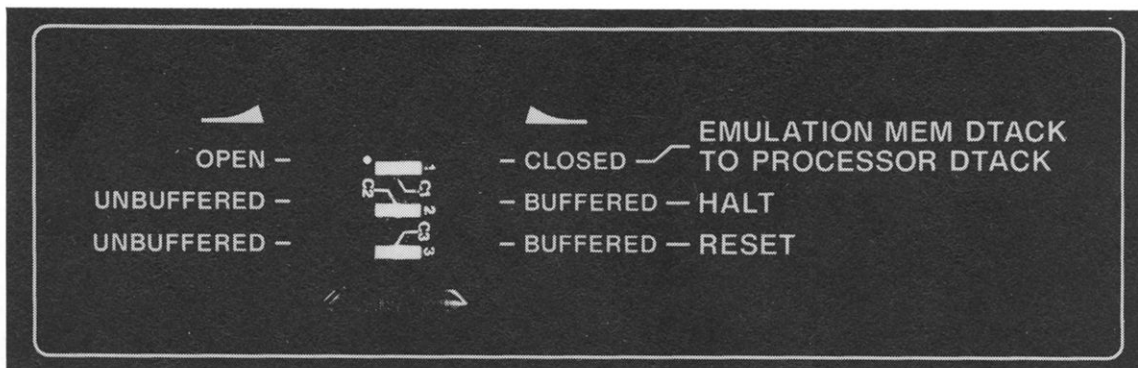


Figure 5-5. Configuration Switches

EMULATION MEM DTACK- When this switch is closed, it connects the plug-in lead labeled /DTACK to the emulation pod's 680XX processor /DTACK input. This may simplify the emulator hook-up if /DTACK in the target system is wire OR'ed at the processor pin. If this is true, the switch may be closed to make the connection, and hookup of the plug-in lead would be unnecessary.

HALT and RESET- These two switches may be set to either the BUFFERED or UNBUFFERED position (the meaning is identical for both switches). The /HALT and /RESET signals of the 680XX operate bi-directionally; the target system may drive them or the processor may drive them via a double bus fault or the RESET instruction. The emulation host system also drives these signals to reset the processor; this is considered an unnatural state for a 680XX. You may select the UNBUFFERED mode, in which case the signal passes from the 680XX to the host system, and activity by the host system will be seen by the target system. You may choose the BUFFERED mode in which case a buffer will be placed in line with the target system; the target system will be shielded from the host system activity, but it will also be shielded from the processor asserting the signal. Unbuffered is the recommended mode of operation.

NOTES

Chapter 6

USING THE EMULATOR--EXAMPLES

OVERVIEW

This chapter will:

- Provide review information on the 680XX microprocessor.
- Provide guidelines for out-of-circuit emulation.
- Provide guidelines for in-circuit emulation.
- Tell you how to enter and exit the emulation program.
- Provide examples using emulation and analysis features.

INTRODUCTION

How you use the emulator features will be determined by your target system and the types of software problems you encounter. The examples in this chapter are elementary, but, several elementary examples put together create powerful measurements. The examples used in this chapter are those obtained using the sample program given in chapter 3.

The analysis examples illustrated in this chapter can be performed only if an internal analyzer is included in the emulation system. If you have internal analysis, experiment with the trace function. Do not limit yourself to tracing on an address--trace on data and status (note the capability to trace on an interrupt acknowledge cycle). The analyzer allows you to be specific when you qualify a trace; take full advantage of this capability.

EXECUTION

After configuration, the execution portion of emulation is entered. In this case the processor has been reset. This condition is reported on the STATUS line of the display. In this example the absolute file must be loaded into emulation memory using the load command in the form "*load <ABSFILE>*". If the address range into which a program is to be loaded resides entirely in emulation memory, the processor remains in the reset state. If any portion of the program resides in memory which has been mapped as target memory, the processor will be released from the reset state. After loading all portions of the file which are to reside in emulation memory, a handshake will be performed to determine if the processor is executing in the emulation monitor program and, if not, a break will be performed. When the processor is in the monitor the target memory portion of the program will be loaded. This sequence can be performed manually by using the options of the load command which specify the portion of memory to be loaded.

RUNNING THE PROGRAM

Once the example program has been loaded, the processor can be released from the reset state with the "run" command. Note that if you have not stepped thru chapter 3 "Getting Started", you will not be able to release reset with the "run" command. Refer to chapter 3 in the section "Copying the Emulation Monitor Program to Your Userid" for information on "un-commenting" the VECTOR table. If the command is issued as the single keyword "run", the processor will use the start-up vector or routine to start execution in the emulation monitor. The STATUS line will display "Running in monitor" indicating that the HP 64000/monitor handshake is being performed. From this state the run command can be issued to begin execution of the program. If the command "run" is given, program execution will begin at the transfer address specified in the source program. This is either the label given with the END pseudo-op at the end of an assembly language module, or the main routine of a PASCAL program. If the transfer address is found to be 0 then the current SSPT and USPT values will not be changed. Thereafter "run" will cause execution to begin at the next program counter address as specified in the register display. If "run from <ADDRESS>" is issued, execution begins at the address specified.

PROCESSOR RESET

When the emulator is released from reset, the internal registers are initialized in the same manner in which the 680XX processor registers are initialized. If the emulator monitor is entered automatically from reset, all of the registers will remain in their reset state except the SSPT and USPT registers. These registers will point to the valid stack area established within the emulation monitor data area.

PROCESSOR STATUS MESSAGES

After the emulator has been given the run command the emulation software tests for bus activity to ensure that the emulator is active. If no bus cycles occur for a period of 100 milliseconds, a "Slow clock" flag is set in the hardware. When the emulation software finds this flag to be true, one of the following status messages will be displayed:

No memory cycles: Displayed when the 680XX has executed a STOP instruction. To terminate this condition enter "break" or "reset" from the keyboard. This message can also appear if the /HALT switch on the pod has been set to "BUFFERED". The hardware that detects the halt condition is on the target side of the halt buffer. Thus, if the switch is set "buffered" and the 680XX performs a double fault, the halt message will not appear. You may determine whether a STOP instruction or double fault caused the "no memory cycles" message by temporarily setting the halt switch to "unbuffered" and observing whether the halt message appears.

Reset: Displayed when the inactivity is caused by the emulation system or the target setting the RESET line. The message will also appear if the "reset" softkey is depressed.

- halt: Displayed when the inactivity is caused by the emulation system or the target system asserting the /HALT line of the 680XX.
- bus grant: The bus grant message indicates that the 680XX /BR is asserted and that the DMA device has not done any cycle activity. Note that unless the DMA options that allow DMA cycle activity are selected (analysis of DMA activity or DMA access to emulation memory), no activity would be expected. The condition can be terminated by releasing the target's DMA hardware.
- wait: The wait message indicates that the 680XX is waiting for a /DTACK memory response or other memory response. The condition the may be terminated by asserting /DTACK, /VPA , or /BERR; or if this is not convenient, the 680XX can be reset by depressing the reset softkey.

ANALYSIS STATUS INFORMATION

Table 6-1. 680XX Status Definition

The 8 bits of analysis status are as follows:

```

b7 b6 b5 b4 b3 b2 b1 b0

b0 = 0   read cycle
      = 1   write cycle
b1 = 0   word cycle
      = 1   byte cycle
b2 =     only applicable when b1 = 1
      = 0   lower byte
      = 1   upper byte
b3 = 0   bus cycle mode
      = 1   execute mode
  
```

b6 b5 b4 - processor function codes

	<u>Bus Cycle Mode</u>	<u>Execute Only Mode</u>
0 0 0	Undefined	User Postword
0 0 1	User Data	User Data
0 1 0	User Program	User Opcode
0 1 1	Undefined	Undefined
1 0 0	Undefined	Supervisor Postword
1 0 1	Supervisor Data	Supervisor Data
1 1 0	Supervisor Program	Supervisor Opcode
1 1 1	CPU Space	CPU Space

```

b7 = 0 680XX cycle
      = 1 DMA cycle
  
```

Table 6-2. Softkey Labels and Mnemonic Status Definitions

Name	Value	Definition
read	XXXX XXX0B	Read Access
write	XXXX XXX1B	Write Access
words	XXXX XX0XB	Word Access
bytes	XXXX XX1XB	Byte Access
byte_low	XXXX XX1XB	Low Byte Access
byte_high	XXXX X11XB	High Byte Access
opcode	XX10 1XX0B	Opcode Access
postword	XX00 1XX0B	Postword Access
user	X0XX XXXXB	User State
program	XX10 XXXXB	Program Reference
supervis	X1XX XXXXB	Supervisor State
data_acc	XX01 XXXXB	Data Access
int_ack	X111 XXXXB	Interrupt Acknowledge
cycle_dma	1XXX XXXXB	DMA Cycle
execution	XXXX 1XXXB	Execution Mode

MEMORY RESOURCES DURING EMULATION

During configuration the memory resources of the 680XX processor must be mapped as emulation memory, target memory, or as guarded memory. Memory is mapped to 4K byte resolution on the 68000, and 256 byte resolution on the 68008. Emulation memory is memory that is physically located in the HP 64000 system and is dedicated for use by the emulator only. Target memory is memory that is physically located in your target system. Memory mapped as guarded is memory that, under normal conditions, should not be accessed by your target system. Any reference to the address space mapped as guarded memory will result in an emulation memory break and the display of the error message:

```
"ERROR: M680XX-Running in monitor      Access guarded mem. <ADDRESS>".
```

OUT-OF-CIRCUIT EMULATION

Out-of-circuit emulation (no target system connected) is usually performed with the intent of developing or debugging software. With internal emulation, the only clock that can be used is the internal clock of the emulator; therefore, code execution time will be relative to the internal clock speed. This should be kept in mind if the target system will have a different clock speed than the internal clock of the emulator (10 MHz). Refer to Appendix C for information concerning radio frequency interference when using the emulator in out-of-circuit mode.

Guidelines for Out-of-Circuit Emulation

- Select internal clock during emulation configuration.
- Map all memory required to emulation memory during emulation configuration.

IN-CIRCUIT EMULATION

The HP 64000 can perform emulation in real-time or nonreal-time with a target system connected. If the real-time performance of the target system is important, emulation should be done in real-time with particular attention paid to the type of run commands issued during emulation.

In some cases emulation may be required to run in real-time because running in nonreal-time is not possible, such as with target systems that process interrupts and/or depend on a real-time clock for operation. Target systems of these types can not be emulated thoroughly if real-time emulation is not available. Therefore, it is important to be aware of the types of emulator commands that will cause the emulator to operate in nonreal-time.

In addition, you should realize the implications of using emulation memory in part or in whole for emulation. The use of emulation memory could affect real-time emulation depending on the implementation of the target system.

Emulation Memory and Target System Memory

The use of emulation memory and/or target system memory can significantly affect the operation of the emulator. Ideally, emulation of a microprocessor should be done with as much of the final target system hardware as possible. Since this is not feasible at the beginning of the development cycle, the HP 64000 emulator provides emulation memory to replace target system memory during the project development stage.

In general, the final target system memory will not have the same specifications as the HP 64000 emulation memory. Selection of the clock for emulation can affect memory accesses by the target processor. Therefore, it is recommended that the external clock be used whenever possible to assure synchronous operation between emulator and target systems. The emulator will, however, allow use of the internal clock with a target system.

Guidelines for In-Circuit Emulation

- Select the external clock during emulation configuration if one is available on the target system.
- If any memory exists on the target system, map that memory as target memory during emulation configuration.
- Read all of the emulation probe cable connector installation instructions in Appendix G2 before you install the target system microprocessor socket.
- Turn on the development station before the target system and turn the development station off first.



If your target system includes any CMOS components, turn on the target system first, then turn on the development station. Likewise; turn the development station off first, then turn off the target system.

BEGINNING AND ENDING EMULATION

Emulation can be entered in two ways: by answering the emulation configuration questions or by specifying an emulation command file which you have programmed to answer the configuration questions.

If more than one measurement system module (emulator, software analysis, timing analyzer, state analyzer) is present in the development station or the High Level Software Analyzer software for your microprocessor is present on your system disc, the command "meas_sys" will appear at the first level of softkeys. If an emulator is the only module present, then "emulate" will be present at the first level of softkeys.

When you end emulation using the "end" command, the current state of the processor is stored in the emul_com file. An additional file of type "trace" is created which contains the current analysis specification. This information allows emulation to be re-entered without resetting the processor and analysis hardware. This is done by specifying "options" "continue" in addition to the emul_com file name with the "emulate" command. When entering an emulation session through "measurement_system" and "eM680XX_S", an emulation command file is the only available option. An emulation session within measurement_system will always be continued, if possible. Editing of an emul_com file will be allowed only if there is a conflict, between the configuration file and the hardware, that must be resolved before entering the emulation session.

To begin emulation via "meas_sys":

PRESS the *meas_sys* softkey, then **RETURN**

The resulting display:

Measurement System: Current Configuration

Module	Slot	Status	Description
eM680XX	8		Emulator for M680XX

STATUS: Awaiting measurement system request userid YOURID

eM680XX_8 sw_anly _____ _____ _____ print_dsp _____ end

The following command will select the 680XX emulator and answer the emulation configuration questions:

```
eM680XX_S DEMO RETURN ( S is the slot number of the control board)
```

To end emulation press the *end* softkey, then the **RETURN** key. You are now back to the measurement system display illustrated on the previous page. To return to the system monitor level of softkeys, press the *end* softkey and the **RETURN** key again.

A command file for assembling, linking, and entering emulation via meas_sys would look like this:

```
assemble MODULE  
assemble Mon_680XX  
link DEMO  
measurement_system  
eM680XX_S DEMO  
load DEMO
```

*eM680XX_S means that the emulator control board resides in motherboard option slot number eight in this example.

Refer to the "measurement_system" syntax example for multi-module systems (not using options continue) given in chapter 7.

EMULATION EXAMPLES

Demonstration Configuration

The demonstration program which is used in the following examples is listed in chapter 3. The requirements for the examples are:

Assembled Modules:

```
MODULE  
Mon_680XX
```

Linker Specifications:

```
all default except:  
MODULE located at PROG=00002000H  
Mon_680XX located at PROG=000007000H  
Absolute file name= DEMO
```

Emulation Configuration:

```
Micro-processor clock source? internal  
Restrict to real-time runs? no  
Number of significant address bits? 24 (68000) & 20 for the (68008).  
Break processor on write to ROM? yes  
Emulation memory:
```

Emulator/Analyzer 68000/68008
Using the Emulator--Examples

000H thru 0FFFH emulation rom
2000H thru 2FFFH emulation rom
7000H thru 7FFFH emulation ram
default guarded

Modify simulated I/O? no
Re-configure emulator pod? no
Modify interactive measurement specification? no
Command file name? DEMO

The system command file that we will use is DEMO. This is the file that was made in chapter 3 by editing the DEMO_CMD file. It is not necessary to use all of the commands in the DEMO_CMD file since it will not be necessary to reassemble or relink the programs being used. The DEMO system command file from chapter 3 was made as follows:

```
measurement_system
eM680XX_S DEMO (S indicates the slot number of control card)
modify memory word 2000H thru 2FFFH to 4E71H
load DEMO
```

Typing in DEMO from the keyboard will cause the emulation questions to be answered (DEMO:emul_com file) and the emulation memory to be loaded (DEMO:absolute file). When the memory is loaded the emulation processor is reset.

NOTE

The example displays shown in this chapter were created using a 68000 emulation system. The displays for the 68008 may be slightly different.

To Display Registers:

```
run RETURN  
display registers RETURN
```

The resulting display:

M680XX Registers

```
Next PC 000000 STATUS 2700 < s      >      USPT 0000761A  SSPT 0000761A  
D0-D7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  
A0-A7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000761A
```

STATUS: M680XX--Running in monitor

____ 10:48

_display registers

run trace step display modify break end ---ETC---

To Step Registers <#STEPS> From <ADDRESS>:

```
display registers RETURN  
step 2 from 2000H RETURN
```

The resulting display:

M680XX Registers

M680XX Registers

```
Next PC 000000 STATUS 1EBE < xnzv > USPT 0000761A SSPT 00007622  
D0-D7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  
A0-A7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000761A
```

PC 002000 Opcode 4200 CLR.B D0

```
Next PC 002004 STATUS 2719 < sxn c> USPT 0000761A SSPT 0000761A  
D0-D7 000000FF 00000000 00000000 00000000 00000000 00000000 00000000 00000000  
A0-A7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000761A
```

PC 002004 Opcode 66FC BNE.B 0002002H

```
Next PC 002002 STATUS 2719 < sxn c> USPT 0000761A SSPT 0000761A  
D0-D7 000000FF 00000000 00000000 00000000 00000000 00000000 00000000 00000000  
A0-A7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000761A
```

STATUS: M680XX--Step complete Trace complete _____ 15:48

_step 2 from 2000H

run trace step display modify break end ---ETC---

To Display Memory - Blocked Byte:

display memory 2000H blocked byte **RETURN**

The resulting display:

Memory:bytes:blocked												
address	data								:hex		:ascii	
002000-07	42	00	53	00	66	FC	4E	F9	B	S	f	N y
002008-0F	00	00	70	00	4E	71	4E	71		p	N	q N q
002010-17	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002018-1F	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002020-27	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002028-2F	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002030-37	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002038-3F	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002040-47	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002048-4F	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002050-57	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002058-5F	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002060-67	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002068-6F	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002070-77	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q
002078-7F	4E	71	4E	71	4E	71	4E	71	N	q N q	N	q N q

STATUS: M680XX--Step complete Trace complete _____ 15:48

_display memory 2000H blocked byte

run trace step display modify break end ---ETC---

To Display Memory - Mnemonic:

display memory mnemonic (RETURN)

The resulting display:

```
Memory :mnemonic
address
-----
002000 CLR.B D0
002002 SUBQ.B #1,D0
002004 BNE.B 0002002H
002006 JMP 0007000H
00200C NOP
00200E NOP
002010 NOP
002012 NOP
002014 NOP
002016 NOP
002018 NOP
00201A NOP
00201C NOP
00201E NOP
002020 NOP
002022 NOP
002024 NOP
002026 NOP
002028 NOP
00202A NOP
00202C NOP
00202E NOP
002030 NOP

STATUS: M680XX--Step complete          Trace complete          ____ 15:48

_display memory mnemonic

run      trace      step      display      modify      break      end      ---ETC---
```

To Display Memory - Absolute Byte:

display memory absolute byte **(RETURN)**

The resulting display:

Memory:bytes:absolute		
address	data :hex	:ascii
002000	42	B
002001	00	
002002	53	S
002003	00	
002004	66	f
002005	FC	
002006	4E	N
002007	F9	y
002008	00	
002009	00	
00200A	70	p
00200B	00	
00200C	4E	N
00200D	71	q
00200E	4E	N
00200F	71	q

STATUS: M680XX--Step complete Trace complete _____ 15:48

_display memory absolute byte

run trace step display modify break end ---ETC---

To Display Memory - Real:

display memory real **RETURN**

The resulting display:

```
Memory : short real
address      data :real
-----
002000      3.20811E  1
002004      5.95747E 23
002008      4.01780E-41
00200C      1.01211E  9
002010      1.01211E  9
002014      1.01211E  9
002018      1.01211E  9
00201C      1.01211E  9
002020      1.01211E  9
002024      1.01211E  9
002028      1.01211E  9
00202C      1.01211E  9
002030      1.01211E  9
002034      1.01211E  9
002038      1.01211E  9
00203C      1.01211E  9

STATUS: M680XX--Step complete      Trace complete      ____ 15:48

_display memory real

run      trace      step      display      modify      break      end      ---ETC---
```

To Display Memory Blocked Byte Offset:

display memory blocked byte offset_by 1000D **RETURN**

The resulting display:

```
Memory:bytes:blocked:offset=000003E8H
```

address	data								:hex								:ascii							
001C18-1F	42	00	53	00	66	FC	4E	F9	B	S	f		N	y										
001C20-27	00	00	70	00	4E	71	4E	71	p	N	q	N	q											
001C28-2F	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C30-37	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C38-3F	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C40-47	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C48-4F	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C50-57	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C58-5F	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C60-67	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C68-6F	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C70-77	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C78-7F	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C80-87	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C88-8F	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								
001C90-97	4E	71	4E	71	4E	71	4E	71	N	q	N	q	N	q	N	q								

STATUS: M680XX--Step complete Trace complete ____ 15:48

_display memory blocked byte offset_by 1000D

run trace step display modify break end ---ETC---

To Modify and Display Memory - Byte:

modify memory byte 2000H thru 2080H to 0FFH (RETURN)
display memory 2000H (RETURN)

The resulting display:

Memory:bytes:blocked								
address	data							
002000-07	FF	FF	FF	FF	FF	FF	FF	FF
002008-0F	FF	FF	FF	FF	FF	FF	FF	FF
002010-17	FF	FF	FF	FF	FF	FF	FF	FF
002018-1F	FF	FF	FF	FF	FF	FF	FF	FF
002020-27	FF	FF	FF	FF	FF	FF	FF	FF
002028-2F	FF	FF	FF	FF	FF	FF	FF	FF
002030-37	FF	FF	FF	FF	FF	FF	FF	FF
002038-3F	FF	FF	FF	FF	FF	FF	FF	FF
002040-47	FF	FF	FF	FF	FF	FF	FF	FF
002048-4F	FF	FF	FF	FF	FF	FF	FF	FF
002050-57	FF	FF	FF	FF	FF	FF	FF	FF
002058-5F	FF	FF	FF	FF	FF	FF	FF	FF
002060-67	FF	FF	FF	FF	FF	FF	FF	FF
002068-6F	FF	FF	FF	FF	FF	FF	FF	FF
002070-77	FF	FF	FF	FF	FF	FF	FF	FF
002078-7F	FF	FF	FF	FF	FF	FF	FF	FF
002080-80	FF							

STATUS: M680XX--Step complete Trace complete _____ 15:48

_display memory 2000H

run trace step display modify break end ---ETC---

To Modify and Display Memory (Continued):

modify memory 2000H thru 2012H to 1,2,3 **(RETURN)**

Press the **(RECALL)** key located above the cursor controls on the keyboard until you see "*display memory 2000H*" **(RETURN)**

The resulting display:

Memory:bytes:blocked										
address	data								:hex	:ascii
002000-07	01	02	03	01	02	03	01	02		
002008-0F	03	01	02	03	01	02	03	01		
002010-17	02	03	01	FF	FF	FF	FF	FF		
002018-1F	FF	FF	FF	FF	FF	FF	FF	FF		
002020-27	FF	FF	FF	FF	FF	FF	FF	FF		
002028-2F	FF	FF	FF	FF	FF	FF	FF	FF		
002030-37	FF	FF	FF	FF	FF	FF	FF	FF		
002038-3F	FF	FF	FF	FF	FF	FF	FF	FF		
002040-47	FF	FF	FF	FF	FF	FF	FF	FF		
002048-4F	FF	FF	FF	FF	FF	FF	FF	FF		
002050-57	FF	FF	FF	FF	FF	FF	FF	FF		
002058-5F	FF	FF	FF	FF	FF	FF	FF	FF		
002060-67	FF	FF	FF	FF	FF	FF	FF	FF		
002068-6F	FF	FF	FF	FF	FF	FF	FF	FF		
002070-77	FF	FF	FF	FF	FF	FF	FF	FF		
002078-7F	FF	FF	FF	FF	FF	FF	FF	FF		

STATUS: M680XX--Step complete Trace complete _____ 15:48

_display memory 2000H

run trace step display modify break end ---ETC---

USING ANALYSIS COMMANDS

Analysis may be performed either by first initiating the program run and then specifying the trace parameters or by specifying the trace parameters first and then initiating the program run. In either case, once a trace command is initiated, the analysis module monitors the system buses of the emulation processor to detect the states specified in the trace command. When the trace specification has been satisfied, a message will appear on the status line showing "trace complete". At that time the contents of the trace memory can be displayed. If the trace memory contents exceed the page size of the display, the **NEXT PAGE**, **PREV PAGE**, **ROLL UP**, or **ROLL DOWN** keys may be used to display all the trace memory contents.

Trigger and storage qualification can be specified without initiating a trace by using the specify trace command, and traces can be initiated without altering the trigger and storage qualifications by using the execute command.

The trace command consists of the components described in the following paragraphs:

a. <TRIGGER> The "trigger" is the event on the emulation bus to be used as the starting, ending, or centering event for the trace.

b. <QUALIFIER> The storage specification determines which of the traced states will be stored in the trace memory for display upon completion of the trace. The trace memory can be filled by those states which occur immediately before or immediately after the specified trigger event, or half of the memory can be filled by states which precede the trigger and half by those which follow the trigger event. Events can be selectively saved by pressing the trace only and entering the specific events to be saved. When this option is used, only the indicated states occurring in the specified position relative to the trigger are stored in the trace memory.

c. <COUNT> The count option specifies whether time or the occurrence of a state will be counted during the trace. The data can be displayed either "relative" to the count at the previous stored state, or "absolute" with respect to the trigger. All count measurements can be displayed in either absolute or relative mode. The absolute count is the total count from the trigger to each measured state. A plus sign (+) preceding the trace number indicates that the state occurred after the trigger state. A minus sign (-) indicates that the state has occurred before the trigger state.

The "relative count" mode displays the count between consecutive states stored in the trace buffer. It can be used to measure execution times of subroutines and instructions or the time between the occurrence of the same state in the execution of a program.

d. <BREAK> The break specification causes an exit from your target system executing program to the emulation monitor program at a predetermined point in the emulation monitor program.

e. *again* Entry of the "again" parameter causes the trace to be performed again using the previous trace parameters.

f. *repetitively* Entry of the "repetitively" parameter causes a new trace to be initiated after the results of the previous trace are displayed. The trace will continue until a stop_trace or a new trace command is issued.

ANALYSIS EXAMPLES

For the following examples the emulator is configured, emulation memory is loaded, and the emulation processor is reset. If you are starting this exercise as a new session, type in DEMO from the keyboard and press **(RETURN)**. This will cause the emulation questions to be answered (DEMO:emul_com file) and the emulation memory to be loaded (DEMO:absolute file). When the memory is loaded the emulation processor is reset. If you are still in a session from the previous examples and you have modified the memory, you will need to reload you absolute file. To do this press the *load* softkey, type in DEMO from the keyboard, then press **(RETURN)**.

To See Real-time Tracing of Processor Operations

When this exercise was started (refer to the paragraph entitled "Emulation Examples") runs were not restricted to real-time. In order to make sure that the following examples perform in real-time, you will need to modify the emulation configuration to restrict to real-time runs. To do this:

Press the *modify config* softkeys, then **(RETURN)**.

Press **(RETURN)** again and the question "Restrict to real-time runs?" will appear on the screen. Press the *yes* softkey, then the **(RETURN)** key. Continue to press the **(RETURN)** key until the STATUS line reads M680XX--Reset; indicating that you are back in the emulation session. You are now in the real-time mode.

To Trace After <ADDRESS>:

trace after 2000H **(RETURN)**
run from 2000H **(RETURN)**
display trace mnemonic **(RETURN)**

The resulting display:

Trace: bus cycle data				break: none	count:
line#	address	opc/data	mnemonic	opcode or status	time, relative
after	002000	4200	CLR.B	D0	
+001	002002	5300	SUBQ.B	#1,D0	<1. uS
+002	002004	66FC	BNE.B	0002002H	1. uS
+003	002006	4EF9	JMP	*****	<1. uS
+004	002002	5300	SUBQ.B	#1,D0	1. uS
+005	002004	66FC	BNE.B	0002002H	1. uS
+006	002006	4EF9	JMP	*****	1. uS
+007	002002	5300	SUBQ.B	#1,D0	1. uS
+008	002004	66FC	BNE.B	0002002H	<1. uS
+009	002006	4EF9	JMP	*****	1. uS
+010	002002	5300	SUBQ.B	#1,D0	1. uS

STATUS: M680XX--Running Trace complete _____ 15:48

_display trace mnemonic

run trace step display modify break end ---ETC---

To Display Trace Absolute:

display trace absolute **RETURN**

The resulting display:

```
Trace: bus cycle data          break: none          count:
line#  address opc/data status:hex          time, relative
-----
after  002000  4200    64
+001   002002  5300    64          <1.   uS
+002   002004  66FC    64           1.   uS
+003   002006  4EF9    64          <1.   uS
+004   002002  5300    64           1.   uS
+005   002004  66FC    64           1.   uS
+006   002006  4EF9    64           1.   uS
+007   002002  5300    64           1.   uS
+008   002004  66FC    64          <1.   uS
+009   002006  4EF9    64           1.   uS
+010   002002  5300    64           1.   uS

STATUS: M680XX--Running          Trace complete          _____ 15:48

_display trace absolute

run   trace  step  display  modify  break  end  ---ETC---
```

To Display Trace Status Binary:

display trace status binary count absolute **RETURN**

The resulting display:

Trace: bus cycle data				break: none	count:	
line#	address	opc/data	status:binary		time,	absolute
after	002000	4200	01100100		0.	uS
+001	002002	5300	01100100		+ 0.	uS
+002	002004	66FC	01100100		+ 1.	uS
+003	002006	4EF9	01100100		+ 1.	uS
+004	002002	5300	01100100		+ 1.	uS
+005	002004	66FC	01100100		+ 2.	uS
+006	002006	4EF9	01100100		+ 3.	uS
+007	002002	5300	01100100		+ 3.	uS
+008	002004	66FC	01100100		+ 3.	uS
+009	002006	4EF9	01100100		+ 4.	uS
+010	002002	5300	01100100		+ 4.	uS

STATUS: M680XX--Running Trace complete ____ 15:48

_display trace status binary count absolute

run trace step display modify break end ---ETC---

To Display Trace Status Mnemonic:

display trace status mnemonic count relative **RETURN**

The resulting display:

Trace: bus cycle data			break: none	count:	
line#	address	opc/data	status:mnemonic	time, relative	
after	002000	4200	supr program read		
+001	002002	5300	supr program read	<1.	uS
+002	002004	66FC	supr program read	1.	uS
+003	002006	4EF9	supr program read	<1.	uS
+004	002002	5300	supr program read	1.	uS
+005	002004	66FC	supr program read	1.	uS
+006	002006	4EF9	supr program read	1.	uS
+007	002002	5300	supr program read	1.	uS
+008	002004	66FC	supr program read	<1.	uS
+009	002006	4EF9	supr program read	1.	uS
+010	002002	5300	supr program read	1.	uS

STATUS: M680XX--Running Trace complete _____ 15:48

_display trace status mnemonic count relative

run trace step display modify break end ---ETC---

NOTES

Chapter 7

COMMAND SUMMARY AND SYNTAX

OVERVIEW

This chapter will:

- Describe each of the operational commands used in emulation.
- Describe each of the display and list commands used in emulation.
- Describe the trace commands used in emulation analysis.
- Provide the syntax for using the commands.

INTRODUCTION

This chapter lists, and briefly discusses, the commands used by the emulation subsystem. The commands are divided into three categories; operational commands, display and list commands, and analysis and interactive commands. The commands are first listed by function, the syntax for each command is then given in alphabetical order, and where applicable (for more complex syntactical options), a command syntax diagram is given. In this chapter we will use the example programs from Chapter 5. These programs are entitled PROGRAM1 and PROGRAM2. You developed an emulation command file entitled PROGRAM:emul_com. If you are not already in *meas_sys* then press:

```
meas_sys RETURN
```

to enter the measurement system. When prompted with the "eM680XX_S" in the measurement system, press:

```
eM680XX_S PROGRAM RETURN (S indicates the slot number of the control board)
```

This will load the emulation configuration map that was created in chapter 4.

```
load PROGRAM RETURN
```

If you are already in an emulation session, press:

```
end RETURN
```

Then press: *eM680XX_S* PROGRAM **RETURN**

You are now ready for an emulation session.

OPERATIONAL COMMANDS

The operational commands are used to provide a way for you to perform specific operations in your emulation environment. The syntax for each of the operational commands is given later in this chapter. The syntax description is intended to acquaint you with the different operational commands. The syntactical variables used in the following discussions are described in detail in Appendix A.

Command Line Comment Delimiter

The comment delimiter is a semicolon, and is interpreted in such a way that any text following the semicolon, to the end of the command line, will be ignored by the emulation subsystem.

In the example:

```
run from START; causes program execution to begin
```

only the command line text preceding the semicolon will be acted upon.

Operational Command Options

The operational command options available in the 680XX emulation software include the commands listed below. The syntax for these commands is given alphabetically later in this chapter.

- break
- end
- emulate
- execute
- halt
- load
- measurement_system (for multi-module systems - using options continue)
- measurement_system (for multi-module systems - without using options continue)
- modify
- modify analysis_mode_to
- modify configuration
- modify io_port
- modify memory
- modify register
- modify software_breakpoints
- reset
- run
- specify
- step
- stop_trace
- store
- trace
- wait

DISPLAY AND LIST COMMANDS

The display command allows you to display various information on the CRT. The list command allows you to list various information to a printer, or to a file. There are six basic types of information which may be viewed by using either the display or list command. These are:

- Memory data
- Register contents
- Trace information
- Global and local symbols
- Software breakpoint information
- I/O ports

Display and List Command Options

The display and list command options available in the 680XX emulation software include the commands listed below. A brief discussion of each of the types of information which can be viewed by using the display and list command follows the listing. The syntax for these commands is given alphabetically later in this chapter.

- display/list memory
- display/list registers
- display/list trace
- display/list global_symbols
- display/list local_symbols_in
- display/list software_breakpoints
- display/list io_port

Memory Data



For data taken from memory, the starting address in memory or a list of memory address ranges may be specified. Whether the data comes from emulation or user memory depends upon the memory map assignments made during configuration of the emulation command file. Unless otherwise specified, memory data is displayed statically with the actual memory address shown. (The static display shows the memory contents existing when the display command is executed.) The data is displayed in hexadecimal form with corresponding ASCII characters as shown in figure 7-1.

Memory:bytes:blocked													
address	data								:hex	:ascii			
002000-07	3E	7C	3F	FC	23	F9	00	00	>		?		# y
002008-0F	40	00	00	00	35	00	4E	B9	@				5 N 9
002010-17	00	00	25	00	23	F9	00	00		%			# y
002018-1F	35	04	00	00	40	04	4E	F9	5			@	N y
002020-27	00	00	20	00	00	0F	02	29)
002028-2F	00	8C	00	41	00	08	00	4D		A			M
002030-37	08	5F	00	27	02	49	04	01	-	'			I
002038-3F	00	00	00	0F	00	21	00	4E					! N
002040-47	00	01	00	AA	02	49	00	6D		*			I m
002048-4F	00	45	00	18	00	41	00	5F	E			A	-
002050-57	00	00	00	0B	00	01	00	03					
002058-5F	00	0B	00	0D	00	19	00	06					
002060-67	00	13	00	48	00	4C	02	49		H		L	I
002068-6F	00	09	00	4B	02	0C	00	49		K			I
002070-77	00	21	40	4B	00	88	00	41	!	@	K		A
002078-7F	00	0C	00	01	00	8D	00	47					G

Figure 7-1. Memory Contents - Hexadecimal and ASCII

Syntax for the display memory and list memory commands is very similar. The repetitively option, however, is available for display commands only. The display and list commands can be modified so that memory data is displayed or listed using one or more of the following techniques:

- a. Data may be viewed in a repetitive mode which causes the display to be constantly updated. This can be useful if the data in the memory are continuously changing. The display, however, is not updated in real time.
- b. Data may be viewed in mnemonic form rather than in hexadecimal form as shown in figure 7-2. However, it is advisable to use a form consistent with the data being displayed. For instance, it makes sense to display memory containing program code in mnemonic form, but mnemonic form does not make sense for viewing memory locations containing random arithmetic values. The starting address for a mnemonic display should be the beginning of an opcode.
- c. Real number display/list. Data may be viewed as real numbers in either the short form (four bytes) or the long form (eight bytes).
- d. Memory addresses may be displayed "offset" from the actual value. The address offset allows the actual addresses to be offset by a value specified by the user. If the value is correctly chosen, the address space displayed will start at location 0000H and will correspond to the listing generated by the assembler or compiler. For example, if a module originating at address X is linked with other modules, it may be assigned a new starting address X+Y where Y is a value that depends on the number and size of the other modules being linked. Offset, therefore, allows the user to enter "Y" so that the addresses appear the same as in the source listing.

The display address will increment or decrement by units of one when using the  or  keys to view memory data in the mnemonic format. In this way the currently displayed mnemonic page

can be aligned beginning at a new starting address. The **ROLL UP** (or **ROLL DOWN**) key in a mnemonic display will disassemble the next (or previous) address from the first (or last) displayed address, leaving the rest of the display unchanged. (**ROLL UP** and **↑**), and **ROLL DOWN** and **↓** keys, are equivalent in either absolute or blocked modes.)

The **NEXT PAGE** and **PREV PAGE** keys will replace all of the data with new data. The **NEXT PAGE** will place the next instruction address and succeeding instruction addresses and corresponding data on the screen. The **PREV PAGE** key will place the preceeding instruction addresses and corresponding data on the screen. In some cases, in the **PREV PAGE** mode, there may be a slight delay before the data is placed on the screen. The delay results when the system steps backwards through the memory until sufficient data has been gathered to fill the screen.

Memory address	mnemonic
002000	MOVEA.W #03FFCH,A7
002004	MOVE.L 0004000H,03500H
00200E	JSR 0002500H
002014	MOVE.L 0003504H,04004H
00201E	JMP 0002000H
002024	ORI.B #029H,A7
002028	ORI.L #000410008H,A4
00202E	ORI.W #0085FH,A5
002032	ORI.B #049H,-[A7]
002036	SUBI.B #000H,D1
00203A	ORI.B #021H,A7
00203E	ORI.W #00001H,A6
002042	ORI.L #00249006DH,00045H[A2]
00204A	ORI.B #041H,[A0]+
00204E	ORI.W #00000H,[A7]+
002052	ORI.B #001H,A3

Figure 7-2. Memory Contents - Mnemonic

Register Contents

Register data is displayed as shown in figure 7-3. The program counter (PC) value can be offset by a value and the next PC can also be offset as specified by the user. The offset is done for the same reason as described above for memory data.

Trace Information

Trace information may also be displayed or listed using the display/list command. Figure 7-4 shows a trace memory display.

M680XX Registers

```
Next PC 002512 STATUS 2700 < s > USPT 0001061A SSPT 00003FF8
D0-D7 00000003 00000000 00000000 00000000 00000000 00000000 00000000 00000000
A0-A7 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00003FF8
```

Figure 7-3. Register Contents

Global and Local Symbols

These symbols may be viewed on the display. Local symbols are symbols defined in the source file for a single program module. Global symbols are those that are declared to be global in any source file. They are defined using the assembler pseudo instruction, GLB (or \$GLOBVAR+\$ in the compiler). When the display command is used to examine either of these symbol types, the display will contain the symbol name, absolute address, and, for symbols located in emulation memory, their present value (and for local symbols the relative value of PROG, DATA, COMN).

Software Breakpoint Table

A total of 16 breakpoints can be stored in the software breakpoint table. The breakpoint table can be viewed with the display/list software breakpoints command. Information in the table includes the current status of each entry. For a more complete discussion see the section on software breakpoints, later in this chapter. The use of an offset is possible with display/list software breakpoints.

Trace: execution data				break: none	count:
line#	address	opc/data	mnemonic opcode or status	time, relative	
after	002000	3E7C	MOVEA.W #03FFCH,A7		
+002	002004	23F9	MOVE.L 0004000H,03500H	1.	uS
+005	004000	0000	supr data read	1.	uS
+006	004002	0001	supr data read	1.	uS
+008	003500	0000	supr data write	1.	uS
+009	003502	0001	supr data write	<1.	uS
+011	00200E	4EB9	JSR 0002500H	1.	uS
+014	003FF8	0000	supr data write	1.	uS
+015	003FFA	2014	supr data write	<1.	uS
+016	002500	2039	MOVE.L 0003500H,D0	1.	uS
+019	003500	0000	supr data read	1.	uS
+020	003502	0001	supr data read	<1.	uS
+021	002506	D0B9	ADD.L 0004008H,D0	1.	uS
+024	004008	0000	supr data read	1.	uS
+025	00400A	0002	supr data read	1.	uS
+026	00250C	23C0	MOVE.L D0,0003504H	<1.	uS

Figure 7-4. Trace Memory Display

ANALYSIS AND INTERACTIVE COMMANDS

The analysis commands are used to specify the particular part of a program that is to be traced and displayed. The trace measurement may be made once and displayed statically or the same measurement may be made repetitively and the results continually updated.

The trace command causes 256 states to be collected and stored in the trace memory. The trace memory is displayed relative to the trigger position. The trigger may occur at the beginning (after), at the middle (about), or at the end (before) of the trace memory contents. (Note: The display is capable of listing only 16 lines per page, and therefore the **PREV PAGE**, **NEXT PAGE**, **ROLL UP**, or **ROLL DOWN** keys are used to view all measured states.)

Emulation can interact with other modules of a multiple module system over the intermodule bus, or with external equipment through the BNC ports on the rear of the development station. Commands that involve interaction are: specify, execute, trace, stop_trace, and halt. Emulation can participate in coordinated measurements and can also begin execution of a program in concert with the initiation of a measurement. The operational commands, discussed earlier in this chapter, contain details for specify, execute, stop_trace, and halt. Details of measurement interaction possibilities appear in chapter 5 under the heading "Configuring for Interactive Measurements". Details of the trace command follow.

A shorthand syntax may be used when entering the information required by the <STATE> variable. The words "address", "data", and "status" can be omitted as long as commas are used to separate

the fields which contain the entries for each state. For example, "address 2000H data 00FFH status 14H" could be entered as follows: "2000H,00FFH,14H". Likewise, "address 2000H status 14H" could be entered as "2000H,,14H" using the shorthand syntax. Notice that when a particular field has no entry, commas must still be used to separate the fields. The first comma specifies the end of the address field, and the second comma specifies the end of the data field.

The trigger and qualifier parts do not have the entire syntax described above. Only one may have a range on address and only one may have an "or"ed term. The softkeys and grammar reflect this and will not allow entry of illegal specifications.

In all cases the term <VALUE> is an expression consisting of addition, subtraction, multiplication, division, parentheses, numbers, and symbols. In hexadecimal, binary, and octal numbers don't cares (X) may be used. They may not, however, be combined with arithmetic operations and may not be used in the address <VALUE> of a <RANGE_STATE>.

<STATUS_IDENT> is any one of the predefined mnemonic status values. Using "and" capability, status identifiers and/or values can be combined. It is possible, for example, to enter status 00000000B and status 11111111B; a combination that will result in the error message, "Status expression error".

The "and" function for status expressions operates bitwise on values entered, or on the predefined values of the mnemonic status identifiers. Table 7-1 defines the results of the "and" function for any bit.

Table 7-1. "And" Function Results

	X	0	1
X :	X	0	1
0 :	0	0	E
1 :	1	E	1

Where X is the symbol for a "don't care" bit, and E represents an invalid entry that will result in the message "Status expression error".

USING ANALYSIS COMMANDS

Analysis may be performed either by first initiating the program run and then specifying the trace parameters or by specifying the trace parameters first and then initiating the program run. In either case, once a trace command is initiated, the analysis module monitors the system buses of the emulation processor to detect the states specified in the trace command. When the trace specification has been satisfied, a message will appear on the status line showing "trace complete". At that time the contents of the trace memory can be displayed. If the trace memory contents exceed the page size of the display, the **(NEXT PAGE)**, **(PREV PAGE)**, **(ROLL UP)**, or **(ROLL DOWN)** keys may be used to display all the trace memory contents.

Trigger and storage qualification can be specified without initiating a trace by using the *specify trace* command, and traces can be initiated without altering the trigger and storage qualifications

by using the *execute* command. The *specify* and *execute* softkeys are available only with multi-module systems.

The trace command consists of the components described in the following paragraphs.

- a. **<TRIGGER>** - The "trigger" is the event on the emulation bus to be used as the starting, ending, or centering event for the trace.
- b. **<QUALIFIER>** - The storage specification determines which of the traced states will be stored in the trace memory for display upon completion of the trace. The trace memory can be filled by those states which occur immediately before or immediately after the specified trigger event, or half of the memory can be filled by states which precede the trigger and half by those which follow the trigger event. Events can be selectively saved by pressing trace only and entering the specific events to be saved. When this option is used, only the indicated states occurring in the specified position relative to the trigger are stored in the trace memory.
- c. **<COUNT>** - The count option specifies whether time or the occurrence of a state will be counted during the trace. The data can be displayed either "relative" to the count at the previous stored state, or "absolute" with respect to the trigger.
- d. **<BREAK>** - The break specification causes an exit from the executing program to the emulation monitor at a predetermined point in the emulation program.
- e. **again** - Entry of the "again" parameter causes the trace to be performed again using the previous trace parameters.
- f. **repetitively** - Entry of the "repetitively" parameter causes a new trace to be initiated after the results of the previous trace are displayed. The trace will continue until a stop_trace or a new trace command is issued.

Data Qualification

When emulating processors that permit both word and byte transfers, it may be necessary to include status qualification when tracing data values.

When byte fetches are captured by internal analysis, only the meaningful data is examined. Thus the command "trace about data 1234H" could capture word transfers of the the value 1234H, upper byte transfers of the 12H, or lower byte transfers of the value 34H.

Notice that the position of the value in the data specification is related to its physical location. To trace only high byte transfers of the value 12H, status qualification should be used to restrict storage to high byte accesses only, and data should be specified as 12\$\$H, where "\$" can be any hex digit or X since the lower digits will be ignored when tracing only high byte activity and function only as place holders.

NOTE

When using the 68008 emulator, the data is passed to the analyzer in 16 bit format. Therefore, it may be necessary to "qualify" your data as in the above discussion.

ANALYSIS MODES

While a program is running, the emulator places a copy of the executed states on the analysis bus. The internal emulation analyzer monitors the analysis bus and performs a trace by capturing the states from the analysis bus and storing them in its trace memory.

The illustrations in this text show traces made by the internal emulation analyzer. The traces were made using the same source program and the same trace specifications. The content of the traces is different. This difference is due to the analysis mode that was used when these traces were taken. The two kinds of analysis modes (`bus_cycle_data` and `execution_data`) are discussed in the following paragraphs.

BUS_CYCLE_DATA MODE

The `bus_cycle_data` mode of analysis performs a trace as each bus cycle occurs, and the emulator places a copy of every state that is produced by the microprocessor on the analysis bus. The analyzer stores each state in memory (including unused prefetches). The trace list shows every bus cycle that occurred, and it shows those cycles in the actual order of occurrence. This is the default mode of analysis.

EXECUTION_DATA MODE

The `execution_data` illustration shows a trace list that does not include every state in each bus cycle that was run. In this mode of measurement, the emulator does not place states on the analysis bus immediately as they occur. Instead, it holds the states in a queue while it watches program flow. Then the emulator places only the states that are executed by the program on the analysis bus. For example, prefetches of information that were actually used by the microprocessor are placed on the analysis bus. Prefetches of information that were later flushed (such as opcodes that were prefetched but not executed when the branch was taken) are not placed on the analysis bus. The emulator makes sure that each prefetch is used by the microprocessor before it places the associated state on the analysis bus and allows the analyzer to capture it.

Trace lists taken from microprocessors that pipeline information usually are hard to read. Prefetched information may appear in a trace list several bus cycles before it is actually used in the program. The emulator provides a trace list that is easier to read during the execution mode by rearranging the order of the states before it places them on the analysis bus (data operations are placed on the analysis bus following the opcodes that generated them and ahead of the next opcodes). See appendix F for the restrictions placed on this mode of analysis.

BUS_CYCLE_DATA TRACE DISPLAY

PRESS:

modify analysis_mode_to bus_cycle_data (RETURN)

trace after START (RETURN)

run from START (RETURN)

line#	address	opc/data	mnemonic	opcode or status	count:	time, relative
Trace: bus cycle data break: none count:						

after	002000	3E7C	MOVEA.W	#03FFCH,A7		
+001	002002	3FFC	supr	program read	<1.	uS
+002	002004	23F9	MOVE.L	0004000H,03500H	1.	uS
+003	002006	0000	supr	program read	1.	uS
+004	002008	4000	supr	program read	<1.	uS
+005	00200A	0000	supr	program read	1.	uS
+006	004000	0000	supr	data read	1.	uS
+007	004002	0001	supr	data read	<1.	uS
+008	00200C	3500	supr	program read	1.	uS
+009	003500	0000	supr	data write	<1.	uS
+010	003502	0001	supr	data write	1.	uS
+011	00200E	4EB9	JSR	0002500H	1.	uS
+012	002010	0000	supr	program read	<1.	uS
+013	002012	2500	supr	program read	1.	uS
+014	002500	2039	MOVE.L	0003500H,D0	1.	uS
+015	002502	0000	supr	data write	<1.	uS
STATUS: M680XX--Reset Trace complete _____12:35						
run from START						
___run___ ___trace___ ___step___ ___display___ ___modify___ ___break___ ___end___ ---ETC---						

Figure 7-5. Bus_Cycle_Data Trace Page 1

The display in figure 7-5 is page one of a two-page trace list taken in figures 7-5 and 7-6 with figures 7-7 and 7-8. The program information is identical. However, note the missing supervisor program reads, unused pre-fetches, and see how the trace data flows more smoothly with the execution_data mode on.

Press (NEXT PAGE) to see the next page of trace data.

Emulator/Analyzer 68000/68008
Command Summary and Syntax

```
Trace: bus cycle data           break: none           count:
line#  address opc/data mnemonic opcode or status      time, relative
-----
+016   003FF8   0000     supr data write                1.    uS
+017   003FFA   2014     supr data write                <1.   uS
+018   002504   3500     supr program read              1.    uS
+019   002506   D0B9     ADD.L  0004008H,D0             1.    uS
+020   003500   0000     supr data read                 <1.   uS
+021   003502   0001     supr data read                 1.    uS
+022   002508   0000     supr program read              1.    uS
+023   00250A   4008     supr program read              <1.   uS
+024   00250C   23C0     MOVE.L D0,0003504H             1.    uS
+025   004008   0000     supr data read                 <1.   uS
+026   00400A   0002     supr data read                 1.    uS
+027   00250E   0000     supr program read              1.    uS
+028   002510   3504     supr program read              1.    uS
+029   002512   4E75     RTS                            <1.   uS
+030   003504   0000     supr data write                1.    uS
+031   003506   0003     supr data write                1.    uS
+032   002514   0809     unused prefetch                <1.   uS
+033   003FF8   0000     supr data read                 1.    uS
+034   003FFA   2014     supr data read                 <1.   uS
```

```
STATUS: M680XX--Reset           Trace complete           _____12:35
```

run from START

```
__run__ __trace__ __step__ __display__ __modify__ __break__ __end__ ---ETC---
```

Figure 7-6. Bus_Cycle_Data Trace Page 2

The trace list in figure 7-6 shows page two of the trace made using the bus cycle mode of analysis.

EXECUTION_DATA DISPLAY

"*modify only_mode_to_execution_data* **RETURN**"

"*reset* **RETURN**"

"*trace after START* **RETURN**"

"*run from START* **RETURN**"

Trace: execution data			break: none	count:
line#	address	opc/data mnemonic opcode	or status	time, relative
after	002000	3E7C	MOVEA.W #03FFCH,A7	
+002	002004	23F9	MOVE.L 0004000H,03500H	1. us
+005	004000	0000	supr data read	1. us
+006	004002	0001	supr data read	1. us
+008	003500	0000	supr data write	1. us
+009	003502	0001	supr data write	1. us
+011	00200E	4EB9	JSR 0002500H	1. us
+014	003FF8	0000	supr data write	1. us
+015	003FFA	2014	supr data write	1. us
+016	002500	2039	MOVE.L 0003500H,D0	<1. us
+019	003500	0000	supr data read	1. us
+020	003502	0001	supr data read	1. us
+021	002506	D0B9	ADD.L 0004008H,D0	<1. us
+024	004008	0000	supr data read	2. us
+025	00400A	0002	supr data read	<1. us
+026	00250C	23C0	MOVE.L D0,0003504H	1. us
+029	003504	0000	supr data write	1. us

STATUS: M680XX--Reset Trace complete _____12:35

reset

__run__ __trace__ __step__ __display__ __modify__ __break__ __end__ ---ETC---

Figure 7-7. Execution Cycle Data Page 1

Press **(NEXT PAGE)** to display the next page of the trace data.

```

Trace: execution data                break: none                count:
line#  address opc/data mnemonic opcode or status                time, relative
-----
+030   003506  0003      supr data write                <1.    uS
+031   002512  4E75  RTS                                1.    uS
+032   003FF8  0000      supr data read                   1.    uS
+033   003FFA  2014      supr data read                   <1.    uS

STATUS: M680XX--Reset                Trace complete                _____12:35

reset

__run__ __trace__ __step__ __display__ __modify__ __break__ __end__ ---ETC---
  
```

Figure 7-8. Execution Data Trace Page 2

This execution_data trace list shows the same range of program activity that was captured in the preceding two bus_cycle_data trace lists. The following list is a cross-reference of the line# column in the execution_data display with the line# columns in the bus_cycle_data displays.

execution data	bus cycle data
+000	+000
+002	+002
+005	+006
+006	+007
+008	+009
+009	+010
+011	+011
+016	+014
+014	+016
+015	+017
+021	+019
+019	+020
+020	+021
+026	+024
+024	+025
+025	+026
+031	+029
+029	+030
+030	+031
+032	+033
+033	+034

Notice that the Trace line # column in the execution_data display does not show a complete sequence of numbers. The missing numbers indicate trace events which occurred but were not displayed. As an example:

```
bus_cycle_data
-----
after 002000 3E7C  MOVE.W #03FFCH,A7
+001  002002 3FFC      supr program read      <-- NOTICE THAT THIS LINE DOES
                                                NOT APPEAR IN THE EXECUTION_DATA
                                                LISTING. IT WAS USED TO CREATE
                                                THE MNEMONIC INSTRUCTION DISPLAYED
                                                IN LINE 0. THIS IS BECAUSE THIS MOVE.W
                                                IS A MULTIPLE WORD INSTRUCTION.
                                                THE ADDITIONAL READ OCCURRED TO FIND
                                                THE DATA THAT IS USED BY THE MOVE
                                                INSTRUCTION.
```

Notice the order of the line numbers in the bus_cycle_data trace lists. The order of the execution_data line numbers is not the same as the order of the bus_cycle_data line numbers. Note that that in the execution_data display the states now appear in the order in which they occurred. Note that the rearranging is done in 'real-time' with hardware as opposed to 'non-real time' with software. Therefore, using this mode does not add time to the execution of your code. The following lines appear in the bus_cycle_data displays but not in the emulation_data displays (the numbers of the lines may appear in the emulation_data displays, but the content of the bus_cycle_data lines does not appear):

```
+001 +003 +004 +007 +010 +012 +013 +017 +018 +022 +023 +027 +028
```

Line # +032 was thrown out because it was an unused prefetch.

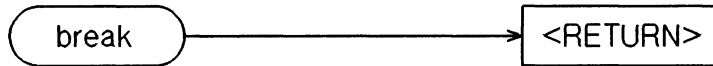
The states associated with these lines in the bus_cycle displays were found to be either unused by the program, or used to create an instruction, as in the above example. Therefore, the emulator never placed these states on the analysis bus.

All executed operands are not shown in the trace data unless they are a trace point (i.e. *trace after* START , the operand associated with START will be displayed.) If you wish to see all of the stored information, specify:

display trace status mnemonic

— break —

Syntax



Default Value

break

Example

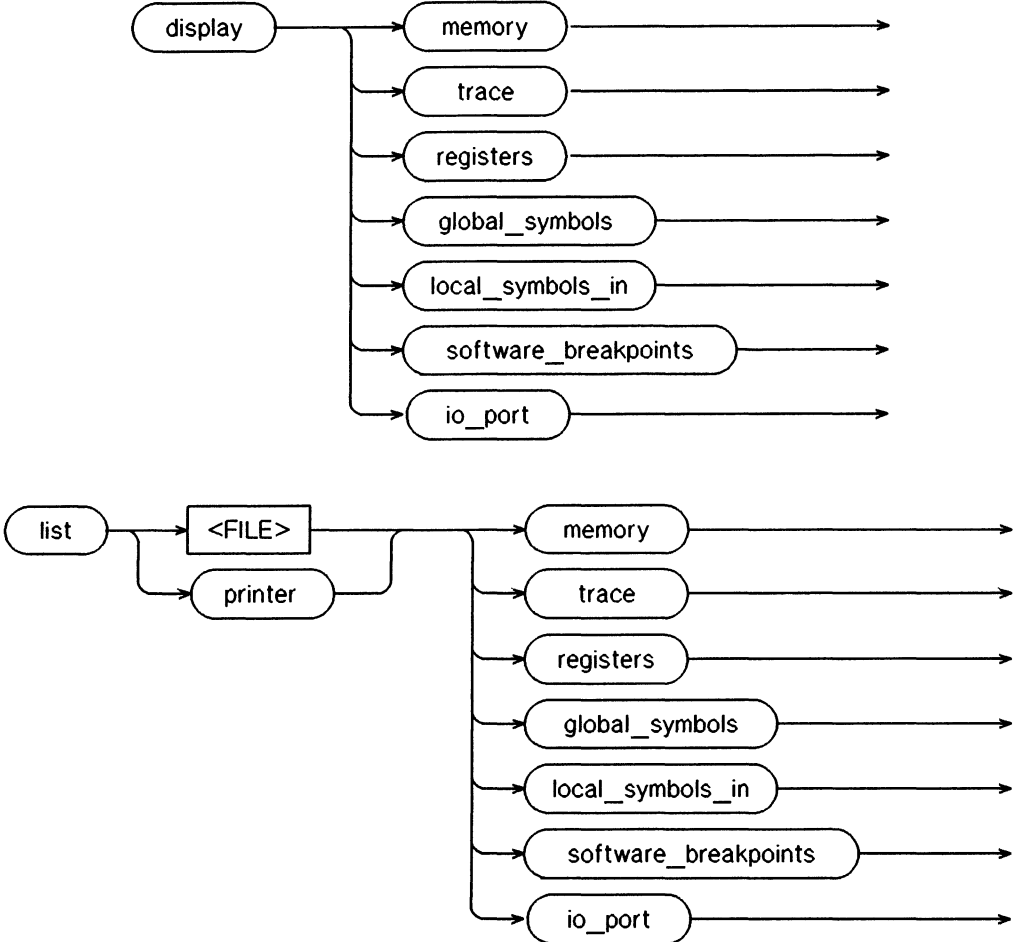
break

Function

Break causes the processor to be diverted from execution of the user program to the emulation monitor. See chapter 8 for details of the break function.

display/list

Syntax



Default Values

Depending on what is listed, defaults may be the options selected for the previous execution of the list or display command.

display/list

(Cont'd)

Function

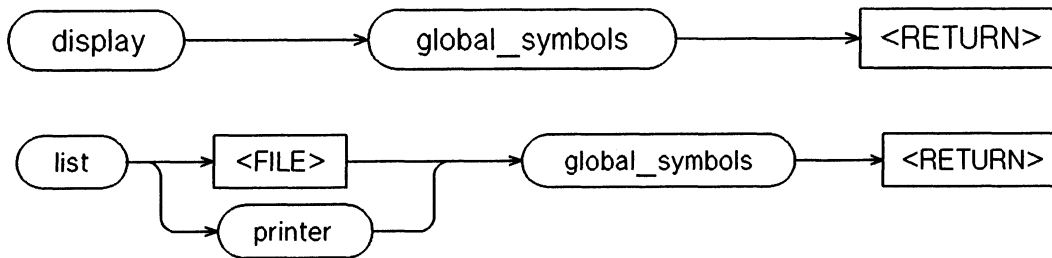
The list command produces a copy of the information selected. The display command displays the information and allows the use of the **ROLL UP**, **ROLL DOWN**, **PREV PAGE**, **NEXT PAGE**, and in some cases the **↑** and **↓** arrow keys. The copy resulting from a list command can be either a listing file stored in the HP 64000 memory or a hard copy produced by the printer. If the information is written to an existing file, the old file is overwritten by the new information.

Parameters

- | | |
|---------|--|
| printer | Printer causes a hard copy to be printed. |
| <FILE> | <FILE> causes the information to be copied to either a new or an existing file identified by <FILE>. The syntax for <FILE> is discussed in Appendix A. |

display/list global_symbols

Syntax



Default Value

none

Examples

```
display global_symbols  
list GLO global_symbols
```

Function

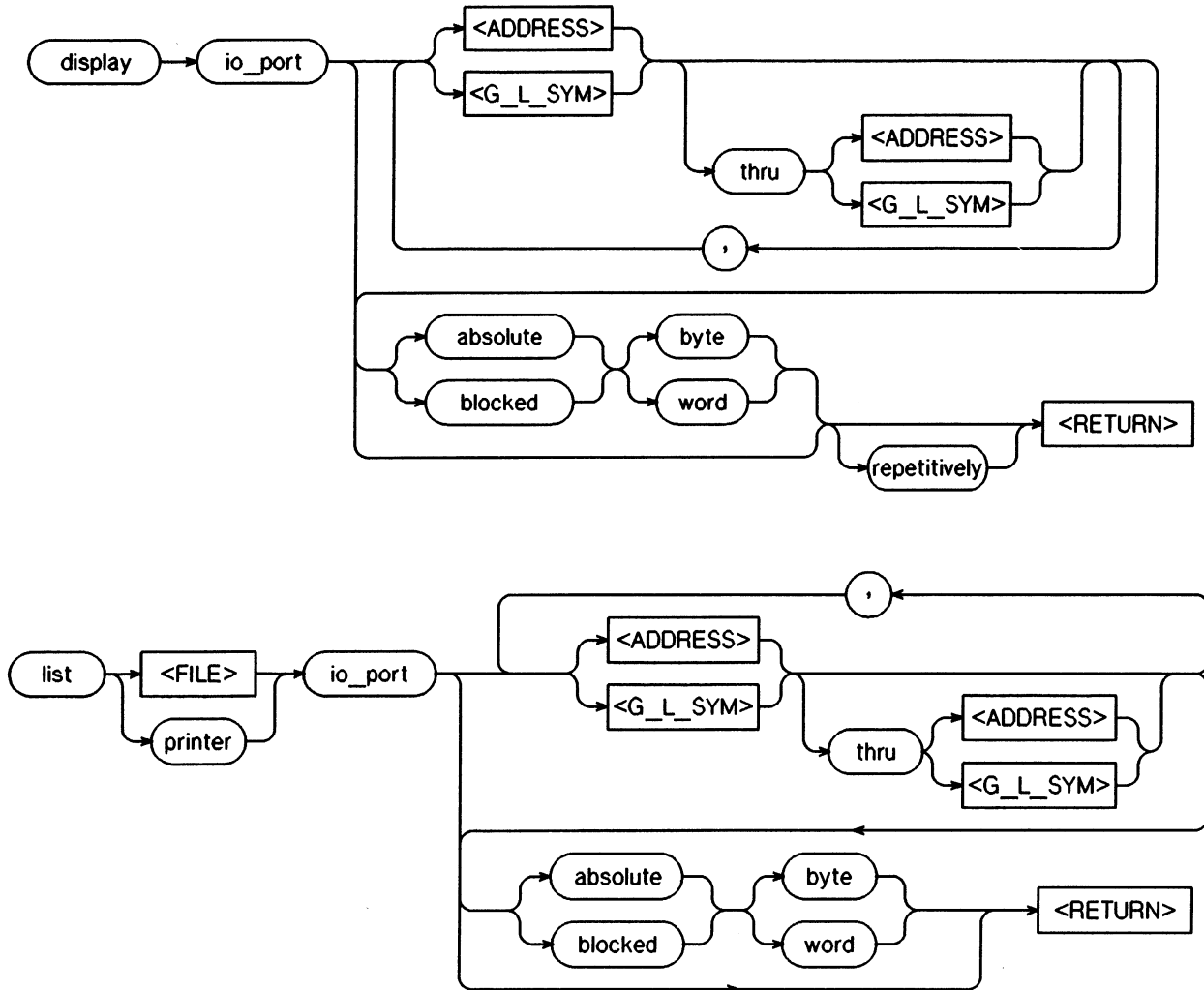
The `display/list global_symbols` command displays the global symbols defined for the current absolute file and the logical addresses and present values of those symbols. Global symbols are looked up in the `link_sym` file which is generated during linking. If the `link_sym` file is not present, no symbols may be displayed or used in expressions. Global symbols are those that are declared to be global in the source file. When the `list/display global_symbols` command is used, the listing will include the symbol name, address, and its present value. The present values are displayed for symbols in emulation memory. For other symbols, present values are displayed as "***".

Parameter

`global_symbols` `global_symbols` represents the symbols and labels defined as global in one of the source programs from which the current absolute file was generated.

— display/list io_port —

Syntax



<ADDRESS> may be a <SYMBOL>, <NUMBER>, or <EXPRESSION>.

<EXPRESSION> is of the form:

<ADDRESS> + <EXPRESSION>

<ADDRESS> - <EXPRESSION>

<ADDRESS> * <EXPRESSION>

<ADDRESS> / <EXPRESSION>

and includes the use of parentheses.

display/list io_port

(Cont'd)

Default Values

Address list defaults to previous list or to 0 if no value has been specified.

Display format defaults to last specified or to blocked.

Mode defaults to last specified or to byte.

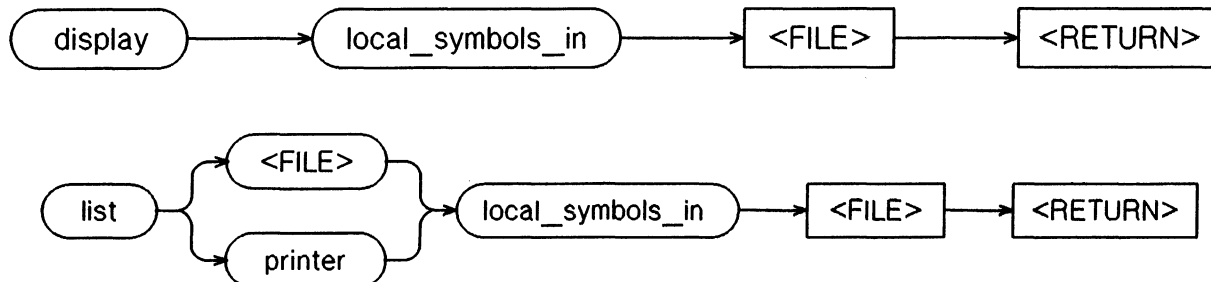
Repetitively is off unless specified.

Examples

```
display io_port blocked word 1,45,60 thru 80, 0FFFFH  
display io_port absolute byte 1,45,60 thru 80, 0FFFFH  
display io_port repetitively  
display io_port START thru OPEN_KEY  
display io_port OPEN_KEY thru START  
display io_port repetitively
```

— display/list loc_sym —

Syntax



Default Value

none

Examples

```
display local_symbols_in TEMP1  
list printer local_symbols_in TEMP1  
list BOB local_symbols_in TEMP1
```

Function

The display/list loc_sym command displays the local symbols and their present values and relative mode as defined in the source (program, data, or common) <FILE>. Local symbols are looked up in the asmb_sym file generated during assembly or compilation. If the asmb_sym file is not present, no local symbols may be displayed or used in expressions.

Address inputs may be either an absolute logical address, or a relative logical address. The logical address format is explained in detail in "Address Conventions" of chapter 6.

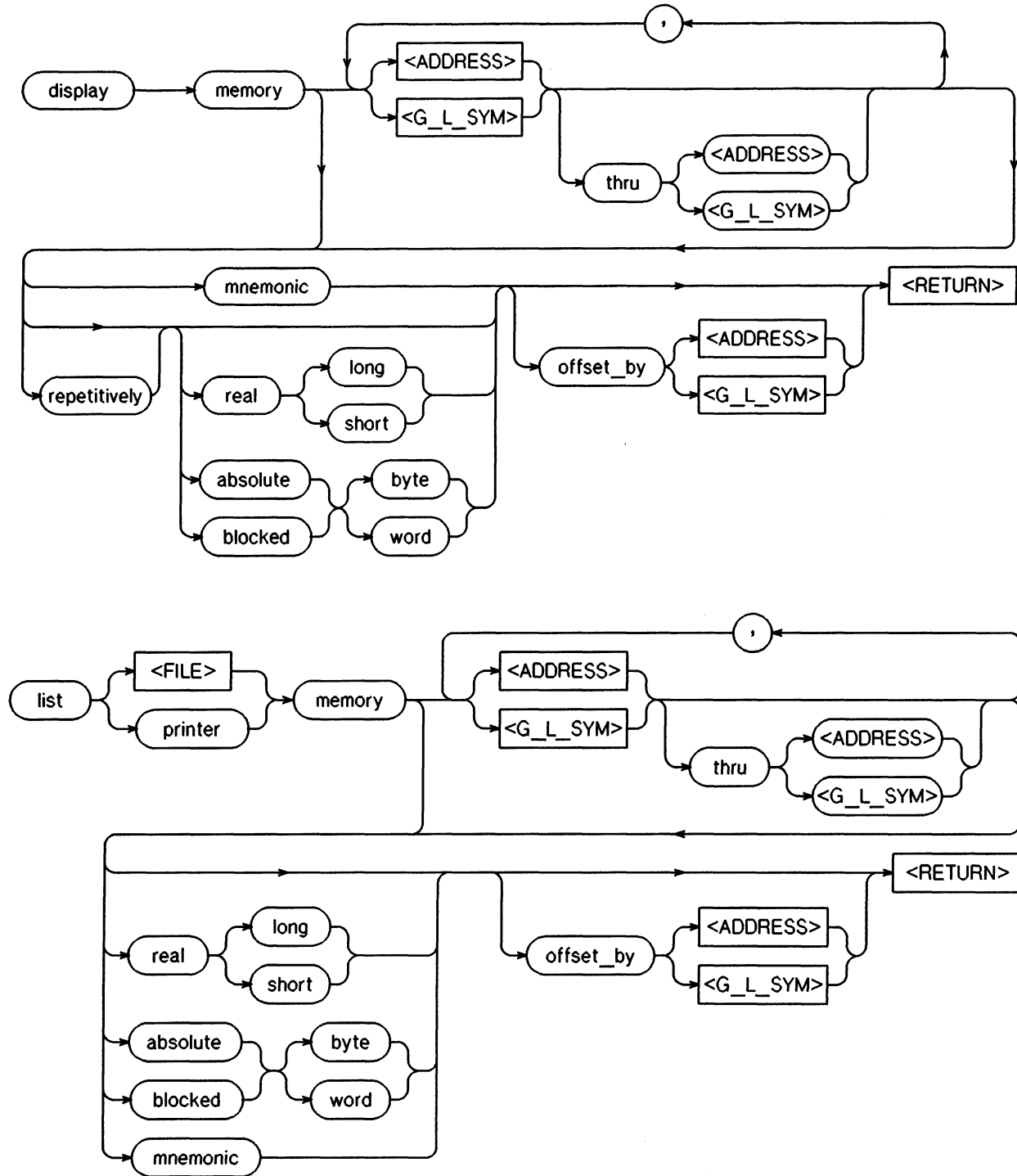
The present values are displayed for symbols in emulation memory. For other symbols, present values are displayed as "***".

Parameters

local_symbols_in	Local_symbols_in refers to the symbols and labels defined as local in the source file identified by <FILE>.
<FILE>	<FILE> represents the source file that contains the local symbols to be displayed. Refer to Appendix A for the syntax requirements of <FILE>.

display/list memory

Syntax



display/list memory

(Cont'd)

Default Values

Initial values are the same as specified by the command "display memory 0 blocked byte offset_by 0".

Defaults are to values specified in previous display or list memory command.

Repetitively must be specified each time display memory is issued.

Examples

display memory START *mnemonic*

display memory 0 *thru* 100H, START *thru* START+5,
500H, TARGET1, TARGET2 *blocked word*

list FILE *memory* 810H *offset_by* :MODULE1

Function

The display/list memory command shows the contents of the specified memory location or series of locations. The memory contents can be viewed either statically or repetitively (display memory only) and either in mnemonic or hexadecimal form. In addition, the memory addresses can be displayed offset by a value which allows the information to be easily compared to the file listing.

display/list memory

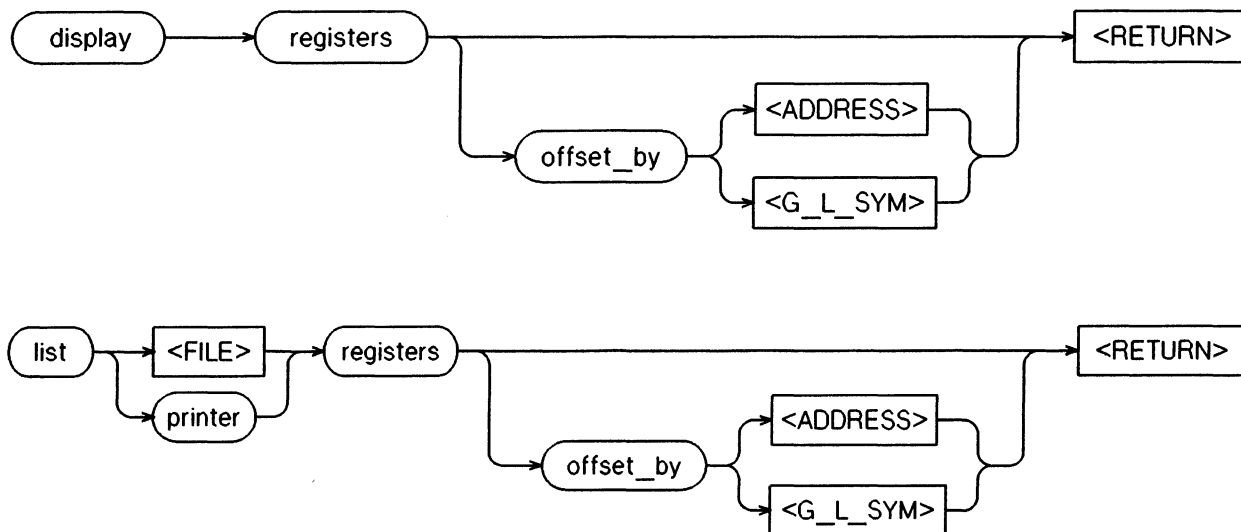
(Cont'd)

Parameters

<ADDRESS>	<ADDRESS> is the address of memory to be displayed. A single address, a list of single addresses, or ranges of addresses may be specified.
<G_L_SYM>	<G_L_SYM> is the name of a global or local symbol.
repetitively (display only)	repetitively causes the display to be periodically updated with the current contents of memory. The program must be interrupted in order to fetch the memory data and update the display (doing so one line at a time).
mnemonic	mnemonic causes the program in memory to be disassembled. The mnemonic opcodes, memory locations, and associated operands are then displayed or listed.
<OFFSET>	<OFFSET> causes the system to subtract the specified <OFFSET> from each of the actual absolute addresses before the addresses and the corresponding memory contents are displayed. The value of <OFFSET> can be selected such that each module in a program appears to start at address 0000H. The display/list of the memory contents will then appear similar to the assembly or compiler listing.

— display/list registers —

Syntax



Default Values

<OFFSET> - Initially 0; thereafter previous value.

Examples

display registers
display registers offset_by 810H
list REG registers offset_by 0A10H

Function

The display/list registers command gives the the current contents of the processor's registers, including the 8 data and 8 address registers, instruction counter (PC), system stack pointer (SSPT), and user stack pointer (USPT). If a step has just been executed, the mnemonic of the last instruction is also given. This process does not occur in real time. Therefore the system must be configured for nonreal-time operations if the registers are to be displayed while the processor is running.

The displayed value of the program counter can be offset from the actual value by a number which allows the register information to be easily compared to the assembled listing.

display/list registers

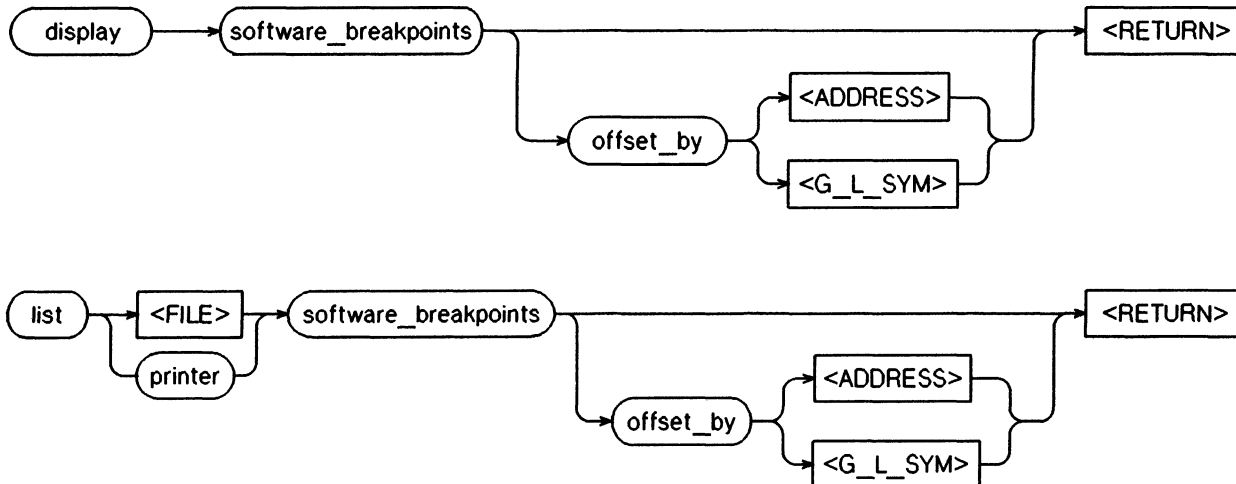
(Cont'd)

Parameters

<OFFSET> <OFFSET> represents the value by which the displayed program counter address is offset from the actual program counter address. The syntax for <OFFSET> is equivalent to the syntax for <VALUE> as described in Appendix A.

— display/list software_breakpoints —

Syntax



Default Values

Offset defaults to none, or to last specified default value.

Examples

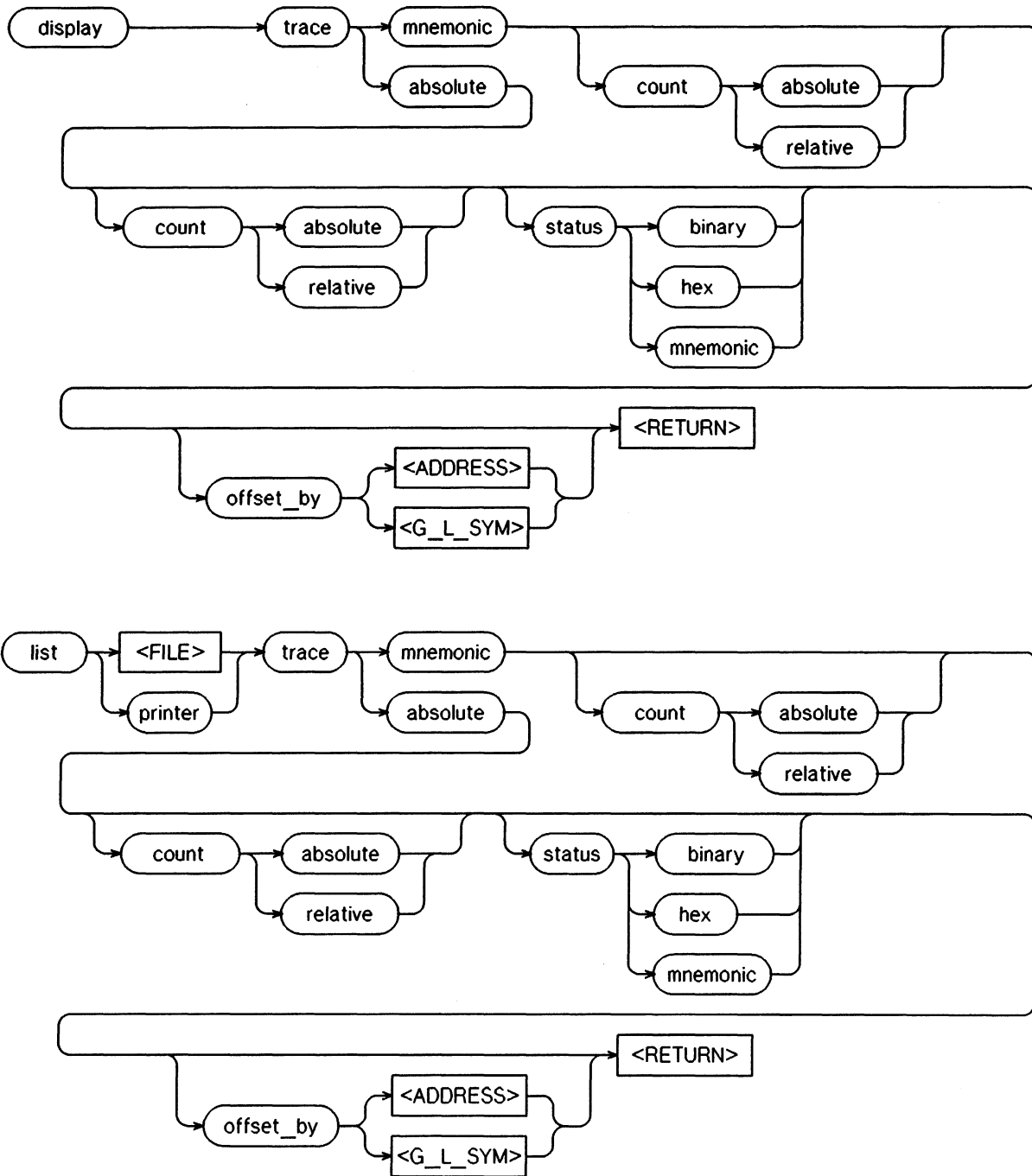
```
display software_breakpoints  
display software_breakpoints offset_by 2003H  
list BREAKFILE:TODD software_breakpoints  
list printer software_breakpoints offset_by START
```

Function

The display/list software_breakpoints command formats the entries and current status of the entered software breaks at the time the command is issued. If the emulation session is continued from a previous session, then this command will reflect the current state including the previously entered breakpoints. The column marked status shows either pending or inactivated. When in the pending state a breakpoint will cause the processor to enter the monitor program upon execution of that breakpoint, an appropriate message will be displayed and then the status of that breakpoint becomes "activated". Entries that show an inactivated status can be reactivated by a "modify software_breakpoints set" command.

display/list trace

Syntax



display/list trace

(Cont'd)

Default Values

Initial values are the same as specified by the command "display trace mnemonic count relative offset_by 0"

<OFFSET> - Initially 0; thereafter previous value.

Examples

```
display trace count absolute  
display trace status binary  
list EXEC trace count relative  
list printer trace offset_by 0100H
```

Function

The display/list trace command shows the contents of the trace buffer. The information can be presented as absolute hexadecimal code or in mnemonic form. The status captured by the analyzer can be displayed mnemonically, independent of the address and data information, or it can be displayed in hexadecimal or binary form.

The offset_by option causes the system to subtract the specified <OFFSET> from the addresses of the executed instructions before the trace is displayed. With an appropriate entry for <OFFSET>, each instruction in the displayed trace will appear as it does in the assembled or compiled program listing.

The display/list count command is used after a trace has been obtained to change the current display of time or state counts to one in which the counts are displayed either relative to the previous event or as an absolute count measured from the trigger event. If time counts are currently selected, the display count command causes an absolute or relative time count to be displayed. If the current display contains state counts, a relative or absolute state count results.

display/list trace

(Cont'd)

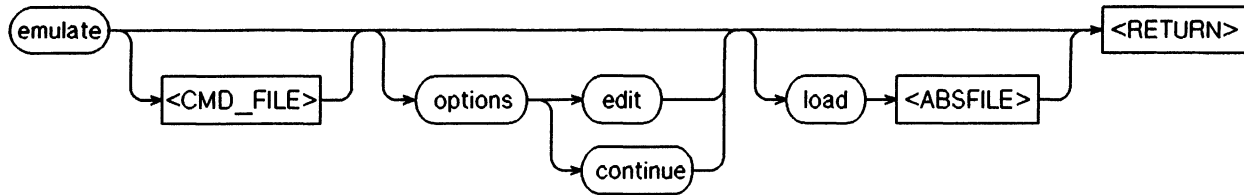
Parameters

mnemonic	mnemonic directs the system to display trace information with opcodes in mnemonic format.
absolute	absolute directs the system to display the status information rather than mnemonic opcodes.
status	
hex	displays status information in hexadecimal form.
binary	displays status information in binary form.
mnemonic	displays status information in mnemonic form.
<OFFSET>	<OFFSET> represents the number by which the address displayed for an executed instruction is offset from the instruction's actual address. The syntax for <OFFSET> is equivalent to the syntax for <VALUE> as described in Appendix A.
count	
absolute	absolute causes the state or time count for each event of the trace to be displayed as the total count measured from the trigger event.
relative	relative causes the state or time count for each event of the trace to be displayed as the count measured relative to the previous event.

— emulate —

For single module systems:

Syntax



Examples

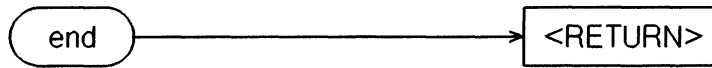
```
emulate  
emulate LOOP  
emulate LOOP load MUCH
```

Function

If no options are selected, emulation configuration is initiated and a new command file is constructed. If `<CMD_FILE>` is specified, an emulation session is initiated using the configuration specified by the command file. When a command file is specified, it is possible to continue a previous session. Or, if an altered configuration is needed, the `edit` option can be selected, allowing a new configuration by editing the previous one. Another option is specifying an absolute file to be loaded into emulation memory upon entry to the session.

end

Syntax



Default Value

end

Example

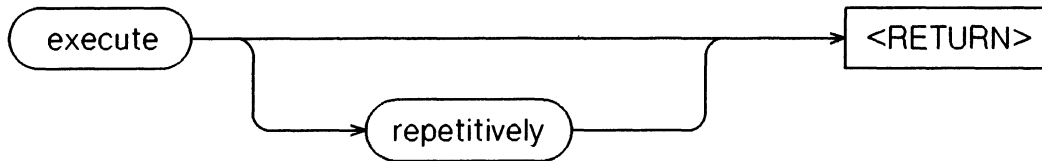
end

Function

The end command terminates the current emulation session and returns the HP 64000 operating system to the station monitor mode. The current states of the processor and trace are recorded in the emulation command file and a trace file of the same name. Emulation can then be resumed using the "emulate <CMDFILE> options continue" command. If emulation is terminated using the RESET key, emulation cannot be resumed, and the emulation command file is not overwritten. In a multiple_module system, the "end" command returns control to the measurement_system monitor program.

execute

Syntax



Examples

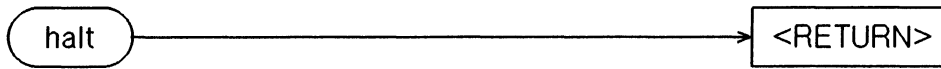
execute
execute repetitively

Function

Execute causes a measurement to begin. The 'execute' softkey label will be replaced with the 'halt' softkey label whenever a measurement is in progress. If emulation is participating in a system measurement, through cross-triggered analysis or the emulation start function (specify run), then the global measurement is initiated. Otherwise, a local measurement is begun and execute functions identically to "trace again", i.e., it executes a trace using the previous specification. A measurement can be executed repeatedly by issuing the execute repetitively command. This will restart the current measurement after each completion, until the user issues a halt command. A key feature of the execute command is that it will start all the modules participating in a system measurement when issued from any one of the modules. If an emulator is started as part of a measurement it will continue running and will not be started again by subsequent executions unless a specify run command is again issued. The 'execute' softkey is displayed only with multiple module systems.

halt

Syntax



Example

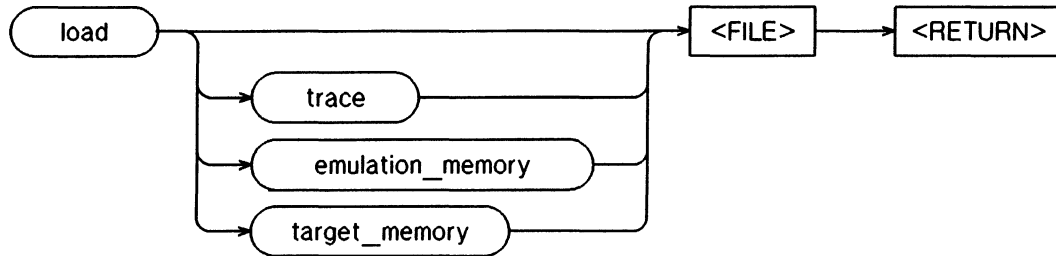
halt

Function

Halt causes the measurement currently executing to stop and turns off the repetitively option. The halt softkey is only displayed during execution in the place of the execute softkey. When the halt command is performed, some or all of the modules involved may have completed their measurement. Halt affects measurements caused by both trace and execute commands. If emulation is entered with a measurement in progress, halt will stop that measurement even if emulation is not interacting in the measurement. The 'halt' softkey is displayed only for multiple module systems.

load

Syntax



Default Value

all memory

Examples

load KW3000
load emulation_memory KW3000
load trace K5

Function

The load command transfers absolute code from the HP 64000 system disc into user RAM or emulation memory. The destination of the absolute code is determined by the memory configuration map which was set up during emulation configuration and the address specified during linking. Load trace allows the display command to access and display a previously stored trace. Load trace also allows execution of the trace specification via the trace again or execute commands.

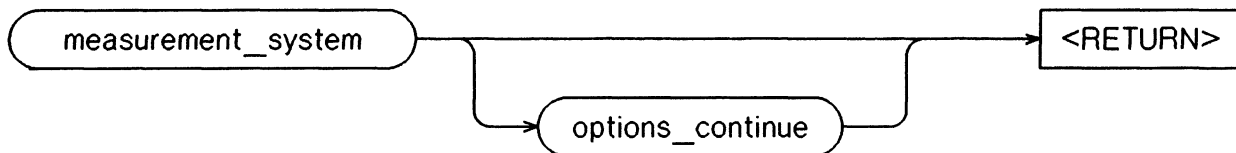
Parameters

<FILE> <FILE> is the identifier of the absolute file to be loaded from the HP 64000 system memory into user RAM or emulation memory or the trace file containing a previously stored trace specification. The syntax requirements for <FILE> are discussed in Appendix A.

measurement_system

For multiple module systems (using options continue):

Syntax



Function

The command "measurement_system" with "options continue" causes system operation to enter the measurement system monitor. The measurement system monitor coordinates and displays the interaction between the modules present and, in multiple module systems, controls entry to and exit from the individual modules of the system. Once in the monitor program, the emulator can be entered by issuing the command "eM680XX_S" , where "S" is the slot number of the emulation control board. The choice is made through the softkeys.

The "continue" option allows reentry to a previous session without disrupting a measurement in progress. If "continue" is not specified, all measurement system modules will be reset to their default configuration and any activity stopped. A "continue" is not possible under any of the following conditions:

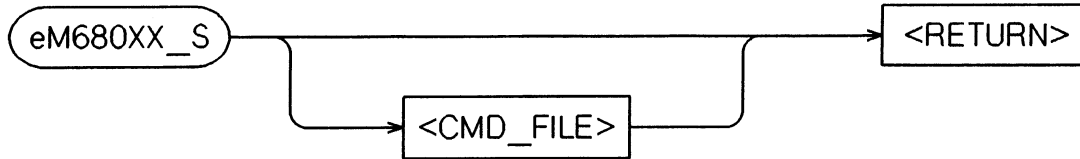
- a. Power has been cycled or the station reset by shift/reset.
- b. Performance Verification (option_test) has been initiated.
- c. The last session was exited by reset/reset.
- d. The measurement system configuration file is not present.

measurement_system

(Cont'd)

For multiple module systems (without using options continue):

Syntax



where "S" is the slot number of the emulator control board, and "<CMD_FILE>" is an optional emulation command file.

Default Value

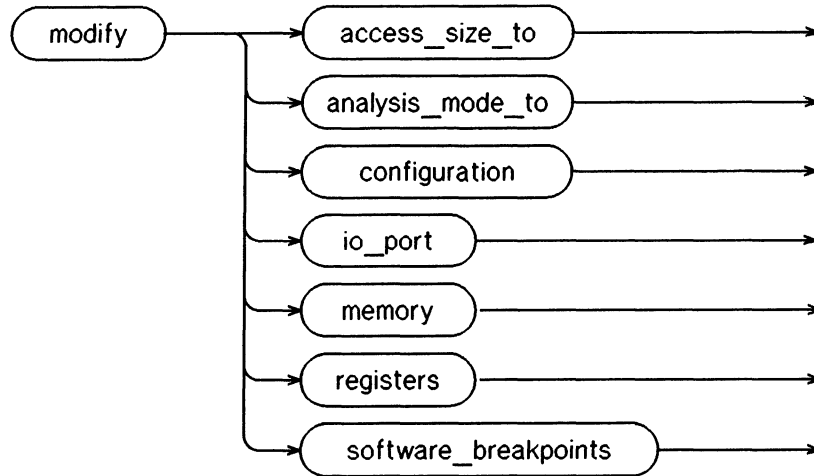
[<CMD_FILE>] The last specified command file.

Function

The emulate command, when issued from the measurement system monitor program, transfers action to the monitor program for the specified emulator. If no command file exists, or there is a conflict between the specified command file and the previous configuration, the emulation configuration questions are initiated and either a new command file is generated, or the specified file is edited.

modify

Syntax

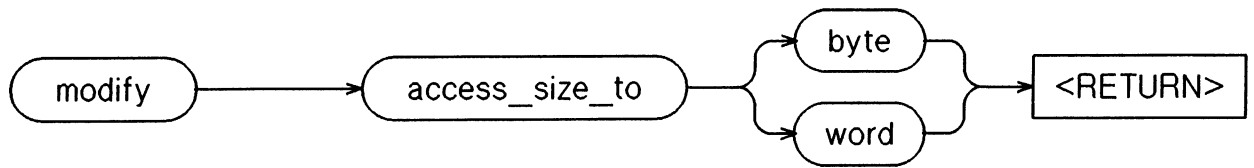


Function

The modify command is used to review or edit the configuration, to modify the contents of memory (as integers or as real numbers), to modify the contents of the processor registers, to write specified values to io_port addresses, or to set, reset, or clear software breakpoints.

— **modify access_size_to** (68000 only) —

Syntax



68000 ONLY

Default Value

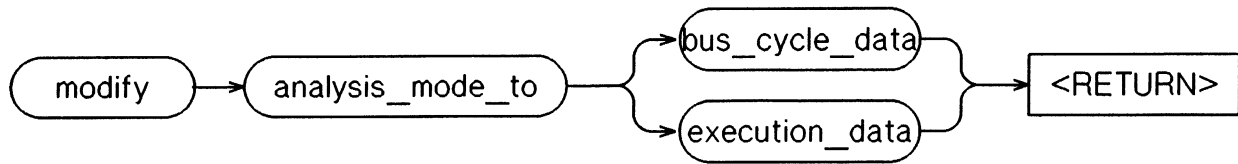
byte

Function

The modify access_size_to function will change the way that memory is accessed. The default is to byte.

modify analysis_mode_to

Syntax



Default Value

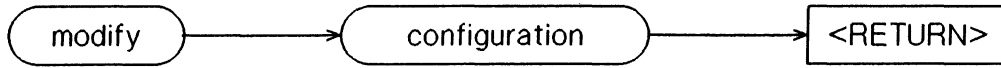
bus_cycle_data

Function

The modify analysis_mode_to function will allow the trace display to be in either the execution_data mode or the bus_cycle_data mode. Refer to the section on Analysis Modes in this chapter for more information on these modes of analysis.

— modify configuration —

Syntax



Default Value

none

Example

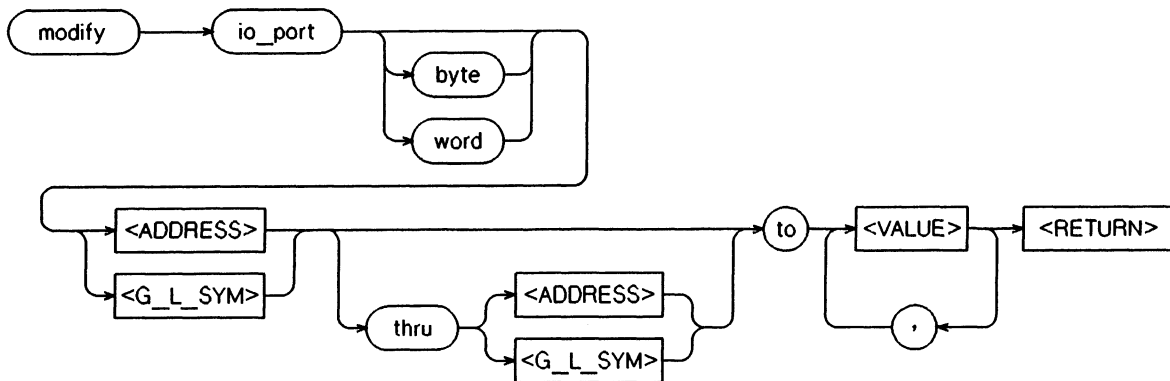
modify configuration

Function

The modify configuration command allows the current command file to be reviewed and edited. Each of the configuration questions is presented with the response previously entered. The prior response can be entered as displayed by pressing **RETURN**, or modified as necessary and then entered by pressing **RETURN**.

modify io_port

Syntax



Default Value

io_port: default is to byte mode.

Examples

```
modify io_port 0 to 12H
modify io_port word PRINTER to 0F3H
modify io_port byte DISPLAY thru DISPLAY+60 to 1,2,3,4,5,6
```

Function

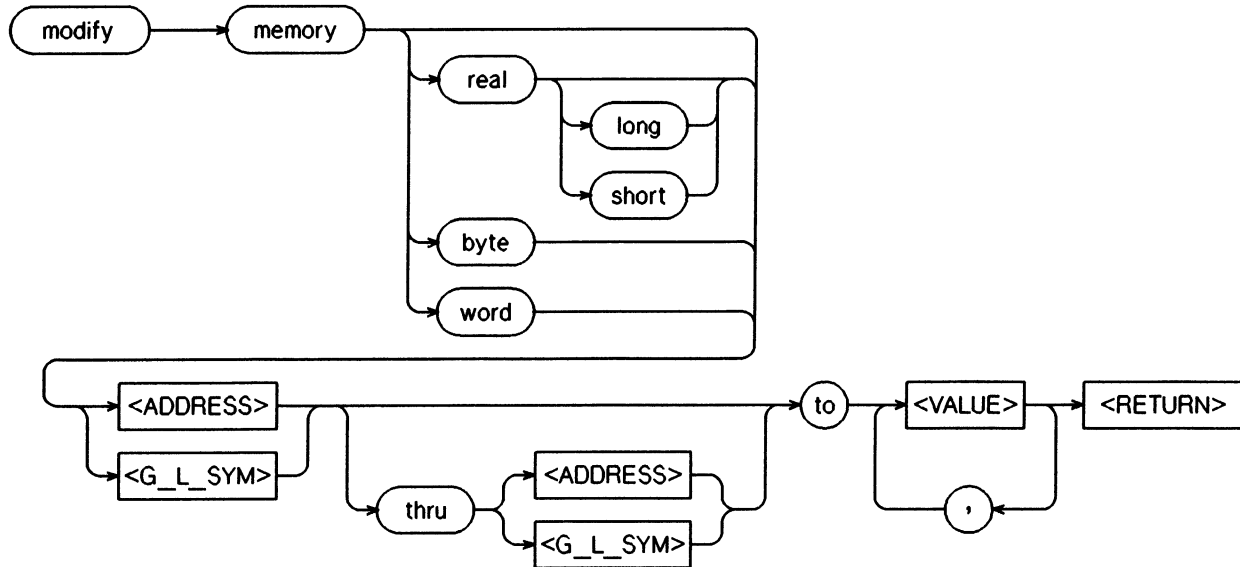
The modify io command causes io writes to a specified io address, or to a range of io addresses. The data may be written as bytes or words, and is specified as a single entry or a list of entries.

Parameters

- <ADDRESS> <ADDRESS> determines which I/O locations will be written to. <ADDRESS> is limited to the I/O address space of the processor, which is 0H thru 0FFFFH. The syntax is described in Appendix A.
- <VALUE> <VALUE> is the number that will be written to a specified io address. If byte mode is selected, only the least significant byte of each value will be used. The syntax is described in Appendix A.

— modify memory —

Syntax



Default Values

For integer memory modifications, initially default is to the display memory mode if in effect, otherwise default is to byte; thereafter default is to the display memory mode, or else to the last modify mode.

For real memory modifications, default is to the display memory mode if in effect, otherwise to short; thereafter default is to the display memory real mode if in effect, or to the last mode.

Examples

```
modify memory word 00A0H to 1234H
modify memory byte DATA1 to 0E3H,01H,08H
modify memory DATA1 thru DATA100 to 0FFFFH
modify memory byte ARRAY thru ARRAY+16 to 0,0FFH
modify memory real 0675H to -1.303
modify memory real long TEMP to 0.5532E-8
modify memory real short FIRSTREAL thru LASTREAL to
  1.11E1,2.22E-3,-4.56,9.99E17
```

modify memory

(Cont'd)

Function

The modify memory command can modify the contents of each memory location in a series to an individual value or the contents of all of the locations in a memory block to a single or repeated sequence of values.

Parameters

- <ADDRESS> <ADDRESS> determines which memory location or series of locations are to be modified. The term <ADDRESS> is explained in detail in the section "Overview of 680XX Memory Space", of chapter 5.
- <VALUE> <VALUE> is the number which is to be loaded into the specified memory location or locations. The syntax for <VALUE> is described in Appendix A. The syntax for real number values (<REALVAL>) is also described in Appendix A.

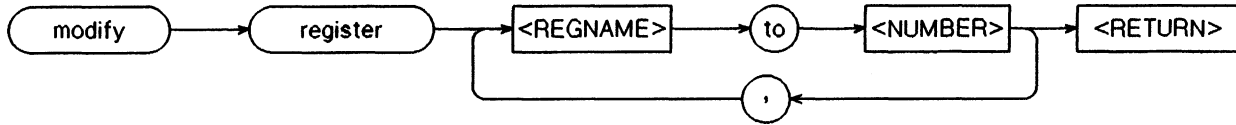
Description

A series of memory locations is modified by specifying the address of the first location in the series to be modified (<ADDRESS>) and the list of the <VALUE>s to which the contents of that location and the succeeding locations are to be changed. Both bytes must be addressed if a memory word is to be modified. The first <VALUE> listed replaces the contents of the specified memory location, the second <VALUE> replaces the contents of the next location in the series, and so on until the list has been exhausted. If only one number or symbol is specified, only the single address indicated is modified. When more than one <VALUE> is listed, the <VALUE> representations must be separated by commas.

An entire block of memory can be modified such that the contents of each location in the block is changed to the single specified <VALUE>, or to a single or repeated sequence. This type of memory modification is achieved by entering the limits of the memory block to be modified (<ADDRESS> thru <ADDRESS>) and the <VALUE> or list of values, <VALUE>, ..., <VALUE>, to which the contents of all locations in the block are to be changed.

— modify register —

Syntax



Default Value

none

Examples

modify register A1 to 39H
modify register B3 to 0AH, A3 to 00011XXB, USPT to 12345H

Function

The modify register command is used to modify the contents of one or more of the microprocessor's internal registers. The entry for <REGNAME> determines which register is modified.

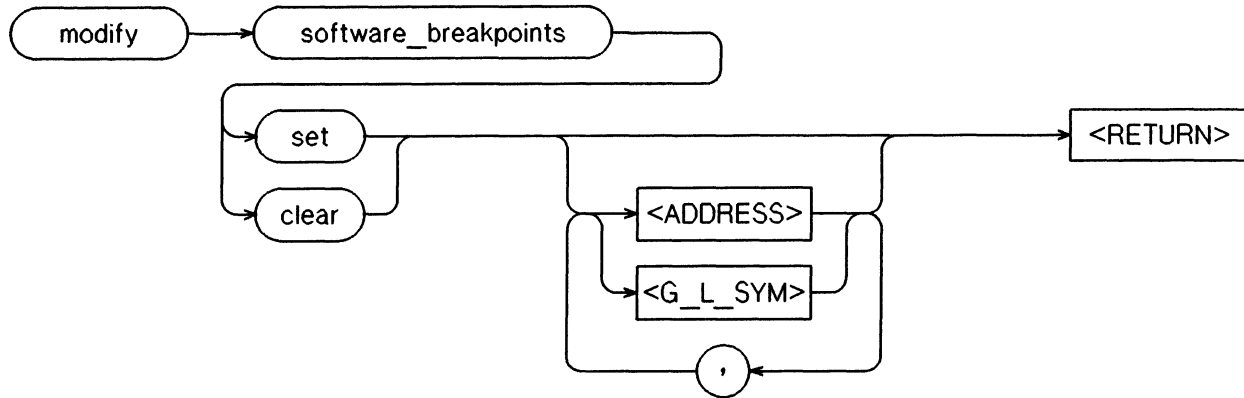
Register modification cannot be performed during real time running of the processor. A break must be performed to gain access to the registers.

Parameters

- | | |
|-----------|--|
| <NUMBER> | <NUMBER> is the number that is to be loaded into the specified register. The syntax for <NUMBER> is described in Appendix A. |
| <REGNAME> | <REGNAME> represents the name of the internal processor register to be modified. The possible entries for <REGNAME> are displayed on the register display. The valid register names are also listed in Appendix B 680XX Register Names and Format. |

modify software_breakpoints

Syntax



Default Values

software_breakpoints [set]:	will search through existing software_breakpoint list and reactivate all entries that are inactivated.
software_breakpoints [clear]:	entire software_breakpoint list will be deleted and memory restored to original values.

Examples

```
modify software_breakpoints set 1234H,0721H
modify software_breakpoints clear 99H,1234H
modify software_breakpoints set LOOP1END,LOOP2END,0EH
modify software_breakpoints clear
modify software_breakpoints set
```

Function

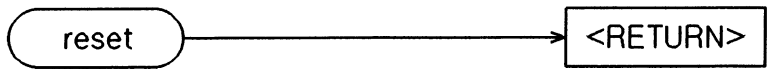
Software breakpoints make possible a "break on execution". Any valid address (number, label or expression) may be specified as a breakpoint. Valid addresses identify the first byte of valid instructions.

Operation of the program can be resumed after the breakpoint by either a "run" or "step" command.

See the section of chapter 8, titled "Software Breakpoints", for further details.

reset

Syntax



Default Value

none

Example

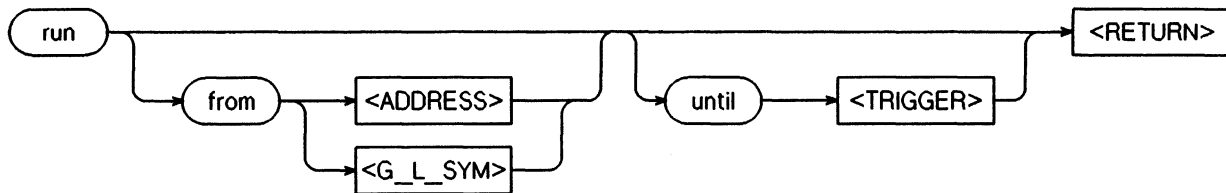
reset

Function

Reset suspends target system operation and re-establishes initial operating parameters, such as reloading control registers. The reset signal is latched when active, and is released by the "run" command.

run

Syntax



See the trace command syntax for a definition of <TRIGGER>.

Default Value

If the <ADDRESS> option is omitted, the emulator will begin program execution at the current address specified by the processor's program counter, or, if an absolute file containing a transfer address has just been loaded, execution will start at that address.

Examples

```
run  
run from 810H  
run from MONITOR_ENTRY  
run until 0AFFH  
run until 01FFH occurs 3
```

Function

If the processor is in a reset state, run will cause the reset to be released, and if a "from" address is specified the processor will be directed to that address. If the processor is running in the emulation monitor, the run command causes the processor to exit into the user program. The program can either be run from a specified <ADDRESS> or from the address currently stored in the processor's program counter, or from a label specified in the program.

run

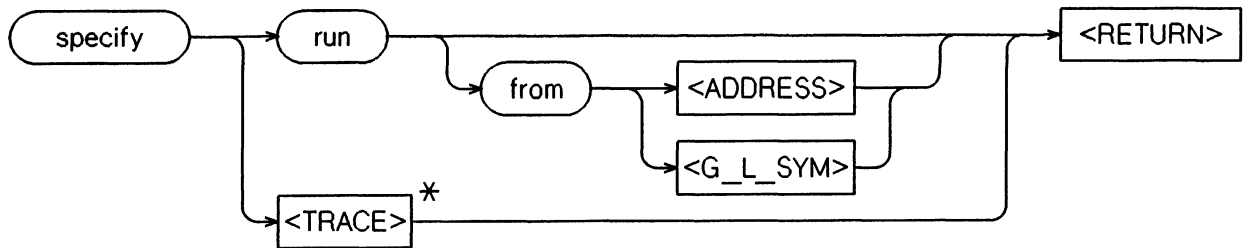
(Cont'd)

Parameters

- <ADDRESS> <ADDRESS> represents a state on the address bus which can be used to start a program run. The syntax requirements for <ADDRESS> are equivalent to those for <VALUE> as defined in Appendix A.
- <TRIGGER> The internal analysis causes a break from a user program to the emulation monitor when a state satisfying the <TRIGGER> term is encountered.

specify

Syntax



* See trace command syntax.

Examples

specify run from START
specify trace after address 1234H

Function

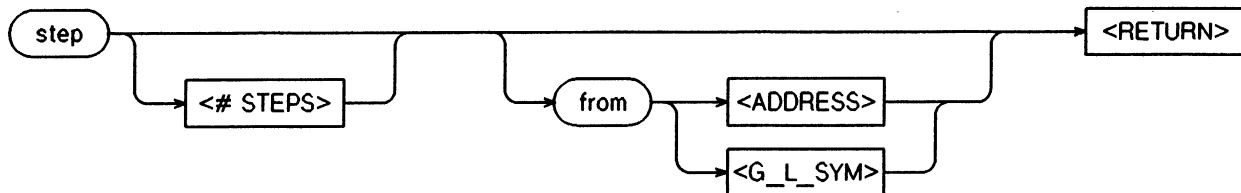
Specify is used to prepare a run or trace command for execution, and is used in conjunction with the execute command. If the processor is not reset, then specify run causes a break from a user program, and initializes the PC to the default address or to the specified address. An execute command will then cause the run to occur. Once an execution has occurred, the run specification is removed and can not be repeated without respecifying the run.

If the processor is reset and no address is specified, then an execute will cause the processor to run from the next condition. If the processor is reset from specified address, then the processor is allowed to run and the next program count is set up for the specified address.

Specify trace causes the trace hardware to be initialized with the given trace specification. An execute command will then cause the trace to be executed. A trace specification is not removed and can be reexecuted without another specify trace command. Specify trace and specify run can be used with a single execute command initiating both the run and the trace, but this mode can only be used if the internal analysis is configured to participate in a system measurement. If internal analysis is not configured, then specify trace and specify run are mutually exclusive and issuing one after the other will negate the first command. If specify trace is followed by execute, the effect is identical to trace. If specify run is followed by execute, the effect is the same as run, except that if a system measurement is configured, it is initiated. The 'specify' softkey label is displayed only with multiple module systems.

step

Syntax



Default Values

- <# STEPS>** If no value is entered for number of times, only one instruction is executed each time the RETURN key is pressed. Multiple instructions can also be executed by holding down the RETURN key.
- <ADDRESS>** If the from <ADDRESS> option is omitted, stepping begins at the next program counter address.
-

Examples

```
step  
step from START  
step 20 from 0A0H
```

Function

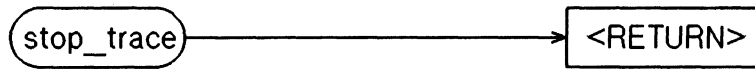
The step command allows program instructions to be sequentially analyzed by causing the emulation processor to execute a specified number of instructions. The contents of the processor registers, the contents of trace memory, and the contents of emulation or user memory can be displayed after each step command has been completed.

Parameters

- <# STEPS>** <# STEPS> determines how many instructions will be executed by the step command. The number of instructions to be executed can be entered in binary(B), decimal(D), octal(O, or Q), or hexadecimal(H) notation.
- <ADDRESS>** <ADDRESS> represents a state on the address bus which can be used to start a program step. The syntax requirements for <ADDRESS> are equivalent to those for <VALUE> as defined in Appendix A.
- <G_L_SYM>** <G_L_SYM> is the name of a global or local symbol in a file.

stop_trace

Syntax



Default Value

none

Example

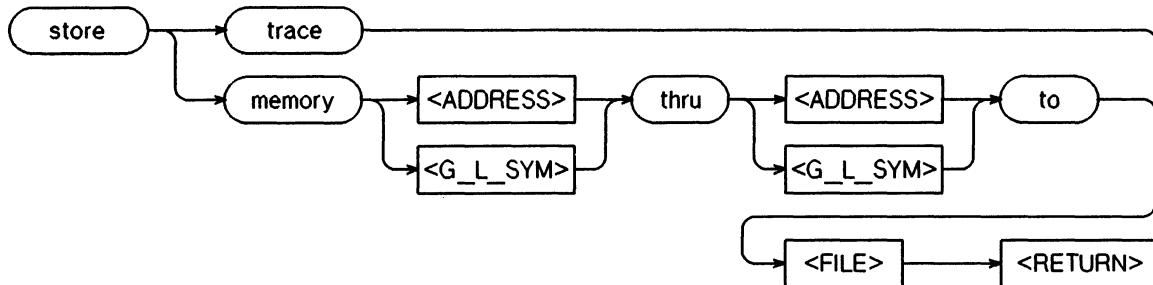
stop_trace

Function

The `stop_trace` command terminates the current trace, and stops the execution of the current measurement. That is, the system stops searching for trigger and trace states. Trace memory, although incomplete, can be displayed. `stop_trace` will also halt internal analysis if it is being used in "run until" mode.

store

Syntax



Default Value

none

Examples

store 800H *thru* 20FFH *to* TEMP2
store EXEC *thru* DONE *to* TEMP3
store trace TRACE

Function

The store command is used to store the contents of specific memory locations in an absolute file or the trace memory in a trace file.

Parameters

<ADDRESS> <ADDRESS> determines the memory locations from which data is to be stored into the specified absolute file.

<FILE> <FILE> is the identifier for the absolute file or trace file in which data is to be stored . The syntax requirements for <FILE> are described in Appendix A.

<G_L_SYM> <G_L_SYM> is the name of a global or local symbol in a file.

store

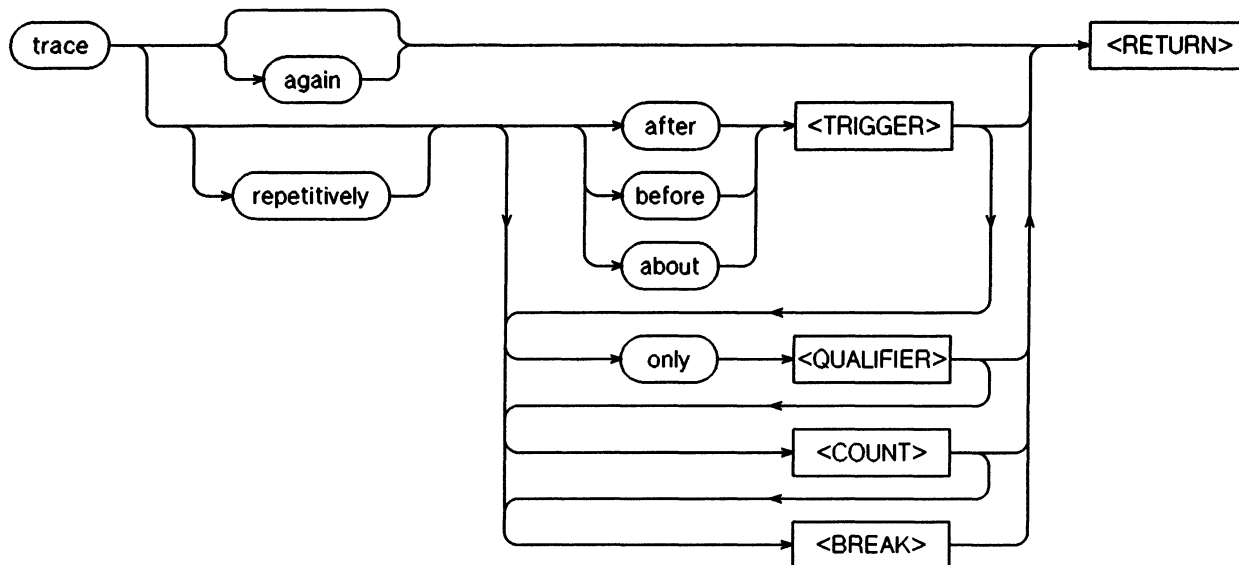
(Cont'd)

Description

<FILE> determines the name under which the absolute or trace file is to be stored. The store command creates a new file having the specified name as long as there is no absolute file presently on the disc with that name. In the cases where a file represented by the <FILE> variable already exists, the system asks whether the old file is to be deleted. If the response is yes, the new file replaces the old one. If the response is no, then the store command is canceled and no data is stored. Transfer address of absolute file is set to zero.

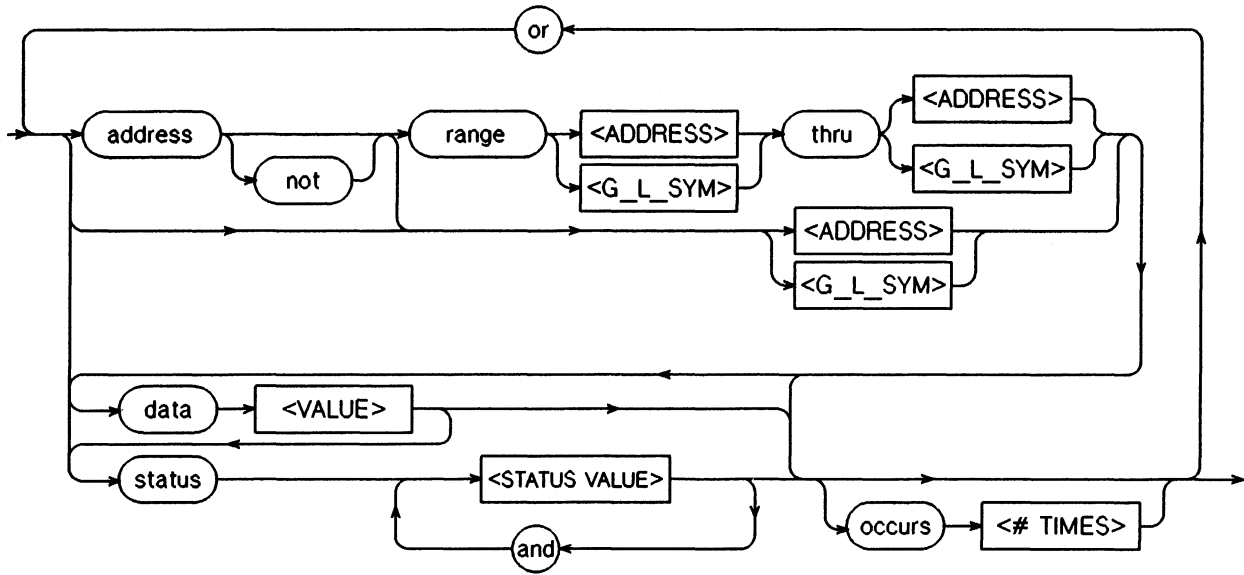
— trace —

Syntax



trace
(Cont'd)

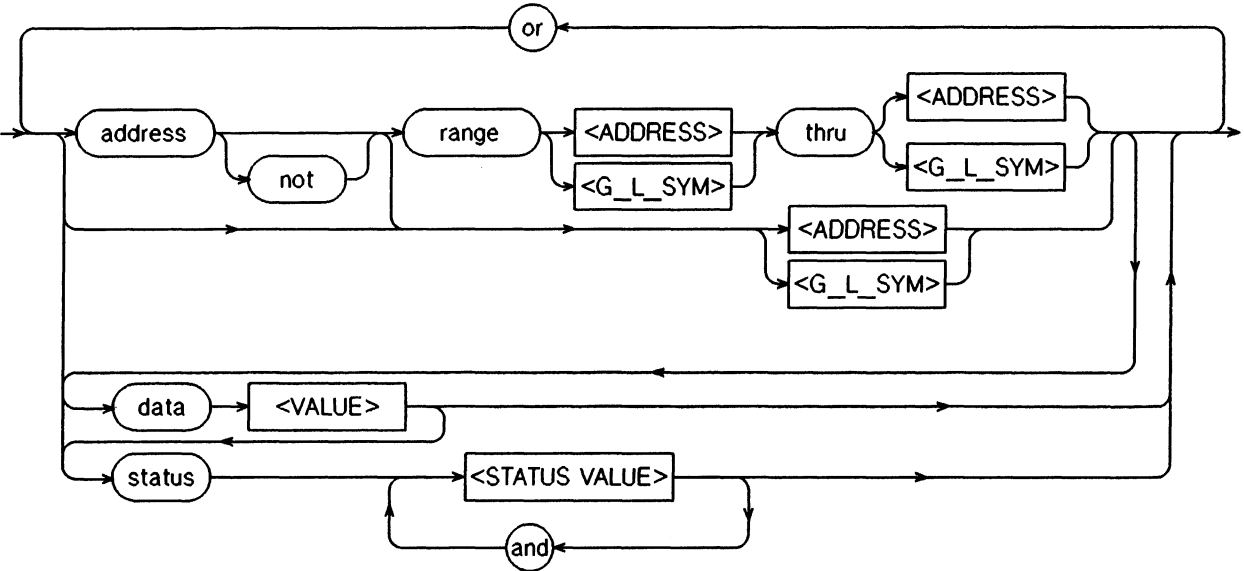
where <TRIGGER> is defined as:



trace

(Cont'd)

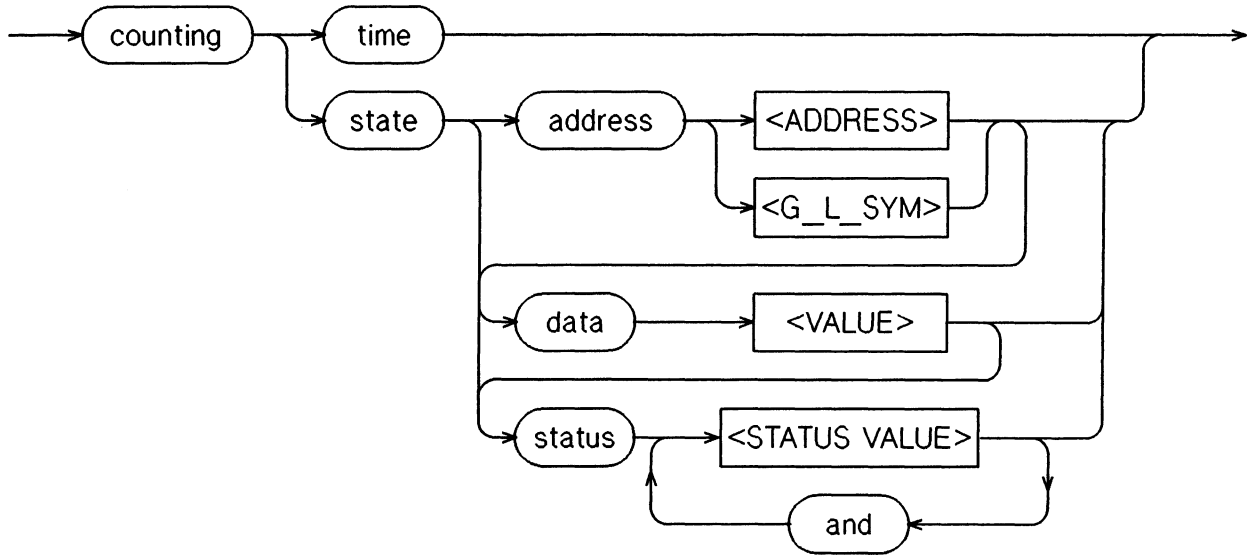
<QUALIFIER> is defined as:



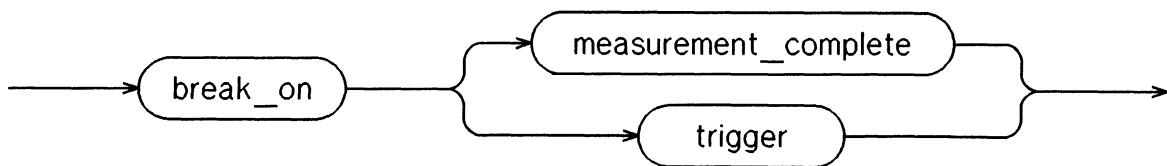
trace

(Cont'd)

<COUNT> is defined as:



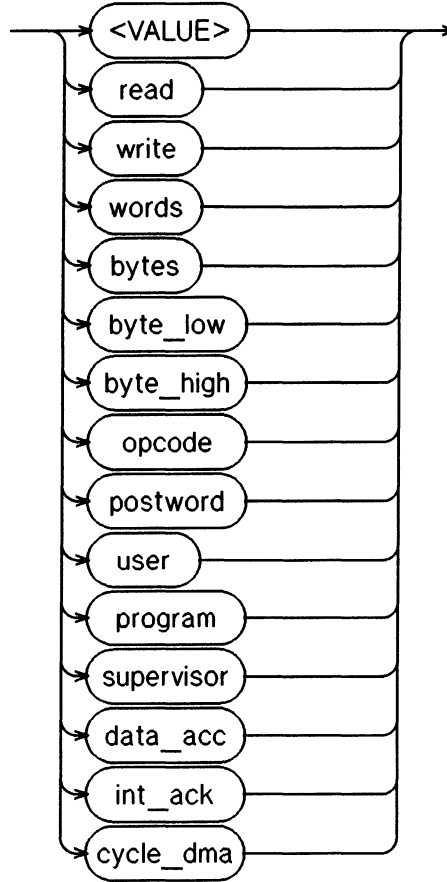
<BREAK> is defined as:



trace

(Cont'd)

<STATUS VALUE> is defined as:



trace

(Cont'd)

Default Values

trace any state

Examples

trace after START
trace only address range 1000H thru 1004H
trace counting state address 1004H
trace after address 1000H occurs 2 only address range 1000H thru 1004H
counting time break_on measurement_complete

Function

The trace command allows you to trace program execution using the internal analyzer.

Parameters

- <TRIGGER> The "trigger" is the event on the emulation bus to be used as the starting, ending, or centering event for the trace.
- <QUALIFIER> The storage specification determines which of the traced states will be stored in the trace memory for display upon completion of the trace. Events can be selectively saved by pressing the trace only and entering the specific events to be saved. When this option is used, only the indicated states are stored in the trace memory.
- <COUNT> The count option specifies whether time or the occurrence of a state will be counted during the trace. The data can be displayed either "relative" to the count at the previous store state, or "absolute" with respect to the trigger. All count measurements can be displayed in either absolute or relative mode. The absolute count is the total count from the trigger to each captured state. A plus sign (+) preceding the trace number indicates that the state occurred after the trigger state. A minus sign (-) indicates that the state has occurred before the trigger state. The "relative count" mode displays the count between consecutive states stored in the trace buffer.
- <BREAK> The break specification causes an exit from your target system executing program to the emulation monitor program at a predetermined point in the emulation monitor program.

trace

(Cont'd)

Parameters (Cont'd)

again Entry of the "again" parameter causes the trace to be performed again using the previous trace parameters.

repetitively Entry of the "repetitively" parameter causes a new trace to be initiated after the results of the previous trace are displayed. The trace will continue until a stop_trace or a new trace command is issued.

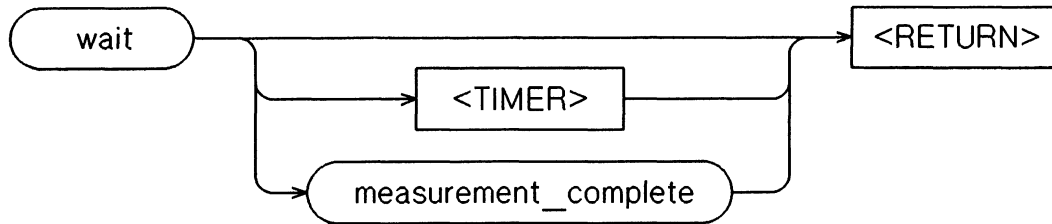
Description

Analysis may be performed either by first initiating the program run and then specifying the trace parameters or by specifying the trace parameters first and then initiating the program run. In either case, once a trace command is initiated, the analysis module monitors the system buses of the emulation processor to detect the states specified in the trace command. When the trace specification has been satisfied, a message will appear on the status line showing "trace complete". At that time the contents of the trace memory can be displayed. If the trace memory contents exceed the page size of the display, the NEXT PAGE, PREV PAGE, ROLL UP, or ROLL DOWN keys may be used to display all the trace memory contents.

Trigger and storage qualification can be specified without initiating a trace by using the specify trace command, and traces can be initiated without altering the trigger and storage qualifications by using the execute command.

wait

Syntax



Default Value

Waiting for any keystroke

Examples

<i>wait</i>	will wait for any keystroke before accepting the next command.
<i>wait 6</i>	will wait for any keystroke or 6 seconds before accepting the next command.
<i>wait measurement_complete</i>	will wait for any keystroke or for a pending measurement to become complete. If no measurement is in progress, wait will be satisfied immediately.

Function

The wait command is a delay command. Delay commands are enhancements that allow flexible use of system command files (although delays are also available outside of system command files). The usefulness of command delays lies in the capability to give the emulation system and target processor time to reach some condition or state before bringing in the next command. The delay commands may be included in the system command file.

Parameters

<TIMER> <TIMER> is the number of seconds you insert for your delay.

NOTES

Chapter 8

THE EMULATION MONITOR PROGRAM

OVERVIEW

Chapter 8 will:

- Describe the emulation break function.
- Describe the emulation monitor.
- Provide data for modifying the monitor to use Software Breakpoints.
- Provide data for customizing the emulation monitor.
- Describe the emulation monitor memory requirements.
- Describe the emulation monitor linking details.
- Provide a flowchart for the emulation monitor program.
- Provide a listing of the emulation monitor program source code.

GENERAL

In this chapter, whenever 680XX is mentioned (even as part of a filename) both the 68000 and the 68008 are assumed unless there are differences between the two processors. In that case, the differences will be noted. When the name 680XX appears in any program listings, just insert the appropriate name of your processor.

INTRODUCTION

A standard emulation monitor source file is supplied with each emulation system. This file must be assembled and linked by the user before the emulation monitor can be loaded and executed. The program supplied will enable all of the emulation system functions to operate properly after the file is assembled, linked and loaded.

In some cases it will be desirable to modify the emulation monitor to accommodate a particular target system or to expand the emulation monitor's capabilities. This is possible only if the basic communication protocol between the emulation monitor and the emulation software is maintained. A flow chart of the standard emulation monitor is given in this chapter. The emulation monitor program assembler code listing for the 680XX processor is given at the end of this chapter. A detailed description of the communication protocol and the standard emulation monitor is presented in the front of this chapter.

THE BREAK FUNCTION AND THE EMULATION MONITOR

The emulation software utilizes a program called the emulation monitor to perform many of the emulation system functions. The emulation monitor executes in a small part of the emulated processor memory space. The emulation software executed by the host processor communicates with the emulation monitor program via a dual port memory referred to as emulation memory. Before the emulation monitor can be executed, the currently executing target system program is interrupted using the break circuitry. Once the desired function is performed by the emulation monitor, the emulation system returns to the target system program.

The emulation break circuitry uses the NMI (INT7) resource of the processor to force the target system program to be interrupted and the emulation monitor to be entered. A break can be generated for an illegal memory reference, a bus condition that the analysis card detects, or a request by the emulation software.

EMULATION MONITOR

The emulation monitor is made up of five sections. The first section defines the vectors for the first eleven processor exception vectors. This section that defines the exception vectors is provided for your convenience and may be edited or deleted as appropriate. It is recommended that until you have developed your own vector look-up table, that you remove all of the provided vectors from comments as described in Chapter 3 entitled "Getting Started." Once you have developed your own table, then either 're-comment' or delete this table. Keep in mind, however, that the RESET vector for the 680XX processor must reside at addresses 00H thru 07H.

The second section defines the five entry points into the monitor.

"MONITOR_ENTRY" is the first entry point; this is the most used entry point as it is the "break" entry point, it expects that the program counter and status registers have been pushed onto the stack. The processor registers are saved in variables. These values are under the heading PREGS. The seven DATA registers D0-D7 are saved to DATA_0_7. The first six address registers, A0-A6, are saved to ADDR_0_6. Address register seven is saved in ADDR_7. The program counter is saved in two locations labeled PCL and PCH. The status register is saved to PSTATUS.

The processor interrupt mask can be restored to its pre-break value to continue target system interrupts while in the monitor. NOTE: This can only work if you modify the monitor program. If you desire to re-enable the interrupts, you will need to be at the level of softkeys that offers the *edit* softkey.

If the monitor program is not already in your USERID:

Press *copy* Mon_680XX:HP to Mon_680XX:YOURID

Press *edit* Mon_680XX:YOURID **RETURN** .

Under the JUMP_ENTRY label, you will find a commented section that describes re-enabling the interrupts. It will be necessary to comment the instruction "BRA SKIP_PRIV_INST" to use the interrupts while in the monitor. PRESS *end* **RETURN** to save your changes. It will also be necessary to re-assemble the monitor program.

After all of the registers are saved and the interrupt mask is restored, only if desired, then the program flow is directed to the command scanner (see the third section of the Monitor.)

The second entry point is "SPECIAL_ENTRY". This entry point is used to enter the Emulation Monitor from either a Bus Error or Address Error exception. The twenty five stack words that are unique to these exceptions are saved in variables STAT1 thru STAT25. Execution will continue at MONITOR_ENTRY.

The third entry point is called "JSR_ENTRY". It expects that only a return program counter be put on the system stack. You may link to this entry point if a "soft-break" into the monitor is desired. Be careful to jump to this subroutine only while in the supervisor mode. If not in the supervisor mode a "Privileged-instruction" exception will occur, thereby interrupting the flow of your program. Execution will continue at MONITOR_ENTRY.

The fourth entry point into the monitor is used when entering the monitor from a "RESET" exception. (assuming that you have elected to have the RESET vector directed to the monitor). RESET_ENTRY will set all of the 680XX registers to the default values.

The fifth and final entry point into the monitor is SWBK_ENTRY. This is the entry point into the monitor program when the 680XX processes a software breakpoint, i.e. a "trap 15" placed into memory by the HP 64000 system.

The third section of the emulation monitor is the command scanner. The scanner normally rests in an idle loop entitled "MONITOR_LOOP." The system global "MONITOR_CONTROL" will be examined, and if bit 15 is a zero, the idle loop is resumed; if bit 15 is a one, a command is present. Bit 15 of MONITOR_CONTROL is set by the Host, it is cleared by the monitor program. The lower byte of "MONITOR_CONTROL" contains a command number against which the command table is compared. If a match is found, a command entry point will be retrieved from the table and the command will be executed. If a match is not found, the program will return to the idle loop. The command is considered complete when bit 15 of MONITOR_CONTROL is set to a zero.

The fourth section contains the Emulation Monitor command execution command modules. The basic monitor contains three commands; "ARE_THERE", "EXIT", and "ACCESS_USER_MEMORY." "ARE_THERE" is used by the host (the HP64000) to determine whether the processor is executing in the monitor or in the target system code. It can also pass an ASCII message to be displayed on the host system status line. "EXIT" will reload the processor registers from the variables that they were stored in prior to entering the monitor. The program will then exit the monitor and return to the target system code. "ACCESS_USER_MEMORY" moves data between the monitor parameter block areas and target system memory. This command is used to modify and display target system memory.

An optional monitor command is the user defined IO_FUNCTION. Since the 680XX does not have a separate I/O space, use of the modify/display io_port command is made possible thru the monitor program.

More details are given on the IO_FUNCTION in the source listing of monitor at the end of this chapter and, also, in the section entitled "Customizing the Emulation Monitor."

MODIFYING THE MONITOR TO USE SOFTWARE BREAKPOINTS

The emulation monitor program comes from the factory with the software breakpoint table provided but, it is within comments. This is done because there are sixteen possible trap vectors to choose from and you will need to 'un-comment' the one that you wish to use.

PRESS *edit* Mon_680XX:YOURID **RETURN**

Use the **ROLL UP** **ROLL DOWN** **NEXT PAGE** **PREV PAGE** keys to find:

```
*****
*   TRAP VECTORS ( SOFTWARE BREAKPOINT VECTORS ) CHOOSE ONE
*   -----
*   The TRAP vector you should choose for the software breakpoint
*   depends on the trap # chosen in the configuration question.
*****
*   ORG 080H
*   DC.L SWBK_ENTRY      ;TRAP #0
*   ORG 084H
*   DC.L SWBK_ENTRY      ;TRAP #1
*   ORG 088H
*   DC.L SWBK_ENTRY      ;TRAP #2
*   ORG 08CH
*   DC.L SWBK_ENTRY      ;TRAP #3
*   ORG 090H
*   DC.L SWBK_ENTRY      ;TRAP #4
*   ORG 094H
*   DC.L SWBK_ENTRY      ;TRAP #5
*   ORG 098H
*   DC.L SWBK_ENTRY      ;TRAP #6
*   ORG 09CH
*   DC.L SWBK_ENTRY      ;TRAP #7
*   ORG 0A0H
*   DC.L SWBK_ENTRY      ;TRAP #8
*   ORG 0A4H
*   DC.L SWBK_ENTRY      ;TRAP #9
*   ORG 0A8H
*   DC.L SWBK_ENTRY      ;TRAP #10
*   ORG 0ACH
*   DC.L SWBK_ENTRY      ;TRAP #11
*   ORG 0B0H
*   DC.L SWBK_ENTRY      ;TRAP #12
*   ORG 0B4H
*   DC.L SWBK_ENTRY      ;TRAP #13
*   ORG 0B8H
*   DC.L SWBK_ENTRY      ;TRAP #14
*   ORG 0BCH
*   DC.L SWBK_ENTRY      ;TRAP #15
*****
```

remove the "*" from the trap that you intend to use. The number of the trap that you 'un-comment' must match the number selected when configuring the emulator. Make sure that you remove the trap from the "ORG" statement as well as the "DC.L" statement. When you are finished;

PRESS *end* (RETURN) to save your changes.

It will be necessary to re-assemble and re-link the monitor if you have not been using this feature and now wish to do so. See chapter 9 for a more detailed description of how Software Breakpoints function.

CUSTOMIZING THE EMULATION MONITOR

The emulation monitor supplied with the subsystem will allow all of the emulation features to operate in most systems. Some systems, however, will require modification to the emulation monitor program in order to maximize the effectiveness of the emulation subsystem. For this reason, the source program for the monitor has been provided and is thoroughly commented. Within the code, each of the standard routines has been discussed to allow you to easily make your modifications.

For example; You may find yourself constantly moving a block of memory from one location to another. Rather than continually calling a subroutine to do this for you, make this feature a part of the monitor program. Supplied with the source code is an example of how to read or write to blocks of memory within the emulation system. Combined with the command that allows access to user memory (Commands 2&4), you could make this feature work. Of course you are free to write your own code to accomplish this as long as you follow the established guidelines listed in the monitor for using the IO_FUNCTION. Also included, is a listing of a modified IO_FUNCTION that allows you to set up a new software break vector and/or restore the old vector. Note the unique way that this program uses the address field of the modify command to pass a command, and uses the data field to pass a parameter. This program also takes advantage of the MONITOR_MESSAGE function to display an ASCII command on the HP 64000 status line that indicates when these functions have been implemented. (See figure 8-1).

NOTE

At no time may your customized portion of the monitor exit the monitor program. Doing so will cause the entire system to become unsynchronized and, therefore, unusable.

Please note, however, that it is not recommended that you make any changes to any portions of the monitor other than to the user defined IO_FUNCTION. Any changes in other sections of the monitor may cause some features to stop working due to modifications on the stack, or because the information that is passed to and from the various sections has been tampered with.

Emulator/Analyzer 68000/68008
Emulation Monitor Program

```

*****
*  COMMAND 3 ... USER DEFINED IO FUNCTION
*
*  THE USER DEFINED IO FUNCTION ALLOWS THE USER TO IMPLEMENT
*  SPECIALIZED DISPLAY/MODIFY FEATURES.  THE FEATURES ARE ACCESSED
*  FROM THE 64000 KEYBOARD BY ENTERING:
*      display io_port [ADDR_LIST]
*  OR  modify io_port [ADDRESS] to [VALUE_LIST]
*  OR  modify io_port [ADDRESS] thru [ADDRESS] to [VALUE_LIST]
*
*  IN RESPONSE TO THIS INPUT THE 64000 WILL WRITE THE FOLLOWING
*  COMMAND BLOCK:
*  MONITOR_CONTROL+0 word  8003H
*  MONITOR_CONTROL+2 word  bit0 : 0 = read, 1 = write
*                          bit1 : 0 = byte, 1 = word
*  MONITOR_CONTROL+4 word  byte address to be used, 16 bits only
*  MONITOR_CONTROL+6 word  data {if byte operation data right
*                               justified}
*
*  WHEN THE COMMAND IS COMPLETE, THE 64000 EXPECTS ONE OF THE
*  FOLLOWING RESPONSES:
*  MONITOR_CONTROL = 0000H = successful operation
*                   0003H = failure
*
*  AS AN EXAMPLE OF WHAT CAN BE DONE WITH THIS COMMAND, THE
*  FOLLOWING CODE WILL READ/WRITE MEMORY IN THE LOWEST AND HIGHEST
*  32K BYTE BLOCKS OF MEMORY.
*  EITHER WORD OR BYTE ACCESSES CAN BE DONE.
IO_FUNCTION
    MOVE.W    MONITOR_CONTROL+4,A0 ;MOVE ACCESS ADDRESS TO REGISTER.
*                               ;NOTE, ADDRESS IS SIGN EXTENDED.
    BTST     #0,MONITOR_CONTROL+3 ;TEST FOR R/W OPERATION.
    BNE      IO_WRITE
IO_READ
    BTST     #1,MONITOR_CONTROL+3 ;TEST FOR BYTE OR WORD OPERATION.
    BNE      IO_READ_WORD
IO_READ_BYTE
    MOVE.B   [A0],MONITOR_CONTROL+7
    JMP      LOOP_REENTRY        ;GO BACK AND WAIT FOR NEXT COMMAND.
IO_READ_WORD
    MOVE.W   [A0],MONITOR_CONTROL+6
    JMP      LOOP_REENTRY        ;GO BACK AND WAIT FOR NEXT COMMAND.
IO_WRITE
    BTST     #1,MONITOR_CONTROL+3 ;TEST FOR BYTE OR WORD OPERATION.
    BNE      IO_WRITE_WORD
IO_WRITE_BYTE
    MOVE.B   MONITOR_CONTROL+7,[A0]
    JMP      LOOP_REENTRY        ;GO BACK AND WAIT FOR NEXT COMMAND.
IO_WRITE_WORD
    MOVE.W   MONITOR_CONTROL+6,[A0]
    JMP      LOOP_REENTRY        ;GO BACK AND WAIT FOR NEXT COMMAND.

```

```
*****
* MODIFIED TO USE ADDRESS FOR A COMMAND AND DATA FIELD FOR A      *
* PARAMETER.  modify io_port COMMAND# to PARAMETER                  *
*****
```

```
SETUP_SWBK      EQU 0      ; io_port COMMAND 0
RESTORE_OLD_VECTOR EQU 1    ; io_port COMMAND 1
                GLB        SETUP_SWBK, RESTORE_OLD_VECTOR
```

* COMMAND DECODE

```
    CMP.W      #SETUP_SWBK,A0
    BEQ        SET_SWBK
    CMP.W      #RESTORE_OLD_VECTOR,A0
    BEQ        RESTORE_VECTOR
```

* NONE OF THE ABOVE

```
    JMP        LOOP_REENTRY      ;GO BACK AND WAIT FOR
                                ;NEXT COMMAND
```

```
OLD_VALUE      DC.L        0
```

```
*****
* THIS COMMAND WILL SET UP THE SOFTWARE BREAK VECTOR TO POINT TO *
* THE SOFTWARE BREAK ENTRY INTO THE MONITOR. THE PARAMETER IS THE *
* TRAP # THAT IS GOING TO BE USED FOR THE SOFTWARE BREAK.        *
*****
```

SET_SWBK

```
    MOVE.L     #SETUP_MESS,MONITOR_MESSAGE
    CLR.L      D0
    MOVE.B     MONITOR_CONTROL+7,D0
    ASL.L     #2,D0
    ADD.L     #80H,D0
    MOVE.L     D0,A0
    MOVE.L     [A0],OLD_VALUE
    MOVE.L     #SWBK_ENTRY,[A0]
    JMP        LOOP_REENTRY      ;GO BACK AND WAIT
                                ;FOR NEXT COMMAND
```

SETUP_MESS

```
    DC.B      28
    ASC       "Software break vector set up"
    EVEN
```

```
*****
* THIS COMMAND WILL RESTORE THE SOFTWARE BREAK VECTOR TO THE VALUE *
* THAT WAS IN IT BEFORE THE SETUP_SWBK COMMAND WAS USED. THE      *
* PARAMETER IS THE TRAP # THAT IS BEING USED FOR THE SOFTWARE BREAK.*
*****
```

RESTORE_VECTOR

```
    MOVE.L     #RESTORE_MESS,MONITOR_MESSAGE
    CLR.L      D0
    MOVE.B     MONITOR_CONTROL+7,D0
    ASL.L     #2,D0
    ADD.L     #80H,D0
```

Emulator/Analyzer 68000/68008
Emulation Monitor Program

```
MOVE.L    D0,A0
MOVE.L    OLD_VALUE,[A0]
JMP      LOOP_REENTRY ;GO BACK AND WAIT FOR
                    ;NEXT COMMAND

RESTORE_MESS
DC.B      19
ASC      "Old vector restored"
EVEN
```

Figure 8-1. Sample Modified IO_FUNCTION

Emulation Monitor Memory Requirements (68000).

NOTE

If you are using the 68008 skip to the section entitled Emulation Monitor Memory Requirements For The 68008.

All of the code for the test program used in this demonstration will be loaded into memory space in the emulation hardware. The memory available in the emulation hardware is divided into 4K blocks of address space by the emulation system. Each 4K block begins on an even address multiple of 1000H.

The relocatable program area of the emulation monitor requires one 4K byte block of memory. This is because if you look at the :reloc file (figure 8-2) you will see under Prog length = 0764H that the program will take up 764 Hexadecimal locations. Because the value 764H is much less than 4000H, the entire 764H locations, can be mapped into one 4K byte block.

These memory requirements assume that the blocks each start on a 4K byte boundary and that the standard emulation monitor is being loaded. To check the memory requirements for the emulation monitor being used, record #1 of the emulation monitor relocatable file should be checked. An example of record #1 of the Mon_68000 relocatable file is shown in figure 8-2A.

Page # 1

```
File = Mon_68000:YOURID:reloc  Fri, 29 Mar 1985,  9:41

Record #  1    size =  94
Name record:
Source = Mon_68000:YOURID
Prog length = 0764H  Data length = 0000H  Comn length = 0000H
Number of externals is  0
Linker is l68000:HP
Wed, 27 Mar 1985,  8:10
Checksum = 1EB3H
```

Figure 8-2A. Mon_68000 Relocatable File - Record #1

Emulation Monitor Memory Requirements (68008)

NOTE

Use this section for configuring the 68008 only. Refer to the section entitled Emulation Memory Requirements (68000) when using the 68000.

The relocatable program area of the emulation monitor requires six 256 byte blocks of memory.

These memory requirements assume that the areas each start on a 256 byte boundary and that the standard emulation monitor is being loaded. To check the memory requirements for the emulation monitor being used, record #1 of the emulation monitor relocatable file should be checked. Look at the Prog length entry. As you can see from the listing, the program occupies 578H locations. If you convert this number to decimal, you obtain a value of 1400. Then you must divide this number by the block size of 256. This operation then gives the value of 5.5. Therefore, six blocks are required. You, of course, could convert the 256 block size to hexadecimal and divide, thereby saving a step. The results are the same either way.

An example of record #1 of the Mon_68008 relocatable file is shown in figure 8-2B.

Page # 1

```
File = RELOC:TEMP:reloc      Fri, 17 May 1985,  8:20

Record #  1    size =  90
Name record:
Source = Mon_68008:TEMP
Prog length = 0578H   Data length = 0000H   Comn length = 0000H
Number of externals is  0
Linker is l68008:HP
Fri, 17 May 1985,  7:43
Checksum = 9385H
```

Figure 8-2B. Mon_68008 Relocatable File - Record #1

Use This Information For Both Processors

The monitor program must reside in emulation memory and be mapped as RAM. This is necessary because the emulation monitor performs both reads from and writes to this area. It is recommended that in order to protect the monitor from being written over by a lost target system program, the first block above and below the data area should be mapped as guarded memory. Doing this will cause a break to be generated before the emulation monitor area has been accessed. Refer to the section in Chapter 6 "Filling in the Emulation Map" for more information on mapping these areas as guarded.

Linking The Emulation Monitor

The emulation monitor must be assembled and linked before it can be used by the emulation system. It can be linked with the target system code to produce one absolute file or it can be linked by itself. The disadvantage of linking the target system code and the emulation monitor at the same time is encountered during emulation. Whenever an absolute file that contains the

emulation monitor is loaded, the emulator will automatically be reset. Under some circumstances this may not be desirable. If the emulation monitor is linked separate from the target system code and the target system code references the symbol MONITOR_ENTRY to access the monitor, then the :link_sym file from the emulation monitor link should be referenced when linking the target system code.

At link time both a program and a data address must be specified. To easily distinguish the target system code from the emulation monitor when displaying the analysis data it is recommended that the emulation monitor be linked well outside the target system code space.

EMULATION MONITOR FLOWCHART

The emulation monitor flowchart is given in figure 8-3.

EMULATION MONITOR SOURCE PROGRAM

The emulation monitor program assembler code listing for the 680XX processor is given at the end of this chapter following the emulation monitor flowchart diagram.

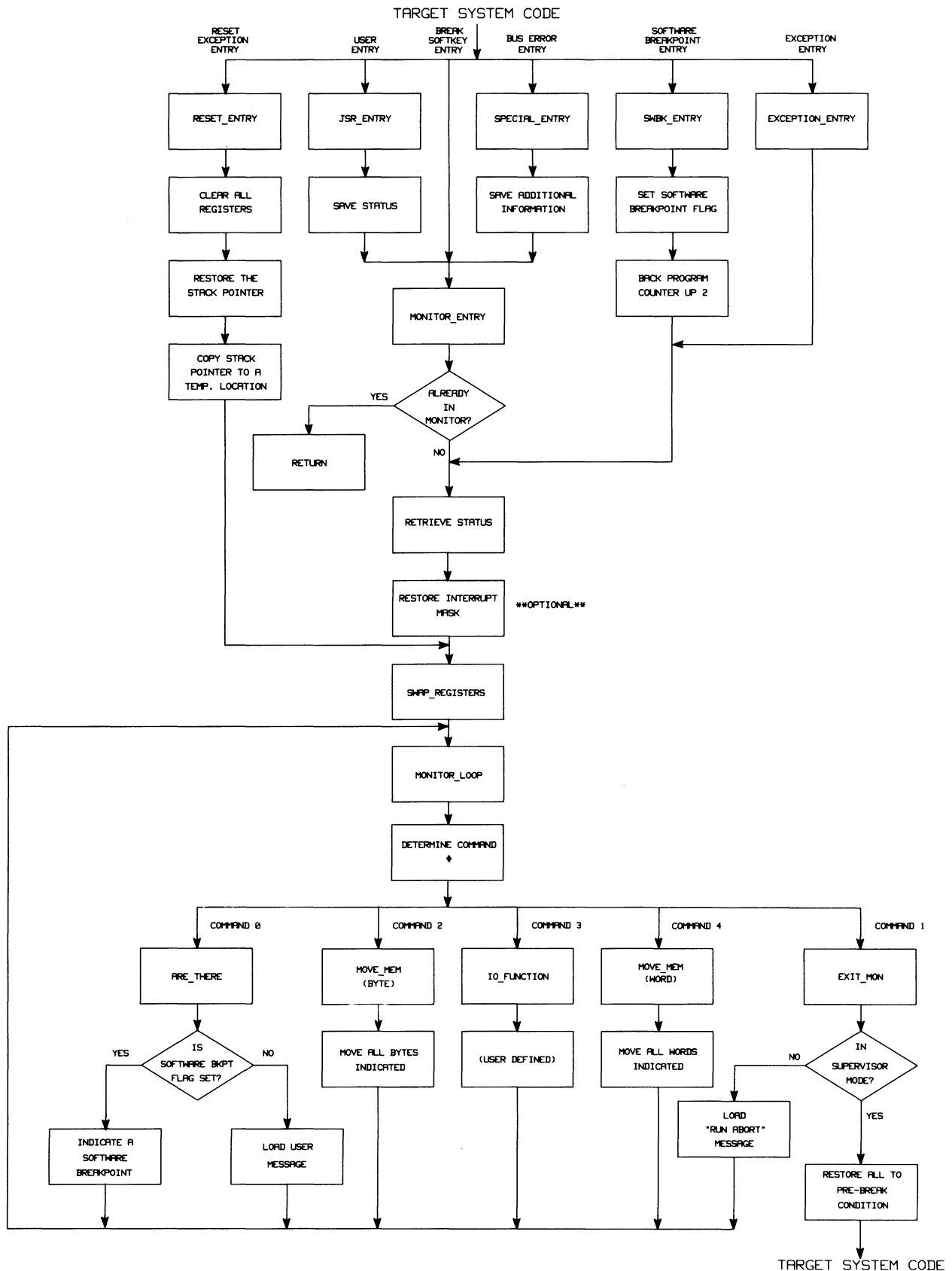


Figure 8-3. Emulation Monitor Flowchart

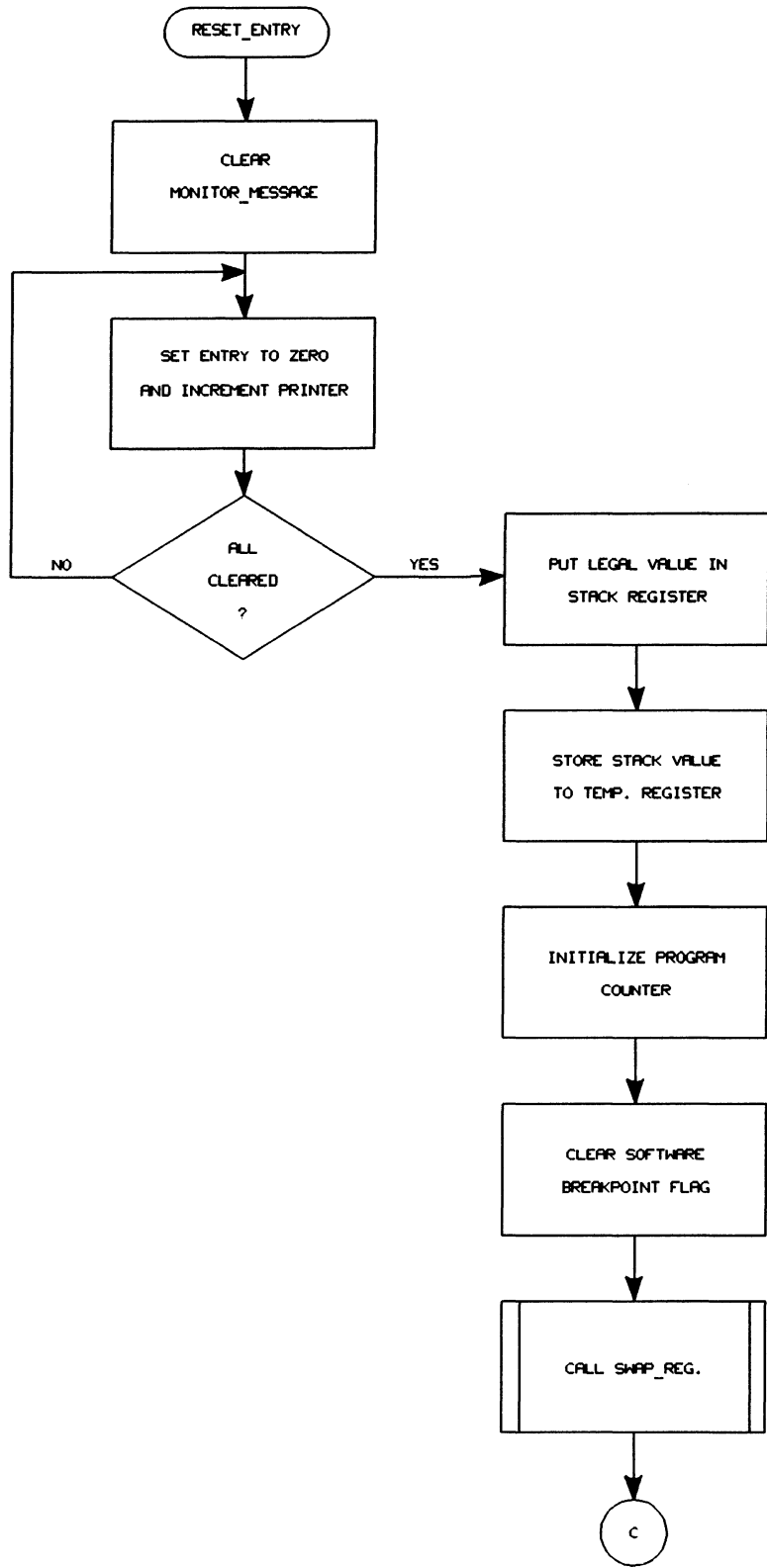


Figure 8-3. Emulation Monitor Flowchart (Cont'd)

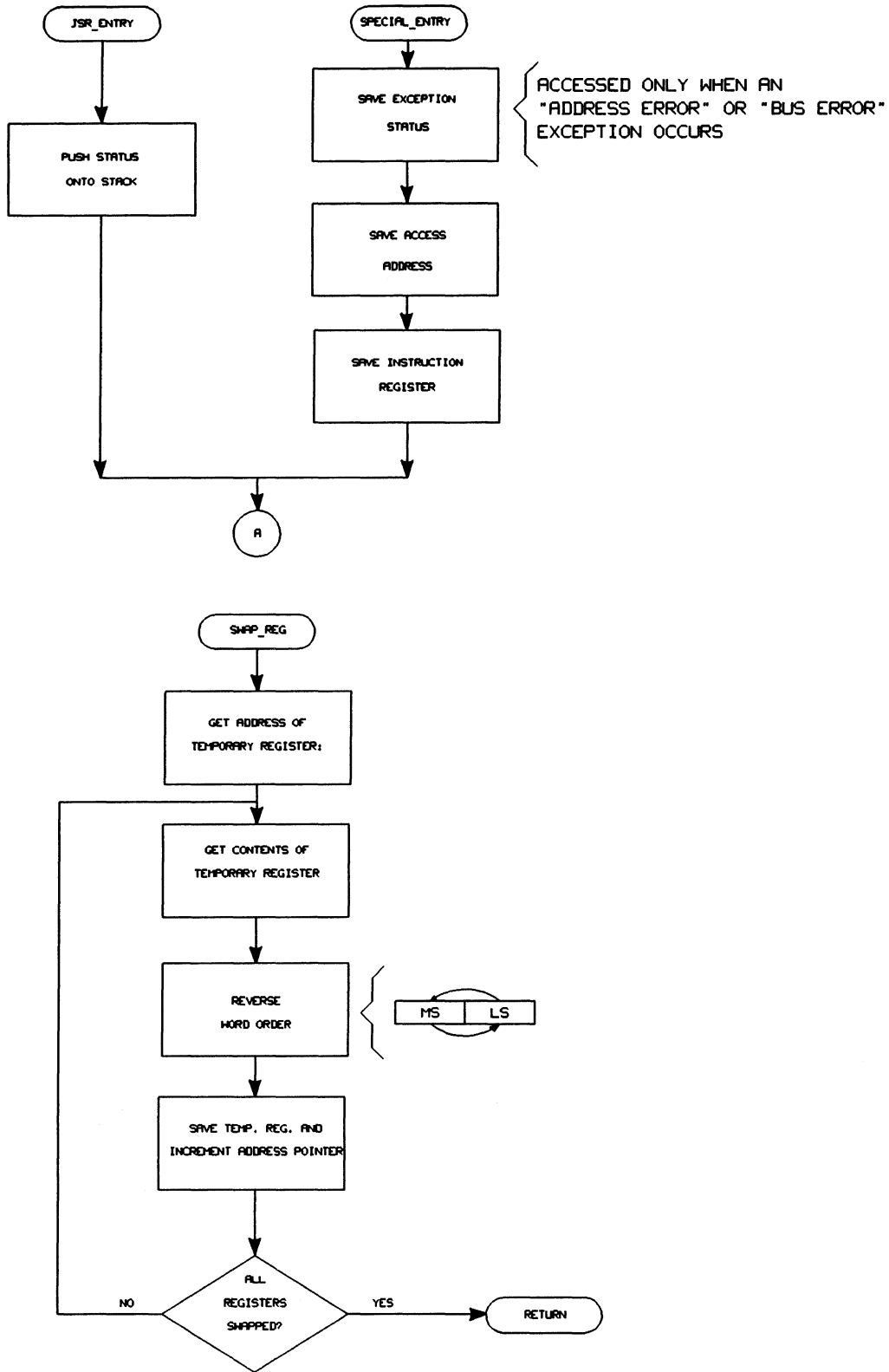


Figure 8-3. Emulation Monitor Flowchart (Cont'd)

Emulator/Analyzer 68000/68008
Emulation Monitor Flowchart

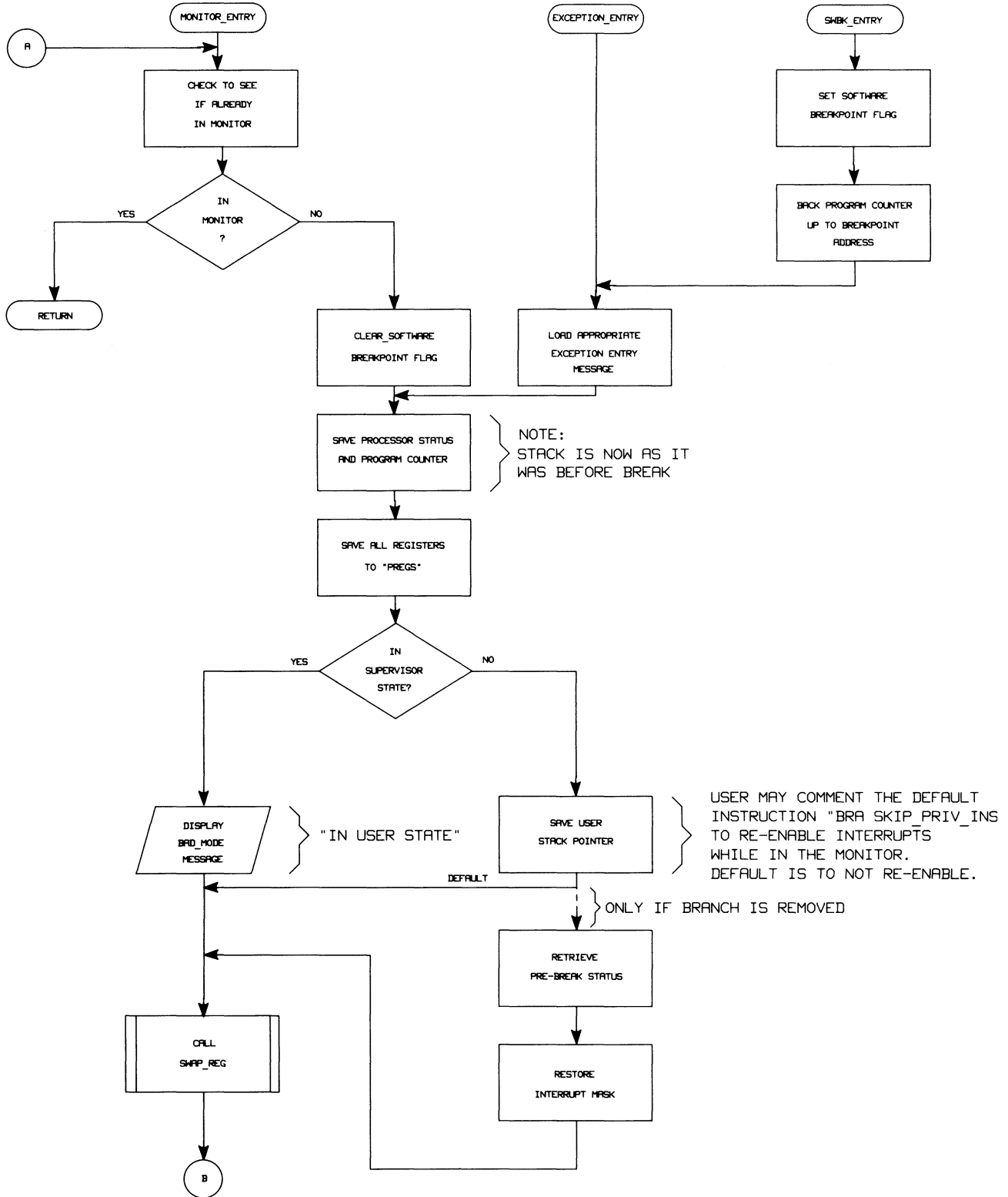


Figure 8-3. Emulation Monitor Flowchart (Cont'd)

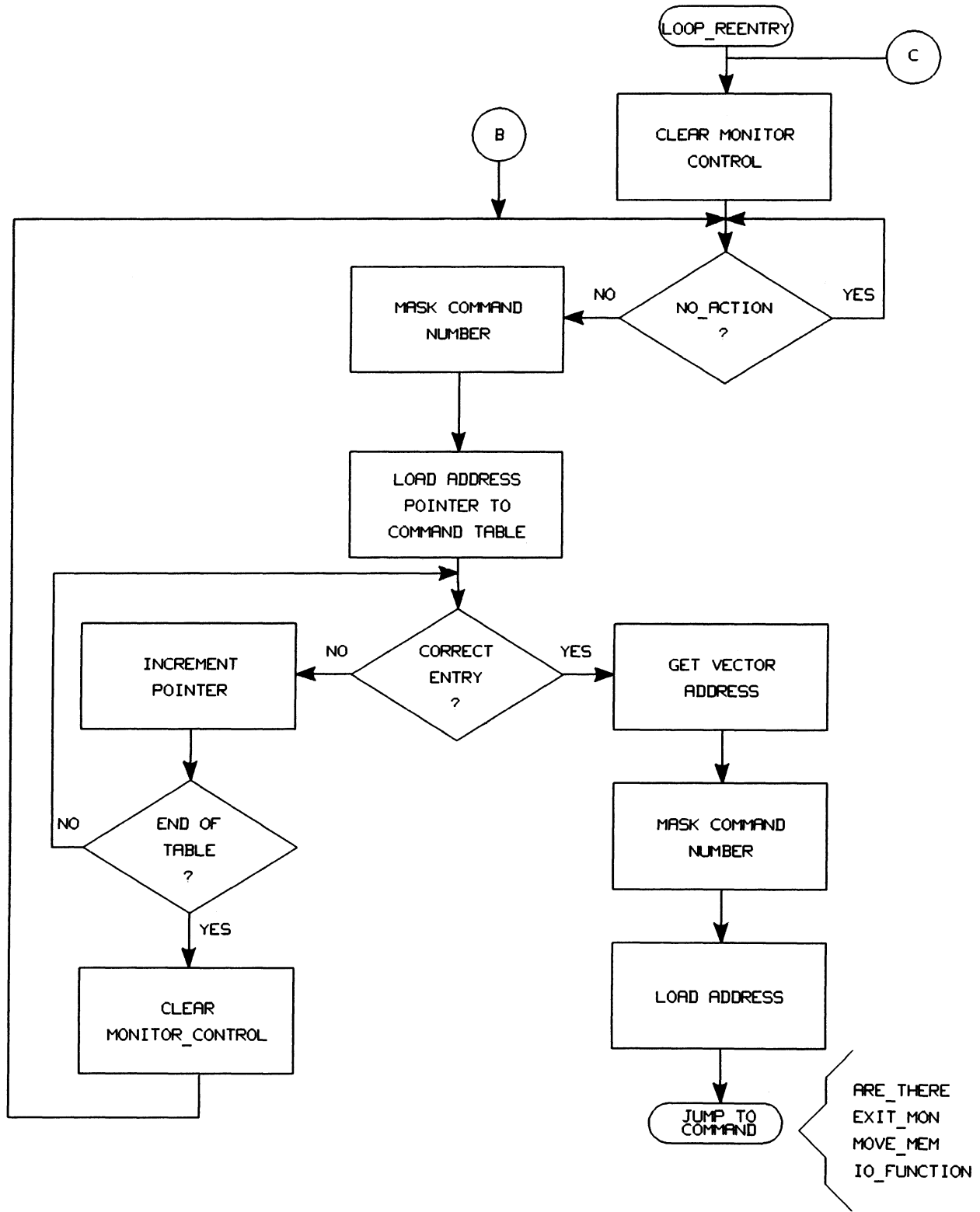


Figure 8-3. Emulation Monitor Flowchart (Cont'd)

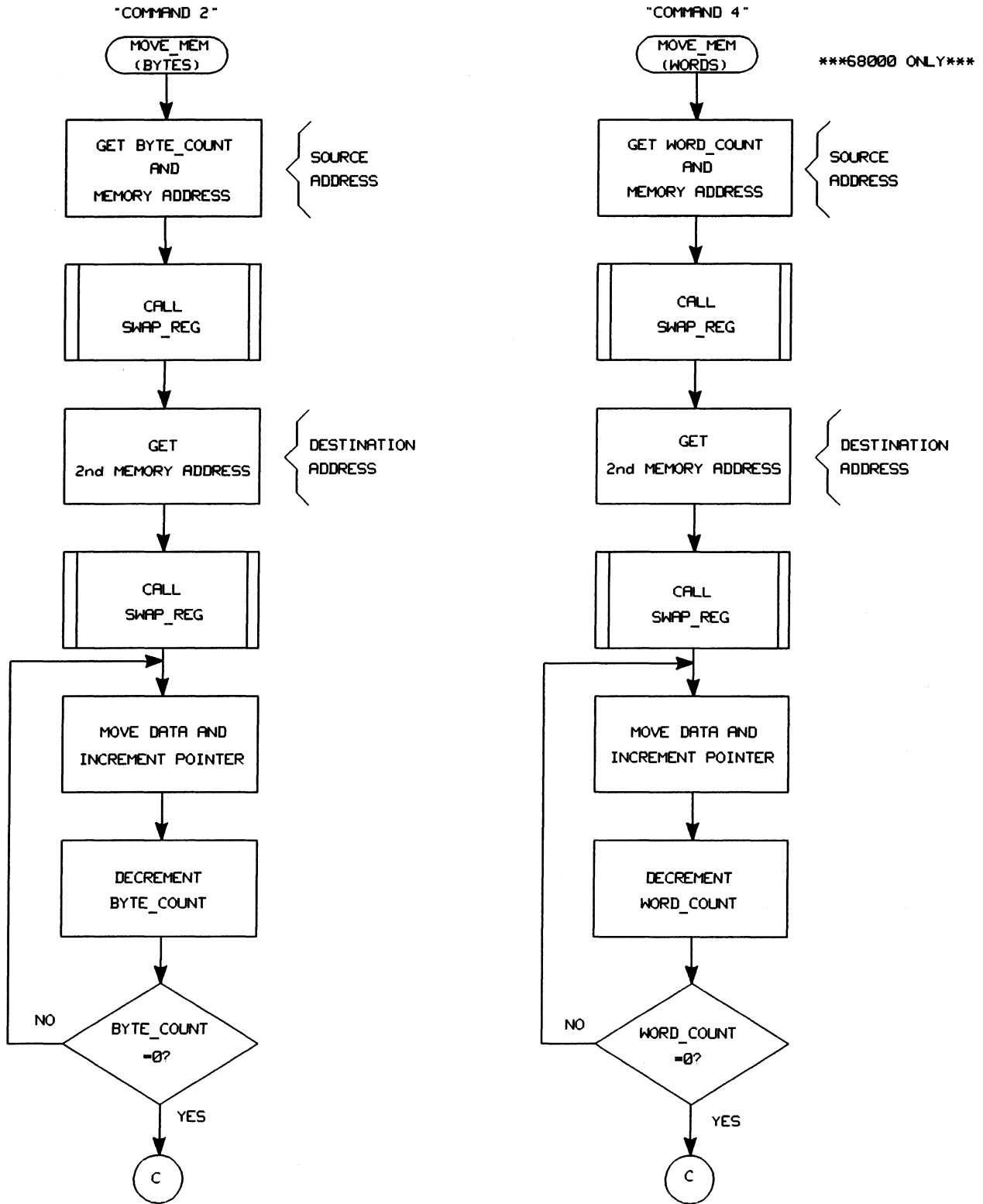


Figure 8-3. Emulation Monitor Flowchart (Cont'd)

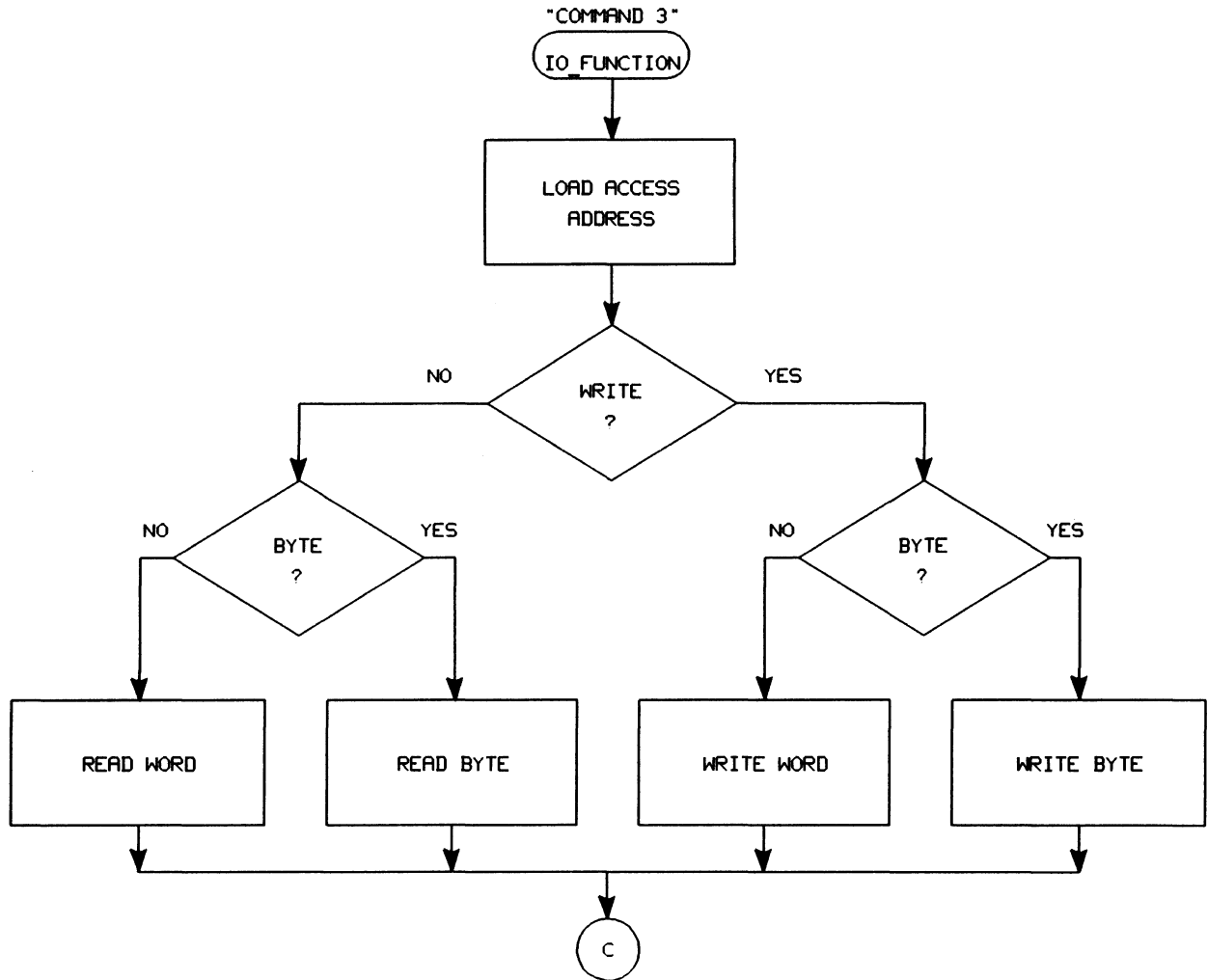


Figure 8-3. Emulation Monitor Flowchart (Cont'd)

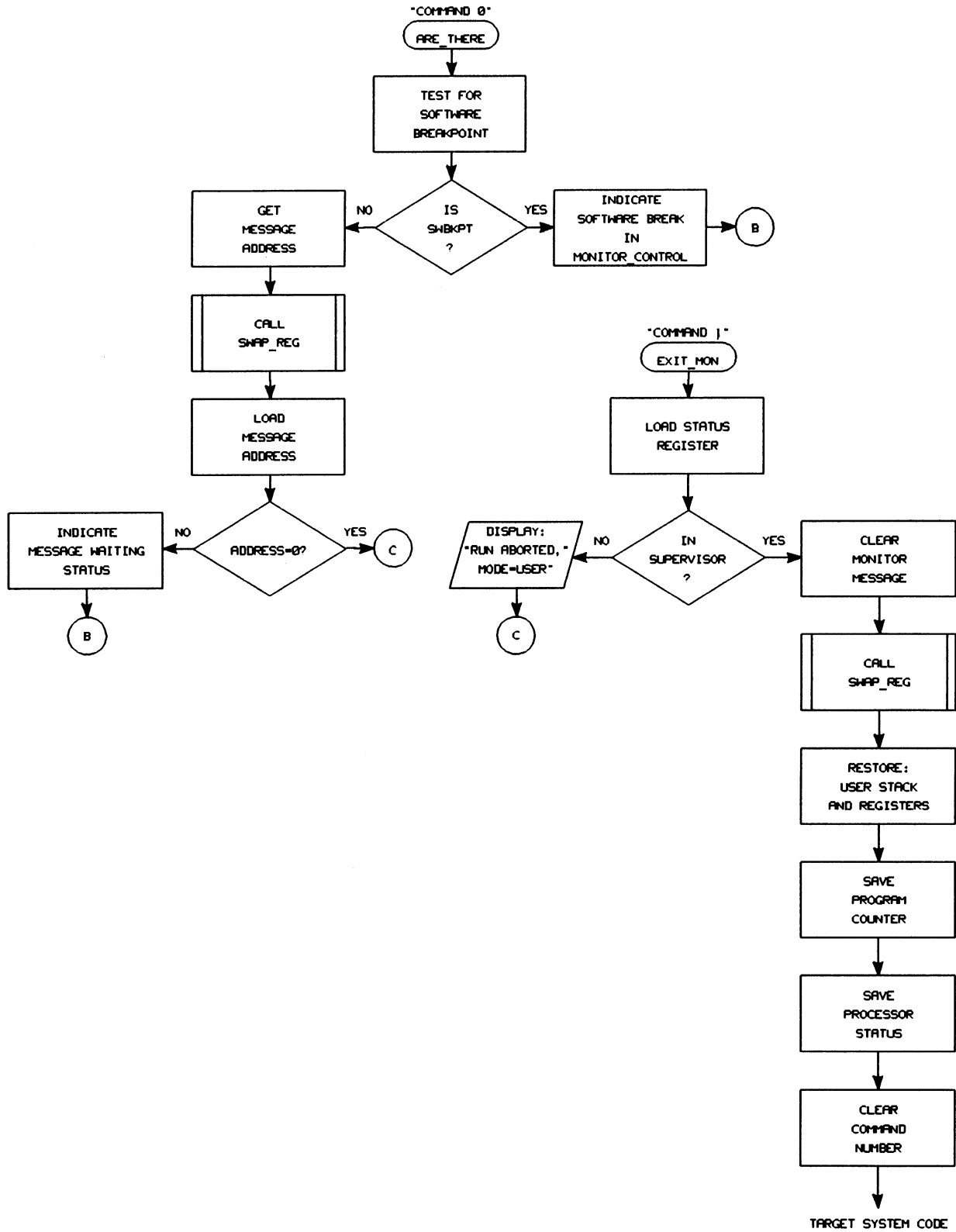


Figure 8-3. Emulation Monitor Flowchart (Cont'd)

```
1 "680XX"
2 *****
3 *
4 *   EMULATION MONITOR FOR 6424XA EMULATOR
5 *
6 *****
7 *   THE EMULATION MONITOR IS THE VEHICLE BY WHICH THE FOLLOWING
8 *   EMULATOR FUNCTIONS ARE EFFECTED:
9 *       READ/WRITE USER MEMORY SPACE
10 *      DISPLAY/MODIFY 68000 REGISTERS
11 *      DISPLAY/MODIFY I/O (USER DEFINED FUNCTION)
12 *      EXECUTE USER PROGRAM
13 *      BREAK AWAY FROM USER PROGRAM
14 *
15 *   TYPICALLY THE EMULATION MONITOR IS ASSEMBLED AND THEN
16 *   LINKED WITH THE "USER'S PROGRAM" TO FORM ONE LOAD MODULE.
17 *   THE EMULATION SESSION BEGINS BY DOWNLOADING THIS MODULE
18 *   INTO MEMORY.  THEN, DEPENDING ON THE DESIRES OF THE USER,
19 *   THE "run" COMMAND CAN BE ENTERED TO RELEASE THE PROCESSOR
20 *   FROM THE "reset" STATE.  IF THE RESET VECTOR HAS BEEN SET
21 *   TO POINT TO THE EMULATION MONITOR'S RESET_ENTRY THE 68000
22 *   WILL BEGIN EXECUTING THE EMULATION MONITOR AND ALL FEATURES
23 *   OF THE EMULATOR WILL BE AVAILABLE.  IF THE RESET VECTOR
24 *   POINTS TO USER CODE THEN THE PROCESSOR CAN BE DIRECTED INTO
25 *   THE EMULATION MONITOR BY ENTERING "break" AFTER "run".
26 *
27 *   IT IS INTENDED THAT THE USER MAY WISH TO CUSTOMIZE THE
28 *   EMULATION MONITOR TO ACCOMODATE SPECIFIC REQUIREMENTS
29 *   OF THE USER'S SYSTEM, FOR EXAMPLE READING/WRITING USER
30 *   MEMORY WHEN A MEMORY MANAGEMENT UNIT IS IN PLACE.
31 *
32 *****
33 *   EMULATION MONITOR GLOBALS
34 *
35 *   THE FOLLOWING SYMBOLS ARE SEARCHED FOR BY THE HOST
36 *   SYSTEM TO DEFINE THE PRESENCE OF THE EMULATION MONITOR
37 *   AND THE LOCATION OF THE MONITOR'S PARAMETERS.
38 *   A DESCRIPTION OF EACH SYMBOL IS MADE WHERE THE SYMBOL
39 *   IS DEFINED.
40 *****
41   GLB MONITOR_CONTROL,MONITOR_REGISTERS,XFR_BUF,MONITOR__6424XS
42 *****
43 *
44 *   EMULATION MONITOR GLOBALS
45 *       (PROVIDED FOR THE CONVENIENCE OF THE USER)
46 *
47 *****
48   GLB MONITOR_MESSAGE
49   GLB MONITOR_ENTRY,RESET_ENTRY,JSR_ENTRY,SPECIAL_ENTRY
50   SKIP
```

Emulator/Analyzer
Monitor Program Listing - 68000/68008

```
51 *****
52 *   THE FOLLOWING TABLE DEFINES THE FIRST 11 EXCEPTION VECTORS
53 *   FOR THE CONVENIENCE OF THE USER.  IF THE USER WISHES TO
54 *   DEFINE ANY OR ALL OF THESE VECTORS, THEY MAY BE CHANGED
55 *   TO OR FROM COMMENTS.
56 *****
57     ORG 0   ---RESET---
58     DC.L SP_TEMP
59     DC.L RESET_ENTRY
60 * ORG 8     ---BUS ERROR---
61 *   DC.L BE_ENTRY
62 * ORG 0CH   ---ADDRESS ERROR---
63 *   DC.L AE_ENTRY
64 * ORG 10H   ---ILLEGAL INSTRUCTION---
65 *   DC.L II_ENTRY
66 * ORG 14H   ---ZERO DIVIDE---
67 *   DC.L ZD_ENTRY
68 * ORG 18H   ---CHK INSTRUCTION---
69 *   DC.L CI_ENTRY
70 * ORG 1CH   ---TRAPV INSTRUCTION---
71 *   DC.L TI_ENTRY
72 * ORG 20H   ---PRIVILEGE VIOLATION---
73 *   DC.L PV_ENTRY
74 * ORG 24H   ---TRACE---
75 *   DC.L T_ENTRY
76 * ORG 28H   ---1010 EMULATOR---
77 *   DC.L EA_ENTRY
78 * ORG 2CH   ---1111 EMULATOR---
79 *   DC.L FE_ENTRY
80
81 *****
82 *   TRAP VECTORS ( SOFTWARE BREAKPOINT VECTORS ) CHOOSE ONE
83 *   -----
84 *   The TRAP vector you should choose for the software breakpoint
85 *   depends on the trap # chosen in the configuration question.
86 *****
87 * ORG 080H
88 *   DC.L SWBK_ENTRY      ;TRAP #0
89 * ORG 084H
90 *   DC.L SWBK_ENTRY      ;TRAP #1
91 * ORG 088H
92 *   DC.L SWBK_ENTRY      ;TRAP #2
93 * ORG 08CH
94 *   DC.L SWBK_ENTRY      ;TRAP #3
95 * ORG 090H
96 *   DC.L SWBK_ENTRY      ;TRAP #4
97 * ORG 094H
98 *   DC.L SWBK_ENTRY      ;TRAP #5
99 * ORG 098H
100 *   DC.L SWBK_ENTRY      ;TRAP #6
101 * ORG 09CH
102 *   DC.L SWBK_ENTRY      ;TRAP #7
103 * ORG 0A0H
104 *   DC.L SWBK_ENTRY      ;TRAP #8
```

```

105 * ORG 0A4H
106 *   DC.L SWBK_ENTRY      ;TRAP #9
107 * ORG 0A8H
108 *   DC.L SWBK_ENTRY      ;TRAP #10
109 * ORG 0ACH
110 *   DC.L SWBK_ENTRY      ;TRAP #11
111 * ORG 0B0H
112 *   DC.L SWBK_ENTRY      ;TRAP #12
113 * ORG 0B4H
114 *   DC.L SWBK_ENTRY      ;TRAP #13
115 * ORG 0B8H
116 *   DC.L SWBK_ENTRY      ;TRAP #14
117 * ORG 0BCH
118 *   DC.L SWBK_ENTRY      ;TRAP #15
119
120 *****
121 *   BREAK VECTOR ( RE-ORG WHERE APPROPRIATE )
122 *       NOT NEEDED FOR THE 6424X
123 *****
124 *   ORG 7CH
125 *   DC.L MONITOR_ENTRY
126 *   SKIP
127 *****
128 *   THE FOLLOWING SEGMENTS ARE RELOCATABLE.
129 *   PROG
130 * MONITOR_6424XS is used to inform the host which monitor is being
131 * used.
132 *
133 MONITOR_6424XS
134 MONITOR_START
135 *****
136 *
137 *   MONITOR_ENTRY IS THE ENTRY POINT INTO THE EMULATION
138 *   MONITOR WHEN THE 68000 BREAKS AWAY FROM THE USER'S
139 *   PROGRAM.  THIS ENTRY POINT ASSUMES THAT THE PC AND
140 *   STATUS REGISTER HAVE BEEN STACKED AS NORMALLY OCCURS
141 *   WHEN AN EXCEPTION HAPPENS.
142 *
143 *   THE PROCESSOR'S REGISTERS ARE SAVED AND THEIR ORDER:
144 *   HIGH WORD/LOW WORD IS SWAPPED TO ACCOMODATE THE CONVENTION
145 *   USED BY THE HOST.
146 *****
147 MONITOR_ENTRY
148 *                               ;LOOK FOR BREAK WHILE ALREADY...
149 *                               ;...IN-MONITOR.
150 *   CMP.L   #MONITOR_START,2[A7]; COMPARE ADDRESS ON STACK WITH...
151 *   BMI     BREAK_OK           ; ...START OF EMULATION MONITOR.
152 *   CMP.L   #MONITOR_END,2[A7]; COMPARE ADDRESS ON STACK WITH...
153 *   BPL     BREAK_OK           ; ...END OF EMULATION MONITOR...
154 *   RTE                               ; ...PROGRAM SPACE.
155 BREAK_OK
156 *   BCLR    #0,SWBKPT_FLAG      ;CLEAR SOFTWARE BREAKPT FLAG.
157 EXCEPTION_ENTRY
158 *   MOVE    [SP]+,PSTATUS       ;PULL & SAVE PROC STATUS.

```

Emulator/Analyzer
 Monitor Program Listing - 68000/68008

```

159     MOVE     [SP]+,PCH           ;PULL & SAVE PC HIGH WORD.
160     MOVE     [SP]+,PCL           ;PULL & SAVE PC LOW WORD.
161 *    STACK IS NOW AS IT WAS BEFORE BREAK.
162     AND      #0FEFFH,SR         ;SET LEVEL 6 INT.
163     JUMP_ENTRY           ;SAVE PC SETUP IN RESET_ENTRY.
164     MOVEM.L   A0-A7/D0-D7,PREGS ;SAVE REG FILE (EXCLUDING USP).
165     MOVE     SR,D0              ;CHECK TO BE SURE THAT 68000....
166     BTST     #13,D0             ;...IS IN SUPERVISOR MODE.
167     BNE      MODE_OK
168     MOVE.L   #BAD_MODE_MESS,MONITOR_MESSAGE ;IF MODE IS USER THEN...
169     BRA      SKIP_PRIV_INSTR    ;...SETUP MESSAGE AND SKIP THE...
170 *                                     ;...PRIVILEGED INSTRUCTION.
171     BAD_MODE_MESS DC.B 22
172     ASC      "Error! Entry Mode=User"
173     DC.B 0
174     MODE_OK
175     MOVE.L   USP,A0
176     MOVE.L   A0,USPT            ;NOW SAVE USP.
177 *****
178 *
179 *    IN ORDER TO KEEP USER INTERRUPTS ENABLED
180 *    THE FOLLOWING SEGMENT WILL RESTORE THE INTERRUPT MASK
181 *    TO ITS PRE-BREAK STATE.
182 *
183 *    USER INTERRUPT ROUTINES ARE EXPECTED TO PRESERVE ALL
184 *    REGISTERS IF INTERRUPTS ARE TO BE ENABLED WHILE
185 *    IN THE EMULATION MONITOR.
186 *
187 *****
188     BRA      SKIP_PRIV_INSTR    ;DEFAULT IS NOT RE-ENABLE INTERRUPTS
189     MOVE     PSTATUS,D0         ;GET COPY OF PRE-BREAK STATUS REG.
190     OR       #0FBFFH,D0        ;COVER ALL BITS EXCEPT
191 *                                     ;..INTERRUPT MASK.
192     MOVE     SR,D1
193     AND      D1,D0
194     MOVE     D0,SR              ;RESTORE INTERRUPT MASK.
195 *****
196     SKIP_PRIV_INSTR
197     LEA     RTN1,A6             ;LOAD RETURN ADDRESS.
198 *    "JSR" NOT USED TO AVOID STACK PROBLEMS.
199     JMP     SWAP_REG            ;REARRANGE REGS TO HOST FORMAT.
200     RTN1
201     BRA     MONITOR_LOOP        ;SKIP TO COMMAND LOOP.
202 *
203 *
204 *****
205 *
206 *    SPECIAL_ENTRY IS THE ENTRY POINT INTO THE EMULATION
207 *    MONITOR WHEN THE 68000 PROCESSES EITHER A BUS ERROR
208 *    OR ADDRESS ERROR EXCEPTION. THE ONLY DIFFERENCE BETWEEN
209 *    THIS ENTRY AND MONITOR_ENTRY IS THAT ADDITIONAL WORDS
210 *    ARE TAKEN OFF THE STACK.
211 *
212 *****

```

```

213 SPECIAL_ENTRY
214     MOVE     [SP]+,STAT1      ;PULL & SAVE EXCEPTION STATUS.
215     MOVE     [SP]+,STAT2      ;PULL & SAVE ACCESS ADDRESS HIGH.
216     MOVE     [SP]+,STAT3      ;PULL & SAVE ACCESS ADDRESS LOW.
217     MOVE     [SP]+,STAT4      ;PULL & SAVE INSTRUCTION REGISTER.
218     BRA      MONITOR_ENTRY
219     SKIP
220 *****
221 *
222 *     SWBK_ENTRY IS THE ENTRY POINT INTO THE EMULATION
223 *     MONITOR WHEN THE 68000 PROCESSES A SOFTWARE BREAKPOINT I.E.
224 *     A "TRAP #" PLACED IN MEMORY BY THE 64000 SYSTEM.
225 *
226 *****
227 SWBK_ENTRY
228     BSET     #0,SWBKPT_FLAG    ;SET FLAG TO INDICATE SOFTWARE BKPT ENTRY.
229     SUB.L    #2,2[A7]          ;BACKUP PC TO POINT TO BREAKPOINT ADDRESS.
230     BRA      EXCEPTION_ENTRY
231 *****
232 *
233 *     THE FOLLOWING ENTRY POINTS PROVIDE STATUS MESSAGES FOR THE
234 *     10 EXCEPTION VECTORS AFTER RESET.  THESE ARE PROVIDED STRICTLY
235 *     FOR THE CONVENIENCE OF THE USER AND MAY BE DELETED OR CHANGED.
236 *
237 *****
238 *           ---BUS ERROR---
239 BE_ENTRY
240     MOVE.L   #BE_MESS,MONITOR_MESSAGE
241     BRA      SPECIAL_ENTRY
242 BE_MESS
243     DC.B    15
244     ASC     "---Bus Error---"
245 *           ---ADDRESS ERROR---
246 AE_ENTRY
247     MOVE.L   #AE_MESS,MONITOR_MESSAGE
248     BRA      SPECIAL_ENTRY
249 AE_MESS
250     DC.B    19
251     ASC     "---Address Error---"
252 *           ---ILLEGAL INSTRUCTION---
253 II_ENTRY
254     MOVE.L   #II_MESS,MONITOR_MESSAGE
255     BRA      MONITOR_ENTRY
256 II_MESS
257     DC.B    25
258     ASC     "---Illegal Instruction---"
259 *           ---ZERO DIVIDE---
260 ZD_ENTRY
261     MOVE.L   #ZD_MESS,MONITOR_MESSAGE
262     BRA      MONITOR_ENTRY
263 ZD_MESS
264     DC.B    17
265     ASC     "---Zero Divide---"
266     SKIP

```

Emulator/Analyzer
 Monitor Program Listing - 68000/68008

```

267 *          ---CHK INSTRUCTION---
268 CI_ENTRY
269     MOVE.L #CI_MESS,MONITOR_MESSAGE
270     BRA  MONITOR_ENTRY
271 CI_MESS
272     DC.B 21
273     ASC "----CHK Instruction---"
274 *          ---TRAPV INSTRUCTION---
275 TI_ENTRY
276     MOVE.L #TI_MESS,MONITOR_MESSAGE
277     BRA  MONITOR_ENTRY
278 TI_MESS
279     DC.B 23
280     ASC "----TRAPV Instruction---"
281 *          ---PRIVILEGE VIOLATION---
282 PV_ENTRY
283     MOVE.L #PV_MESS,MONITOR_MESSAGE
284     BRA  MONITOR_ENTRY
285 PV_MESS
286     DC.B 27
287     ASC "----Privilege Violation--- "
288 *          ---TRACE---
289 T_ENTRY
290     MOVE.L #T_MESS,MONITOR_MESSAGE
291     BRA  MONITOR_ENTRY
292 T_MESS
293     DC.B 11
294     ASC "----Trace---"
295     SKIP
296 *          ---1010 EMULATOR---
297 EA_ENTRY
298     MOVE.L #EA_MESS,MONITOR_MESSAGE
299     BRA  MONITOR_ENTRY
300 EA_MESS
301     DC.B 20
302     ASC "----1010 Exception---"
303     DC.B 0
304 *          ---1111 EMULATOR---
305 FE_ENTRY
306     MOVE.L #FE_MESS,MONITOR_MESSAGE
307     BRA  MONITOR_ENTRY
308 FE_MESS
309     DC.B 20
310     ASC "----1111 Exception---"
311     DC.B 0
312     SKIP
313 *****
314 *
315 *   JSR_ENTRY IS THE ENTRY POINT INTO THE EMULATION
316 *   MONITOR FOR USERS WHO WISH TO JUMP DIRECTLY INTO THE
317 *   MONITOR. THE USER CAN "JSR JSR_ENTRY" WHEN SURE OF BEING
318 *   IN THE SUPERVISOR MODE, OTHERWISE A TRAP EXCEPTION SHOULD
319 *   BE USED. THE TRAP VECTOR SHOULD POINT TO MONITOR_ENTRY.
320 *

```



```

321 *****
322 JSR_ENTRY
323     MOVE     SR,-[A7]           ;PUSH STATUS ONTO THE STACK
324     BRA      MONITOR_ENTRY     ;TAKE NORMAL BREAK ENTRY.
325 *****
326 *
327 *     RESET_ENTRY IS THE ENTRY POINT INTO THE EMULATION
328 *     MONITOR WHEN THE 68000 PROCESSES THE RESET EXCEPTION.
329 *     A DEFAULT STACK IS SET UP AND DEFAULT VALUES ARE GIVEN
330 *     TO THE PROCESSOR'S REGISTERS.
331 *
332 *****
333 RESET_ENTRY
334 *           DEFAULT ALL REGISTER VALUES.
335     CLR.L    MONITOR_MESSAGE    ;IF NON-ZERO, THEN CLEAR ADDRESS.
336     LEA     MONITOR_REGISTERS+2,A0
337 REG_CLR
338     CLR     [A0]+              ;SET ENTRY TO ZERO.
339     CMP.L   #REG_SUB_END,A0    ;DONE?
340     BCS     REG_CLR            ;NO, CONTINUE CLEARING.
341     MOVE.L  A7,USP             ;GIVE LEGAL VALUE TO USP.
342     MOVE.L  A7,ADDR_7         ;COPY THESE VALUES TO THE...
343     MOVE.L  A7,USPT           ;REGISTER TEMPORARIES.
344     MOVE.L  #JUMP_ENTRY,D0    ;INITIALIZE PC TO SAFELY...
345     SWAP    D0                 ;RE-ENTER THE MONITOR.
346     MOVE.L  D0,PCL
347     MOVE    #2700H,PSTATUS
348     BCLR   #0,SWBKPT_FLAG     ;INITIALIZE SOFTWARE BREAKPT FLAG.
349     LEA    RTN2,A6            ;LOAD RETURN ADDRESS.
350 *     "JSR" NOT USED TO AVOID STACK PROBLEMS.
351     JMP     SWAP_REG          ;REARRANGE REGS TO HOST FORMAT.
352 RTN2
353     SKIP
354 *****
355 *
356 *           LOOP_REENTRY CLEARS "MONITOR_CONTROL" WHICH EITHER:
357 *           INITIALIZES A NO_ACTION COMMAND IF EXECUTED VIA
358 *           RESET_ENTRY, OR...
359 *           SIGNALS THE HOST THAT THE LAST MONITOR COMMAND HAS
360 *           BEEN COMPLETED.
361 *
362 *****
363 LOOP_REENTRY
364     CLR     MONITOR_CONTROL    ;SET NO_ACTION COMMAND.
365 *****
366 *
367 *     MONITOR_LOOP IS THE IDLE LOOP OF THE EMULATION MONITOR.
368 *     THE LOOP IS EXITED WHEN BIT 15 OF MONITOR_CONTROL
369 *     BECOMES A "1". THE OTHER PARAMETERS (BYTE_COUNT, SRC_ADDR, ETC.)
370 *     SHOULD BE SET BY THIS TIME.
371 *****
372 MONITOR_LOOP
373     MOVE    MONITOR_CONTROL,D0 ;GET COMMAND NUMBER.
374     BPL    MONITOR_LOOP       ;LOOP ON NO_ACTION.

```

Emulator/Analyzer
 Monitor Program Listing - 68000/68008

```

375 *****
376 *
377 *   THE LOWER BYTE OF MONITOR_CONTROL CONTAINS THE COMMAND
378 *   NUMBER.  THIS BYTE IS ISOLATED AND THE COMMAND TABLE
379 *   BEGINNING AT "CMD_STRT" AND ENDING AT "CMD_END" IS
380 *   SEARCHED FOR A MATCH.
381 *
382 *****
383   AND      #OFFH,D0          ;ISOLATE COMMAND NUMBER.
384   LEA      CMD_STRT,A0      ;LOAD PTR TO COMMAND TABLE.
385 CMD_SRCH
386   CMP.B    [A0],D0          ;IS CURRENT ENTRY THE DESIRED COMMAND?
387   BEQ      CMD_FOUND       ;YES, SKIP ON.
388   ADD.L    #4,A0            ;INC TO NEXT TABLE ENTRY.
389   CMP.L    #CMD_END,A0     ;AT END OF COMMAND TABLE?
390   BCS      CMD_SRCH        ;NO, KEEP LOOKING.
391 *   ILLEGAL COMMAND ATTEMPTED
392   BCLR     #7,MONITOR_CONTROL
393   BRA      MONITOR_LOOP    ;GO BACK AND WAIT FOR COMMAND.
394 *****
395 *   WHEN A VALID COMMAND HAS BEEN FOUND, THE ENTRY ADDRESS
396 *   FOR THE COMMAND CODE IS RETRIEVED FROM THE TABLE AND
397 *   THE ROUTINE IS EXECUTED.
398 *****
399 CMD_FOUND
400   MOVE.L   [A0],D1          ;GET VECTOR ADDRESS.
401   AND.L    #OFFFFFFH,D1    ;MASK OFF COMMAND NUMBER.
402   MOVE.L   D1,A0           ;MOVE TO A0 FOR INDIRECT JUMP.
403   JMP      [A0]            ;EXECUTE THE COMMAND.
404   SKIP
405 *****
406 *
407 *   SWAP_REG IS A SUBROUTINE WHICH REVERSES THE WORD ORDER
408 *   OF THE REGISTER TEMPORARIES A0-A7/D0-D7/USPT.
409 *   THIS IS NECESSARY BECAUSE
410 *   THE HOST SYSTEM EXPECTS DOUBLE WORD VARIABLES TO BE
411 *   ARRANGED LEAST SIGNIFICANT WORD FIRST, MSW LAST
412 *   WHILE THE 68000 EXPECTS THE OPPOSITE ORDER.
413 *   THE SUBROUTINE EXPECTS A6 TO CONTAIN THE RETURN ADDRESS.
414 *   A "JSR" CALL WAS NOT USED TO AVOID STACK PROBLEMS.
415 *****
416 SWAP_REG
417   LEA      PREGS,A0        ;LOAD ADDRESS OF D0 TEMPORARY.
418 SWAP_LOOP
419   MOVE.L   [A0],D0         ;GET TEMPORARY.
420   SWAP     D0              ;REVERSE THE WORD ORDER.
421   MOVE.L   D0,[A0]+        ;RE-WRITE THE TEMPORARY AND
422   CMP.L    #REG_END,A0     ;INCREMENT THE POINTER.
423   BCS      SWAP_LOOP      ;LOOP UNTIL ALL REGS SWAPPED.
424   JMP      [A6]
425   SKIP
426 *****
427 *
428 *   EMULATION MONITOR COMMANDS

```

```

429 *
430 *****
431 *
432 *****
433 *   COMMAND 0 ... ARE THERE?
434 *
435 *   "ARE THERE?" IS USED BY THE HOST TO DETERMINE WHETHER
436 *   THE PROCESSOR IS EXECUTING IN THE MONITOR OR THE
437 *   USER'S PROGRAM. IT ALSO CAN PASS THE ADDRESS OF A
438 *   ASCII MESSAGE TO BE DISPLAYED ON THE HOST SYSTEM'S
439 *   STATUS LINE.
440 ARE_THERE
441   BTST     #0,SWBKPT_FLAG      ;INITIALIZE SOFTWARE BREAKPT FLAG.
442   BEQ      NO_BKPT
443   MOVE     #2,MONITOR_CONTROL  ;INDICATE SOFTWARE BREAKPT ENTRY.
444   BRA      MONITOR_LOOP       ;GO TO IDLE LOOP.
445 NO_BKPT
446   MOVE.L   MONITOR_MESSAGE,D0  ;GET MESSAGE ADDRESS
447   SWAP     D0                  ;REVERSE WORD ORDER FOR HOST.
448   MOVE.L   D0,MESSAGE_ADDR     ;PLACE ADDRESS IN PARM AREA.
449   BEQ      LOOP_REENTRY       ;IF ZERO ADDR, THEN NULL MESSAGE.
450   MOVE     #1,MONITOR_CONTROL  ;INDICATE MESSAGE WAITING STATUS.
451   BRA      MONITOR_LOOP       ;GO TO IDLE LOOP.
452 *****
453 *   COMMAND 1 ... EXIT THE MONITOR
454 *
455 *   "EXIT THE MONITOR" RELOADS THE PROCESSOR'S REGISTER
456 *   FILE AND EXITS TO THE USER'S PROGRAM.
457 EXIT_MON
458   MOVE     SR,D0              ;CHECK TO BE SURE THAT 68000....
459   BTST     #13,D0             ;...IS IN SUPERVISOR MODE.
460   BNE      MODE_OK2
461   MOVE.L   #RUN_ABRT_MESS,MONITOR_MESSAGE
462   BRA      LOOP_REENTRY       ;GO TO IDLE LOOP.
463 RUN_ABRT_MESS DC.B 22
464   ASC      "Run Aborted! Mode=User"
465   DC.B 0
466 MODE_OK2
467   CLR.L    MONITOR_MESSAGE     ;CLEAR MESSAGE ADDRESS.
468   LEA     RTN3,A6             ;LOAD RETURN ADDRESS.
469 *   "JSR" NOT USED TO AVOID STACK PROBLEMS.
470   JMP      SWAP_REG           ;REARRANGE REGS TO HOST FORMAT.
471 RTN3
472   MOVE.L   USPT,A0            ;RESTORE USER STACK PTR.
473   MOVE.L   A0,USP             ;RESTORE USER STACK PTR.
474   MOVEM.L PREGS,A0-A7/D0-D7  ;RESTORE REGISTER FILE.
475   MOVE     PCL,-[SP]          ;PUSH PC LOW WORD.
476   MOVE     PCH,-[SP]          ;PUSH PC HIGH WORD.
477   MOVE     PSTATUS,-[SP]      ;PUSH PROC STATUS.
478   CLR      MONITOR_CONTROL    ;CLEAR COMMAND NUMBER.
479   RTE

```

Emulator/Analyzer
 Monitor Program Listing - 68000/68008

```

480 *****
481 *
482 *   COMMANDS 2 and 4.... ACCESS USER MEMORY
483 MOVE_MEM
484     MOVE      BYTE_COUNT,D3
485     MOVE.L    SRC_ADDR,D2      ; GET MEMORY ADDRESS.
486     SWAP     D2
487 * DATA REG IS USED SO SWAP CAN GET ADDRESS IN CORRECT ORDER.
488     MOVE.L    D2,A0            ;MOVE CORRECTED ADDRESS TO A0
489     MOVE.L    DST_ADDR,D2      ; GET MEMORY ADDRESS.
490     SWAP     D2
491 * DATA REG IS USED SO SWAP CAN GET ADDRESS IN CORRECT ORDER.
492     MOVE.L    D2,A1            ;MOVE CORRECTED ADDRESS TO A0
493 * MASK OFF UPPER BITS
494     MOVE.L    A0,D2
495     ANDI.L    #0FFFFFFH,D2
496     MOVE.L    D2,A0
497     MOVE.L    A1,D2
498     ANDI.L    #0FFFFFFH,D2
499     MOVE.L    D2,A1
500     CMP.B     #2,D0
501     BNE.B     MOVE_WORD
502 MOVE_BYTE
503     MOVE.B    [A0],[A1]+      ; DO MOVE BYTE OPERATION.
504     SUBQ     #1,D3
505     BNE      MOVE_BYTE
506     JMP      LOOP_REENTRY     ;GO BACK AND WAIT FOR COMMAND
507 MOVE_WORD
508     MOVE.W    [A0],[A1]+      ; DO MOVE WORD OPERATION.
509     SUBQ     #2,D3
510     BGT      MOVE_WORD
511     JMP      LOOP_REENTRY     ;GO BACK AND WAIT FOR COMMAND
512 *****
*****
*
*   COMMAND 2 ... ACCESS USER MEMORY
MOVE_MEM
    MOVE      BYTE_COUNT,D3
    MOVE.L    SRC_ADDR,D2      ; GET MEMORY ADDRESS.
    SWAP     D2
* DATA REG IS USED SO SWAP CAN GET ADDRESS IN CORRECT ORDER.
    MOVE.L    D2,A0            ;MOVE CORRECTED ADDRESS TO A0
    MOVE.L    DST_ADDR,D2      ; GET MEMORY ADDRESS.
    SWAP     D2
* DATA REG IS USED SO SWAP CAN GET ADDRESS IN CORRECT ORDER.
    MOVE.L    D2,A1            ;MOVE CORRECTED ADDRESS TO A0
MOVE_AGAIN
    MOVE.B    [A0],[A1]+      ; DO MOVE OPERATION.
    SUB      #1,D3
    BNE      MOVE_AGAIN
    JMP      LOOP_REENTRY     ;GO BACK AND WAIT FOR COMMAND
*****

```

68000 ONLY

68008 ONLY

```

513 *
514 *  COMMAND 3 ... USER DEFINED IO FUNCTION
515 *
516 *  THE USER DEFINED IO FUNCTION ALLOWS THE USER TO IMPLEMENT
517 *  SPECIALIZED DISPLAY/MODIFY FEATURES.  THE FEATURES ARE ACCESSED
518 *  FROM THE 64000 KEYBOARD BY ENTERING:
519 *      display io_port [ADDR_LIST]
520 *  OR  modify io_port [ADDRESS] to [VALUE_LIST]
521 *  OR  modify io_port [ADDRESS] thru [ADDRESS] to [VALUE_LIST]
522 *
523 *  IN RESPONSE TO THIS INPUT THE 64000 WILL WRITE THE FOLLOWING
524 *  COMMAND BLOCK:
525 *  MONITOR_CONTROL+0 word  8003H
526 *  MONITOR_CONTROL+2 word  bit0 : 0 = read, 1 = write
527 *                        bit1 : 0 = byte, 1 = word
528 *  MONITOR_CONTROL+4 word  byte address to be used, 16 bits only
529 *  MONITOR_CONTROL+6 word  data (if byte operation data right
530 *                        justified)
531 *  WHEN THE COMMAND IS COMPLETE, THE 64000 EXPECTS ONE OF THE
532 *  FOLLOWING RESPONSES:
533 *  MONITOR_CONTROL = 0000H = successful operation
534 *                  0003H = failure
535 *
536 *  AS AN EXAMPLE OF WHAT CAN BE DONE WITH THIS COMMAND, THE
537 *  FOLLOWING CODE WILL READ/WRITE MEMORY IN THE LOWEST AND HIGHEST
538 *  32K BYTE BLOCKS OF MEMORY.
539 *  EITHER WORD OR BYTE ACCESSES CAN BE DONE.
540 IO_FUNCTION
541     MOVE.W    MONITOR_CONTROL+4,A0 ;MOVE ACCESS ADDRESS TO REGISTER.
542 *           ;NOTE, ADDRESS IS SIGN EXTENDED.
543     BTST     #0,MONITOR_CONTROL+3 ;TEST FOR R/W OPERATION.
544     BNE     IO_WRITE
545 IO_READ
546     BTST     #1,MONITOR_CONTROL+3 ;TEST FOR BYTE OR WORD OPERATION.
547     BNE     IO_READ_WORD
548 IO_READ_BYTE
549     MOVE.B   [A0],MONITOR_CONTROL+7
550     JMP     LOOP_REENTRY          ;GO BACK AND WAIT FOR NEXT COMMAND.
551 IO_READ_WORD
552     MOVE.W   [A0],MONITOR_CONTROL+6
553     JMP     LOOP_REENTRY          ;GO BACK AND WAIT FOR NEXT COMMAND.
554 IO_WRITE
555     BTST     #1,MONITOR_CONTROL+3 ;TEST FOR BYTE OR WORD OPERATION.
556     BNE     IO_WRITE_WORD
557 IO_WRITE_BYTE
558     MOVE.B   MONITOR_CONTROL+7,[A0]
559     JMP     LOOP_REENTRY          ;GO BACK AND WAIT FOR NEXT COMMAND.
560 IO_WRITE_WORD
561     MOVE.W   MONITOR_CONTROL+6,[A0]
562     JMP     LOOP_REENTRY          ;GO BACK AND WAIT FOR NEXT COMMAND.
563 *****
564 *  THE COMMAND TABLE FOLLOWS:
565 *  IT CONSISTS OF 1 LONG WORD FOR EACH ENTRY.

```

Emulator/Analyzer
 Monitor Program Listing - 68000/68008

```

566 *   THE MSBYTE SHOULD CONTAIN THE COMMAND NUMBER.
567 *   THE LOWER 6 BYTES CONTAIN THE COMMAND ENTRY ADDRESS.
568 *
569 CMD_STRT
570   DC.L      2000000H+MOVE_MEM
571   DC.L      4000000H+MOVE_MEM      ***** 68000 only *****
572   DC.L      0000000H+ARE_THERE
573   DC.L      1000000H+EXIT_MON
574   DC.L      3000000H+IO_FUNCTION
575 CMD_END
576 *****
577 MONITOR_END
578   SKIP
579 *****
580 *
581 *   MONITOR STORAGE FOLLOWS:
582 *
583 *****
584 *
585 *   MONITOR_REGISTERS IS A SYSTEM GLOBAL WHICH DEFINES
586 *   AN ARRAY USED TO TEMPORARILY STORE THE PROCESSOR'S REGISTERS.
587 *   THE FIRST MEMBER OF THE ARRAY CONTAINS THE NUMBER OF BYTES
588 *   IN THE ARRAY.
589 *
590 MONITOR_REGISTERS
591           DC.W      REG_END-$
592 PCL           DC.W      0
593 PCH           DC.W      0
594 PSTATUS      DC.L      0
595 PREGS
596 DATA_0_7    DS.L      8
597 ADDR_0_6     DS.L      7
598 REG_SUB_END
599 ADDR_7       DC.L      0
600 USPT         DC.L      0
601 REG_END
602 *****
603 *
604 *   MONITOR_MESSAGE IS AN OPTIONAL SYSTEM GLOBAL WHICH CONTAINS
605 *   THE ADDRESS OF AN ASCII STRING TO BE DISPLAYED ON THE HOST
606 *   SYSTEM'S STATUS LINE WHEN A BREAK INTO MONITOR OCCURS. THE
607 *   STRING ITSELF SHOULD BEGIN WITH A BYTE CONTAINING THE NUMBER
608 *   OF CHARACTERS IN THE STRING (A MAX OF 30 CHARS. WILL BE
609 *   DISPLAYED). A NULL MESSAGE IS INDICATED BY AN ADDRESS OF
610 *   OF ZERO IN MONITOR_MESSAGE.
611 MONITOR_MESSAGE DC.L 0
612 *
613 *****
614 *   THE FOLLOWING VARIABLES CONTAIN THE DEFAULT STACK SPACE,
615 *   AND A FEW WORDS TO SAVE THE
616 *   ADDITIONAL INFORMATION WHICH IS STACKED DURING BUS ERROR
617 *   OR ADDRESS ERROR EXCEPTIONS.
618 *
619   DS.W      8           ; SAVE 8 WORDS FOR STACK TEMP.

```

```
620 SP_TEMP
621 SWBKPT_FLAG DS.W 0
622 *                               BREAKING WHILE IN MONITOR.
623 *   STAT1-STAT4 SAVE THE BUS ERROR/ADDRESS ERROR INFO.
624 *   THIS INFO IS SAVED FOR THE CONVENIENCE OF THE USER AND
625 *   IS NOT USED BY THE MONITOR FOR ANY OTHER PURPOSES.
626 STAT1          DC.W    0      ;   EXCEPTION STATUS WORD
627 STAT2          DC.W    0      ;   ACCESS ADDRESS HIGH
628 STAT3          DC.W    0      ;   ACCESS ADDRESS LOW
629 STAT4          DC.W    0      ;   INSTRUCTION REGISTER
630 *****
631 *
632 *   MONITOR_CONTROL IS A SYSTEM GLOBAL WHICH DEFINES
633 *   AN ARRAY USED TO PASS PARAMETERS BETWEEN THE EMULATION
634 *   MONITOR AND THE HOST.
635 *
636 MONITOR_CONTROL DC.W    0
637 MESSAGE_ADDR
638 BYTE_COUNT      DC.W    0
639 SRC_ADDR        DC.L    0
640 DST_ADDR        DC.L    0
641
642 *****
643 *
644 *   XFR_BUF IS A SYSTEM GLOBAL WHICH DEFINES
645 *   AN ARRAY USED TO PASS DATA BETWEEN THE HOST AND
646 *   USER MEMORY.
647 *
648 XFR_BUF         DS.W   128
649
650     END
```

NOTES

Chapter 9

EMULATION AND ANALYSIS OPERATIONAL THEORY

OVERVIEW

This chapter will:

- Provide a functional description of the HP 64000 emulation system.
- Describe the emulation system components, including the bus structures.
- Provide a functional description of the internal analyzer.
- Describe some of the more prominent emulator operating considerations.

GENERAL

In this chapter, whenever 680XX is mentioned (even as part of a filename) both the 68000 and the 68008 are assumed unless there are differences between the two processors. In that case, the differences will be noted. When the name 680XX appears in any program listings, just insert the appropriate name of your processor.

INTRODUCTION

The basic development system consists of a logic development station having a flexible disc drive, an optional hard disc and printer, editor software, and various development applications implemented by additional hardware and/or software. The 680XX emulation system is one of these development applications.

The hardware needed for a complete 680XX emulation system includes an emulation control board, emulation probe (pod), memory control board, memory boards, and an internal analysis board. The target system discussed in this chapter represents a system having a microprocessor, control circuits, and memory (ROM/RAM). The target system is not required for emulation. In addition to the 680XX emulation software you need 680XX assembler and linking software to prepare your programs for emulation. The interfaces for the emulation and analysis subsystems are the development station interface consisting of the host processor bus, and the target system interface consisting of the emulation pod (with plug-ins), and the pod bus.

EMULATION OVERVIEW

The basic feature set of a microprocessor emulator includes: displaying/modifying processor registers, displaying/modifying memory, displaying/modifying io_ports, displaying global and local

symbols, loading memory, running programs, setting program breakpoints, and program stepping. The analysis of program execution using a logic analyzer is also considered to be a basic feature of an emulator. Analysis is discussed later in this chapter. Many of the above features or subsets of features are implemented by the emulation processor executing a short program called the "Emulation Monitor". The features implemented are: displaying/modifying processor registers, displaying/modifying "target" memory, displaying/modifying io_ports, executing a program from a specific starting address, and program stepping.

Features which do not require use of the emulation monitor are: displaying/modifying "emulation" memory, asserting/releasing the target processor "reset" control, using analysis functions, and other system activities such as displaying symbols or configuring emulation hardware.

The Emulation Control board (see figure 9-1) interfaces the emulator pod to the emulation memory control and emulation analysis buses. The HP 64000 host processor can exercise the emulation processor "reset" and bus activity stopping functions directly through this board. However, the two main functions of the control board are: to convert unique pod timing signals to compatible memory and analysis bus signals and to provide a channel to the pod for hardware configuration.

The host processor communicates with the emulation monitor by transferring data to and from emulation memory. Data transfer is accomplished through the Memory Controller board into the static ram Memory boards. In addition to providing an access port for the host processor into emulation memory, the memory controller contains a hardware mapper that is programmed to map the emulation processor memory address space into emulation/target/guarded and ram/rom memory spaces.

The Emulation Monitor program is supplied in source form, and is listed in chapter 8. This source listing may be customized to work with specific requirements of the target system. The monitor program should be assembled and linked with the relocatable modules that form the software for the target system. This absolute file can then be loaded into memory using the emulator "load supervisor emulation memory" command.

If parts of the absolute file are to be loaded into memory and mapped as "target" memory, loading will be performed in two steps. The parts of the file which go into emulation memory are loaded first. The emulation processor is allowed to run by releasing "reset". If the emulation processor is executing the monitor program, then the second step of the load is performed. If the emulation processor is not executing the emulation monitor program, an emulator "break" is asserted. See chapter 8 for specific information on the break function. If, at this time, the emulation processor is still not executing the monitor program, the "target" load is aborted and an error message will be displayed on the status line.

Once it is determined that the emulation processor is executing in the emulation monitor program, those parts of the absolute file which are mapped to "target" memory will be loaded. The host processor passes the absolute file data to the monitor program which then writes the data into memory.

If none of the absolute file is loaded into target memory, "reset" is not released; the host processor status will indicate "Reset", and the emulation processor will not be running. The "run" command will have to be entered on the development station keyboard before those features which require the emulation monitor are available.

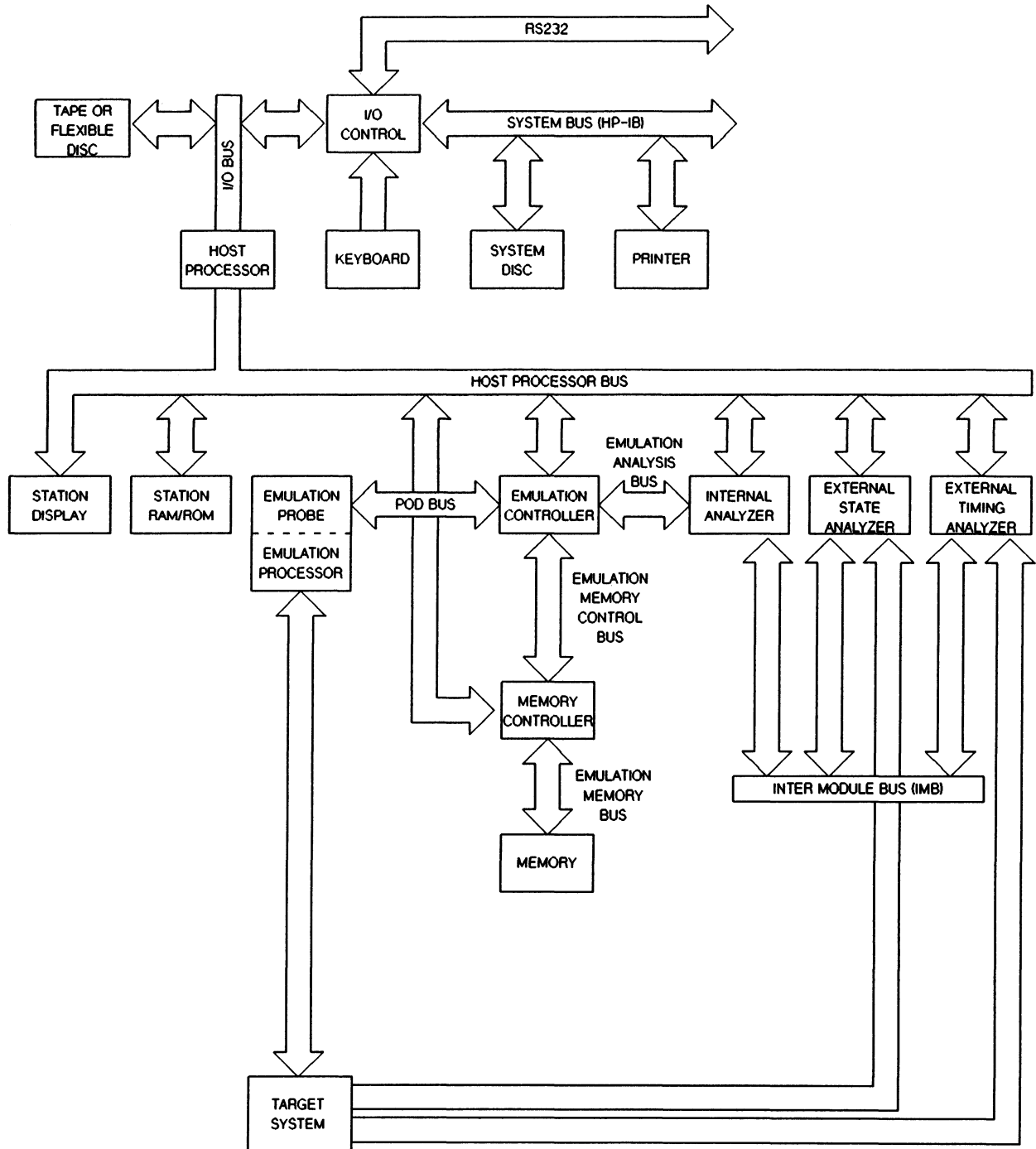


Figure 9-1. HP 64000 Logic Development System Simplified Block Diagram

The host processor determines whether an emulation monitor program is in memory by searching the link_sym file, of the same name as the absolute file, for the following global symbols: MONITOR_CONTROL, MONITOR_REGISTERS, XFR_BUF, MON_6424XS. Unless all of the symbols are found, the emulation monitor will not be recognized. These symbols are required to be mapped to emulation memory, and if any one of them is not present, an error message will be displayed.

Commands from the host processor are written into "MONITOR_CONTROL". When the emulation processor is executing the emulation monitor program, bit 15 of "MONITOR_CONTROL" is tested to see if it is set. If bit 15 is set, a command is present; the monitor uses the lower byte of "MONITOR_CONTROL" as a command number. If that command number is valid, the command is executed and, upon completion, bit 15 of "MONITOR_CONTROL" is cleared by the emulation monitor. The host processor looks for bit 15 to be cleared; if bit 15 is cleared, the command is assumed to have been completed successfully, otherwise, the host processor, after a time period has elapsed, will assume that the emulation processor is not executing in the monitor program.

There is a special command in the emulation monitor referred to as the "Are you there?" command. The host system uses this command to determine whether the emulation processor is currently executing in the emulation monitor. As described above, the "Are you there?" command is placed in "MONITOR_CONTROL"; if bit 15 of "MONITOR_CONTROL" is cleared, then it is assumed that the emulation processor is executing in the monitor program. If the timeout occurs then it is assumed that the emulation processor is executing a user program.

While you may run programs by releasing "reset" and allowing the emulation processor to run from that initial condition (the emulation monitor is not required), typically a specific run starting address is desired. The "run from <address>" emulation command sets the program counter value, which is maintained in the host processor, to <address>. The host processor transfers its copy of the emulation processor's registers to the emulation monitor's data block which starts at "MONITOR_REGISTERS"; then the host processor puts the "exit monitor" command into "MONITOR_CONTROL". Assuming that the emulation processor is executing the monitor program, the emulation processor loads its registers from "MONITOR_REGISTERS" and transfers control to the specified program counter address.

The emulation processor can be diverted from the user program back to the emulation monitor program using the emulator "break" function (see chapter 8 for specific information on the break function). Hardware breakpoints can be set using the logic analyzer; trigger points are set which in turn can assert the break function. Hardware breaks can also be caused by attempts to access guarded memory, or attempts to write into ROM. In addition, the emulation "break" command can be entered to assert the break function.

The "step" function is used to sequence slowly through a portion of program code. This feature is a function of the analysis module and is not available without the analysis module in place in the system. The analysis module is programmed to cause an emulation break to occur upon the recognition of the next PC address as it appears on the emulation bus. The emulation hardware then forces entry into the monitor routine in order to display the processor status. The next step command causes the analysis unit to be re-programmed to look for the next PC address in the processor to begin the break into the monitor again. For further information about program stepping, refer to the paragraph entitled "Emulator Operating Considerations" given later in this chapter.

The emulation system provides the capability to set and clear software breakpoints. This capability allows the user the opportunity to affect a "break on execution", a function that is important to those processors that do extensive prefetching in normal operation. For further information about

software breakpoints, refer to the paragraph entitled "Emulator Operating Considerations" given later in this chapter.

MULTI-MODULE EMULATION AND ANALYSIS

The HP 64000 system allows the use of emulation and analysis features in an interactive manner between an emulator and another module; a module that could be another emulator or a state or timing analyzer. Interaction allows the integration of development work on multi-processor designs, or more elaborate and detailed analysis of a design, or both.

In particular, the capabilities that are supported include: simultaneous initiation of multiple measurements, using the results of one measurement to control another, and coordinating execution of a program with the initiation of a measurement.

SYSTEM BUS STRUCTURES

The HP 64000 system is designed with independent buses for the development station environment and the emulation system. Since the development station (host) processor and the emulation system operate on separate buses, both can be running at the same time with no contention for system resources. Figure 9-1 illustrates the HP 64000 Logic Development Station bus orientation. The basic bus structures for the HP 64000 are briefly described in the following paragraphs.

System Bus--The address, data, and control buses make up the HP 64000 system (HP-IB) bus. Communication between the printer, system disc, and development stations occurs via the system (HP-IB) bus through the I/O control.

Emulation Memory Control Bus--The address, data, and control buses for emulator operation are included in the emulation memory control bus.

Emulation Analysis Bus--Communication between the emulation controller and the analysis module takes place through the emulation analysis bus.

Host Processor Bus--The host processor bus is the path through which the development station processor communicates with the emulation system, the display, and development station processor memory.

I/O Bus--The Input/Output bus is dedicated to input and output devices of the HP 64000 station. It handles data transfers to and from the flexible disc drives, the keyboard, the system disc, the printer, and the host processor.

Intermodule Bus--The intermodule bus connects the appropriate control boards in a multi-module system and carries signals related to sequence, timing, and triggering between modules.

All data transfers in the system occur on these buses. The emulation control board obtains the desired information from the target system memory. The host processor then obtains the data from the emulation control board via the host processor bus. From there, the data is passed to the

display controller for display on the CRT. To display emulation memory, data in the 680XX emulation memory is accessed by the development station processor through the memory board via the development station processor bus. From that point on, data is transferred to the display.

EMULATION CONTROL BOARD FUNCTIONAL DESCRIPTION

The emulation control board performs three interfacing functions. An interface is provided between the emulation pod and the emulation memory, between the emulation pod and the internal analysis bus, and between the emulation pod and the host processor.

The emulation memory interface circuitry generates the memory strobes and status signals required by the emulation memory controller. The address used by the memory controller is captured in the address latches during the address portion of the bus cycle. This address is used by the memory controller to determine if the bus cycle should have access to emulation memory. Bidirectional data buffers on the emulation control board provide the data path between the emulator pod and emulation memory during the data portion of the bus cycle.

The analysis bus interface generates the analysis clock, address, data, and status signals for the analysis bus. The interface operates in either the bus cycle mode or the execution data mode. The operating mode is selectable by the user during emulation. The bus cycle mode generates an analysis clock for every bus cycle. The address, data, and status generated for that clock comes directly from the bus cycle. In the execution data mode, analysis clocks are generated on every non-instruction fetch bus cycle, and whenever the processor executes a fetched instruction. The address, data, and status that is generated for the execution clocks is produced from the execution activity circuitry located on the emulation control board. The information is derived from previous instruction fetch bus cycles, but is not passed to the analysis bus until the instructions have actually been executed by the processor.

The interface between the emulation pod and the host processor allows the host to configure the emulation pod during reset. While the emulation processor is executing, the interface is used to monitor the pod status, and to request the processor to stop momentarily, when necessary.

EMULATOR PROBE (POD) FUNCTIONAL DESCRIPTION

The emulator pod (see figure 9-2) is used to replace the processor in the target system. It contains a 680XX microprocessor and interface circuitry for the target system and the emulation control board. A memory mapper in the pod is used to determine which resources, target system or emulation memory, will be used for each bus cycle. The buffer control circuitry uses the output from the mapper to control the direction, and the enables, of the interface circuitry.

One of the major features of an emulator is the ability to stop the execution of the target system code and provide detailed information about the details of the processor. The control circuitry in the emulation pod controls this process by causing the target system code to be interrupted and the emulation monitor to execute, so that the processor status can be extracted.

Plug-in Leads

There are two plug-in leads on the front end of the Emulator Probe. The plug-in leads are labeled DTACK and USER MEM. The two plug-in leads are described below. (See figure 9-3).

DTACK - This input was designed to connect to the target system when access to emulation memory by DMA devices is desired. "DTACK" provides the emulation memory's "ready" response to the DMA device access.

USER MEM - USER MEM is a low true signal that indicates when a bus cycle is accessing the target system rather than emulation memory. If USER MEM is high, then the system is accessing emulation memory. This allows you to disable your memory when emulation space is indicated and, thus, overlay emulation memory for DMA devices. Note that overlaying emulation memory for processor use is done by the pod, and you are not required to do anything. If the target system hardware disables the bus drivers on emulation memory accesses it is not necessary to use USER MEM.

MEMORY CONTROLLER FUNCTIONAL DESCRIPTION

The Memory Controller is the interface between Emulation Memory, the Emulator Controller, and the HP 64000 operating system. The Memory Controller also maps the targets address received from the Emulation Controller via the emulation memory control bus into available emulation memory. The mapping process is performed by mapper RAMs which reside on the Memory Controller. A read/write operation to Emulation Memory is performed via the Emulation Memory bus.

The Mapper RAMs also output signals which specify what type of memory the given block of memory is supposed to act like (RAM, ROM or guarded memory), or whether a given address is to be regarded as emulation or target address space. The Memory Controller will break into the monitor when a guarded memory access is attempted and, if optionally configured, when a write to ROM is attempted.

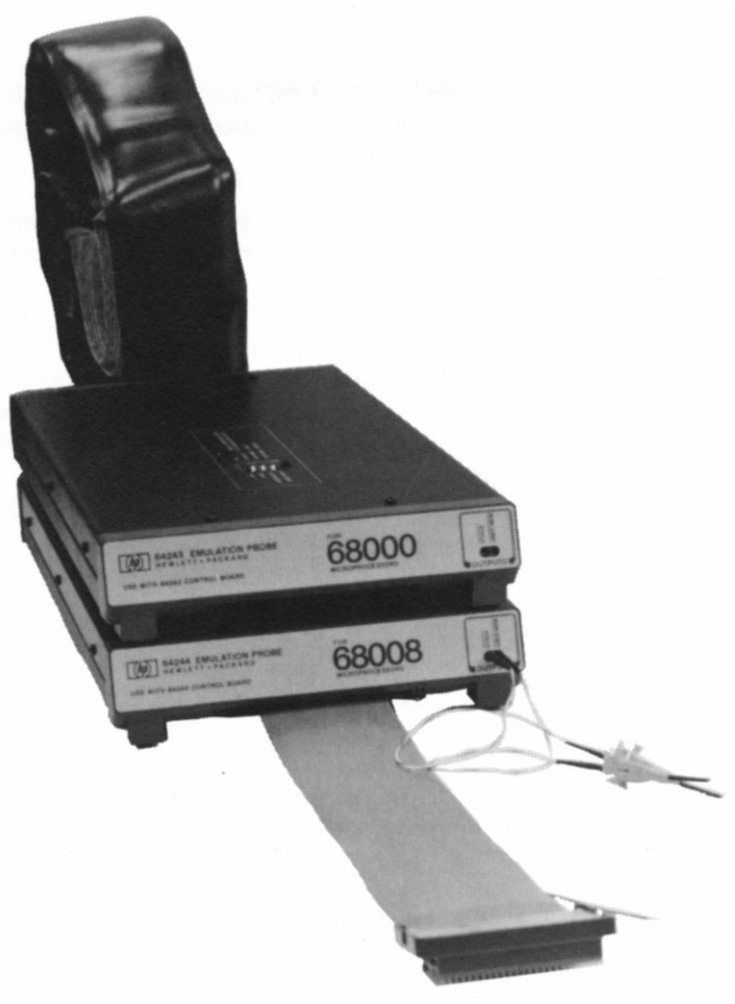


Figure 9-2. Emulation Probe (Pod)



Figure 9-3. Pod Plug-in Leads

INTERNAL ANALYZER FUNCTIONAL DESCRIPTION

The HP 64302A Wide Logic Analyzer is the appropriate internal analyzer to be used with the 680XX emulation system. The entire analyzer is contained on one printed circuit board that fits in the development station card cage. The specifications for the wide logic analyzer are given in table 9-1.

It simultaneously captures address, data, and control states from the emulated target processor via the emulation analysis bus and is capable of storing the states as 256 forty-eight bit words. The analyzer is also capable of displaying the information on the development station CRT in easy-to-read mnemonic format. The wide logic analyzer has the ability to specify trace points in combinations of address, data, and status; this makes the wide logic analyzer a valuable addition for debugging the target system's hardware and software.

The analyzer's operations are divided into two principal modes; one, the halt mode coordinates the CPU and analyzer, and two, the run mode performs emulator state capture, storage, and decoding.

The halt mode is used to: (1) perform host processor initiated activities, such as loading state recognition terms and the control word before the analysis is performed, and (2) to unload the stored states after the trace is complete.

The run mode begins when real-time analysis is started and ends when the trace is completed. During the run mode, the analyzer captures emulator states from the emulation analysis bus, qualifies the state for storage in the analyzer RAMs and determines when the trace is complete. The run mode operations are controlled by the analyzer timing signals and cannot be interfered with by a host processor read or write. In this mode, only control flags can be accessed by the host processor.

In the run mode, through state control, the state storage accumulates states; each state consisting of 48 bits of address, data, and status from the emulation analysis bus and 24 bits of count. The count is either a cumulative time value or a cumulative count of qualified states. State storage contains the trigger state plus all qualified states. Once a trace has been initiated, the analysis module begins to look for "states" that satisfy the trace specification until terminated by a "stop trace" or "halt" command, or until the trace specification is met. Trace specifications are loaded into the analysis board trigger hardware via the host processor bus. The analysis board continually monitors transactions between system elements connected to the emulation analysis bus. When the required trigger occurs on the emulation analysis bus, the analysis board stores the previous or following bus states in a dedicated trace memory. When the display trace command is initiated, the contents of trace memory is retrieved, formatted, and displayed on the CRT screen. When the trace is complete the timing/control function sets the measurement complete flag true and terminates the run mode. The host processor then unloads the state storage RAMs and displays the states to the user. If desired, the data can be displayed in the mnemonics of the processor being emulated.

The wide logic analyzer communicates with the following areas of the HP 64000 Logic Development System.

Emulation Analysis Bus: Monitors the address, data, and control lines of the target microprocessor. Generates emulation break.

Host Processor Bus: Sends and receives control commands to and from the development station's central processing unit (CPU). At the CPU's request, sends all stored information to the CPU to be formatted and displayed.

Inter Module Bus (IMB): Sends and receives control signals to other modules.

Table 9-1. Model HP 64302A Internal Analyzer Specifications

Power up configuration:

No IMB lines driven, except Gated Master Clock.
IMB configuration invalid.
BNC1 and BNC2 not driven.

Storage capabilities:

Pre-store, post-store and combination pre/post-store.
256 49-bit words at a maximum 6 MHz acquisition rate.
48 parallel channels
24 Emulator address bits.
16 Emulator data bits.
8 Emulator control bits.
24 count bits.
State (number of occurrences), or
Time (time measurement between states).

Break capabilities:

Trace Point and Measurement Complete.

Indexing capabilities:

Eight modes using Range, AME1, AME2 and Count
Qualify signals. Range is 24 bit emulator address in 1,0, or X (don't care).
Others are 48 bit emulator bus in 1,0, or X.

Trigger on Nth occurrence: N = 1 to 65,535.

InterModule Bus (IMB) capabilities:

Receive and drive Master Enable.
Receive or drive Trigger Enable.
Receive and/or drive Trigger.
Drive Gated Master Clock.

EMULATOR OPERATING CONSIDERATIONS

Software Breakpoints

The emulation system has the capability to perform a "break on" execution.

Any valid address may be used as a breakpoint. If the address is not valid, the break will not occur. A valid address may take the form of a number, an expression, or a symbol, that locates the first byte of a valid instruction. Valid addresses can be identified from a program listing or by displaying memory in mnemonic format. The program must be loaded into memory before listing or display.

The "modify software breakpoints" and "display/list software breakpoints" commands are used with the break on execution feature.

Setting breakpoints, combined with a "trace before SWBK_ENTRY" command, provides a convenient tool for software analysis. SWBK_ENTRY is a global symbol defined in the monitor program listing in chapter 8.

As many as 16 breakpoints are maintained in a table that is kept intact at the end of the emulation session. The table is available when the emulation session is resumed.

The data found at the breakpoint address is saved in the breakpoint table. A special instruction (TRAP #, where # indicates the trap number chosen in the configuration questions and the table entry in the monitor program that is 'un-commented') replaces the data in the program, and when execution of the program reaches the TRAP # code, a break into the emulation monitor program occurs. The processor will be running in the monitor program and the message:

"Software breakpoint address = 0XXXXXXH"

will be displayed showing the address of the current program counter.

If the software breakpoints are displayed, they will reflect a change in status from "pending" to "inactivated".

The data, from the breakpoint table, will be restored to the program, and the program counter will reflect the breakpoint address.

The system is now ready to continue from the break point in the program via a "run" or "step".

A "step" function (to get the processor past the breakpoint address) followed by "modify software_breakpoint set 00XXXXXXH", will reactivate the breakpoint; or "modify software_breakpoint set" will reactivate all inactivated breakpoints.

If the TRAP # code is executed and the address of execution does not correspond to an entry in the table, the following message will be displayed:

"Undef. software break trap"

USING SOFTWARE BREAK IN REAL TIME. The emulation processor will not execute a communication handshake with the host processor if "run" is restricted to real time. As a result, a transfer from target program to monitor program will not be identified.

When using software breakpoints, along with real time "run", a break to initiate the communication handshake and breakpoint processing must be done manually.

Using the "trace before SWBK_ENTRY" command, waiting for the "trace complete" status message, followed by the "break" command from the softkey, will result in a display of the breakpoint address and a reminder that the processor is "running in monitor".

Chapter 10

PRINTER SIMULATED I/O

OVERVIEW

This chapter will:

- Describe printer simulated I/O.
- Explain how to establish a printer interface.
- List the printer control codes.
- Provide a sample Pascal printer I/O program.

The printer simulated I/O feature allows the emulator to control a printer that is connected to the HP 64000 system. You can open a printer file, write to the printer, and close the printer file. To do this, you must write a program that passes printing control codes and ASCII text to the development station. You assemble or compile the program and link it with your other programs in the emulation environment. The following paragraphs describe how to establish the printer interface with your program.

The printer I/O interface (see figure 10-1) requires a memory location, called the printer control address (CA), to which I/O handshaking codes are sent by your program and the HP 64000 system. The HP 64000 samples the printer CA periodically looking for commands. The printer CA must be initially defined in your program and in the simulated I/O configuration. When more than one simulated I/O interface is to be implemented, you must make sure that each I/O program assigns a unique address for each CA.

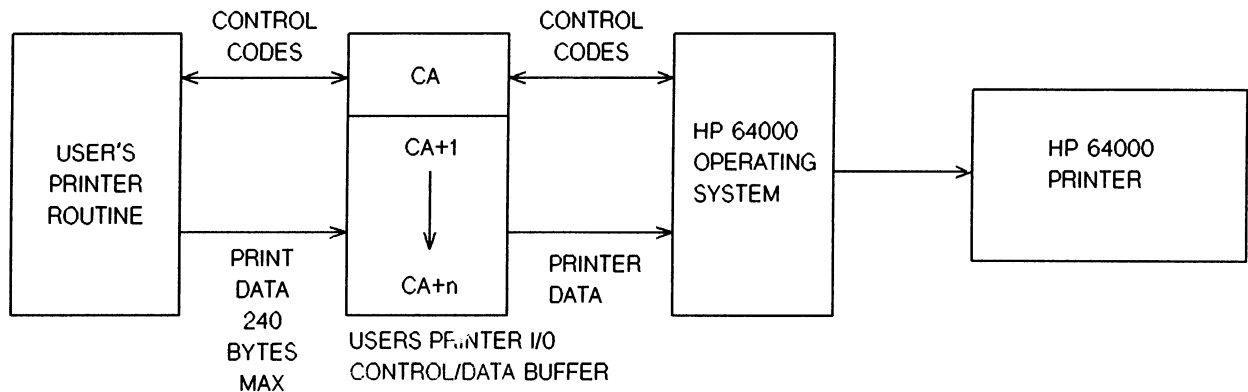


Figure 10-1. Printer Interface

The address for the printer CA is entered during simulated I/O configuration. The emulator must not be restricted to real time runs when using the printer simulated I/O. The printer CA location must be located in memory space assigned as user RAM or emulation RAM. It is recommended that the printer CA location be in emulation RAM since this allows your programs to run faster.

The write-to-printer code includes supplemental information. This supplemental information is contained in the locations following the printer CA, i.e., CA+1 thru CA+n. The supplemental information must be placed in locations CA+1 thru CA+n BEFORE the control code is placed in the printer CA. When this is not done, the HP 64000 may respond to the control code in the printer CA before the supplemental data is set into locations CA+1 thru CA+n.

OPENING THE PRINTER FILE (80H)

Before writing to the printer, your program must request that the printer interface be opened. This is done by placing the code 80H into the printer CA.

NOTE

During the time that a simulated I/O printer file is open, no other user can access the printer. Thus, be sure to close the file when finished.

The HP 64000 responds by opening the printer file and returning a 00 to the printer CA. When the file cannot be opened, error codes are returned as shown in table 10-1.

After the file is opened, your program may write to the printer as described in the next paragraph.

WRITING TO THE PRINTER (82H)

To write a record to the printer, your program must first place the record length, in bytes, into location CA+1. The record length must be a minimum of two bytes and may be a maximum of 240 bytes in two byte increments. That is, the record must always contain an even number of bytes. Odd bytes should be padded with a space (20H).

Next, place the ASCII codes of the characters to be printed in locations CA+2 thru (CA+2)+n. Finally, place the write to printer code, 82H, into the CA.

The HP 64000 responds by supplying the write record to the printer and returning a 00 to the printer CA. The HP 64000 automatically sends a carriage return/linefeed to the printer following the user data. If the write-to-printer record is not accepted, an error code is returned as listed in table 10-1.

CLOSING THE PRINTER FILE (81H)

Your program closes the printer file by placing code 81H into the printer CA. The HP 64000 responds by closing the file and returning code 00 to the printer CA. The HP 64000 will perform a form feed automatically.

When the close is not accepted, an error code is returned to the printer CA as shown in table 10-1.

Table 10-1. Printer Control Codes

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:			
	Address	Contents	Address	Contents	Error Code	
OPEN PRINTER FILE	CA	80H	CA	00	01 thru 08 09: File is already open. 10-14: NA	
CLOSE PRINTER FILE	CA	81H	CA	00	01 thru 08 09: File is already closed. 10-14: NA	
WRITE TO PRINTER	CA	82H	CA	00	01 thru 08 09: File is not open. 10, 11, 13 & 14: NA 12: Record length exceeded 240 bytes.	
	CA+1	Record Length in bytes (240 max.)				The HP 64000 accepts the record and causes it to be printed.
	CA+2 ↓ (CA+2) +n	Record byte 1* ↓ Record byte n*				

*All display characters must be formatted in ASCII code. A code greater than 0F0H will not be accepted by the HP 64000 program.

NA = Not Applicable.
 See Appendix E for complete error listing.

PRINTER EXAMPLE PROGRAM

A Pascal program listing is included as an example (see figure 10-2) . The program is `Print_SIO` (Print, Simulated I/O), shown in figure 10-2. It uses three procedures to open, write-to, and close the printer file. The program establishes the printer CA at address 3F00H, then writes the character string of printable ASCII codes from A to ~ to the printer.

```
"680XX"

PROGRAM PRINT_SIO;
  {This program uses the simulated I/O to the printer and is}
  {written in PASCAL. The routine opens the printer, writes}
  {the printable ASCII characters to the printer, and then }
  {closes the printer.                                     }

$ASM_FILE$
$EXTENSIONS+$

TYPE
  INTEGER = SIGNED_16;
  PTR_BYTE = ^BYTE;
  PTR = RECORD
    CASE BOOLEAN OF
      TRUE : (P : ^ BYTE);
      FALSE : (I : SIGNED_32);
    END;
  BITS = (B7,B6,B5,B4,B3,B2,B1);
  SOB = RECORD
    CASE BOOLEAN OF
      TRUE : (B:BYTE);
      FALSE : (S:SET OF BITS);
    END;

VAR
  PRNT_CA :PTR;    {printer control address}
  ERR_CODE :BYTE;
  ADDRS : PTR;
  ADDRD : PTR;
  LENGTH : BYTE;
  DATA : SOB;
  MASK : SOB;
  TEMP : BYTE;
```

Figure 10-2 Program PRINT_SIO

```
PROCEDURE SIO_WAIT (ADDRESS : PTR_BYTE; VAR ERROR_CODE: BYTE);
{This procedure provides a WAIT loop for HOST response. }
{The procedure waits for the host system to respond from}
{some command by setting bit 7 of data in some specified}
{control address to 0. }

BEGIN
  REPEAT
    ERROR_CODE := ADDRESS^;
    UNTIL ERROR_CODE > -1;
  END;

PROCEDURE XFR_RBUF(ADDRESS:PTR_BYTE);
{ This procedure provides transfer of data from specified }
{ address to the printer data area. }

BEGIN
  ADDRS.P := ADDRESS;
  ADDR.D.I := PRNT_CA.I + 1;
  LENGTH := ADDRS.P^;
  DATA.B := LENGTH;
  ADDR.D.P^ := LENGTH;
  WHILE LENGTH <> 0 DO
    BEGIN
      ADDRS.I := ADDRS.I + 1;
      ADDR.D.I := ADDR.D.I + 1;
      ADDR.D.P^ := ADDRS.P^;
      LENGTH := LENGTH-1;
    END;
  END;

PROCEDURE SIOP_OPEN(VAR ERR_CODE : BYTE);
{ This procedure opens the printer file and returns the }
{ status of 'open' in ERR_CODE . }

BEGIN
  ADDRS.I := PRNT_CA.I;
  ADDRS.P^ := 80H;
  SIO_WAIT(ADDRS.P,ERR_CODE);
END;

PROCEDURE SIOP_CLOSE(VAR ERROR_CODE : BYTE);
{ This procedure closes the printer file and returns the }
{ status of operation in ERR_CODE. }

BEGIN
  ADDRS.I := PRNT_CA.I;
  ADDRS.P^ := 81H;
  SIO_WAIT(ADDRS.P, ERR_CODE);
END;
```

Figure 10-2 Program PRINT_SIO (Cont'd)

```

PROCEDURE SIOP_WRITE(ADDRESS:PTR_BYTE; VAR ERROR_CODE: BYTE);
  { This procedure writes the specified data to the printer }
  { file and returns the status of 'open' in ERR_CODE.      }

BEGIN
  XFR_RBUF(ADDRESS);
  MASK.B := 1;
  DATA.S := DATA.S*MASK.S;
  IF DATA.B <> 0
    THEN BEGIN
      ADDRS.I := PRNT_CA.I + 1;
      LENGTH := ADDRS.P^ + 1;
      ADDRS.P^ := LENGTH;
      ADDRS.I := PRNT_CA.I + LENGTH + 1;
      DATA.B := 20H;
      ADDRS.P^ := DATA.B;
    END;
  ADDRS.I := PRNT_CA.I;
  ADDRS.P^ := 82H;
  SIO_WAIT(ADDRS.P, ERR_CODE);
END;

  { This is the main routine of the program. This routine }
  { writes the printable ASCII characters from A thru ~ to }
  { memory locations beginning at PRNT_CA.I .              }

BEGIN
  PRNT_CA.I := 3F00H;
  TEMP := 30H;
  SIOP_OPEN(ERR_CODE);
  IF ERR_CODE = 0
    THEN BEGIN
      ADDRS.I := PRNT_CA.I + 10H;
      ADDRS.P^ := 120 ;
      REPEAT
        BEGIN
          ADDRS.I := ADDRS.I + 1;
          ADDRS.P^ := TEMP;
          TEMP := TEMP + 1;
        END
      UNTIL TEMP = 120 ;

      ADDRS.I := PRNT_CA.I + 10H;
      SIOP_WRITE(ADDRS.P, ERR_CODE);
    END;
  IF ERR_CODE = 0
    THEN SIOP_CLOSE(ERR_CODE);

END.

```

Figure 10-2. Program PRINT_SIO (Cont'd)

NOTES

Chapter 11

DISPLAY SIMULATED I/O

OVERVIEW

This chapter will:

- Describe display simulated I/O.
- Explain how to establish a display interface.
- List the display control codes.
- Provide a sample Pascal display I/O program.

In this chapter, whenever 68000 is mentioned (even as part of a filename) 68008 is also assumed unless there are differences between the two processors. In that case, the differences will be noted.

The display simulated I/O feature allows the emulator to control the display in the development station. You can open the display file, write to the display and close the display file. When writing to the display, you can use the single control code 82H to scroll the display, or you can use control codes 83H and 84H to write at a specific location on the display.

To do this, you must write a program that passes display control codes and ASCII text to the development station. You assemble, or compile the program and link it with your other programs in the emulation environment. The following paragraphs describe how to establish the display interface with your program.

The display I/O interface (see figure 11-1) requires a memory location, called the display control address (CA), to which I/O handshaking codes are sent by your program and the HP 64000 system. The HP 64000 samples the display CA periodically looking for commands. The display CA must be initially defined in your program and in the simulated I/O configuration. When more than one simulated I/O interface is to be implemented, you must make sure that each I/O program assigns a unique address for the CA.

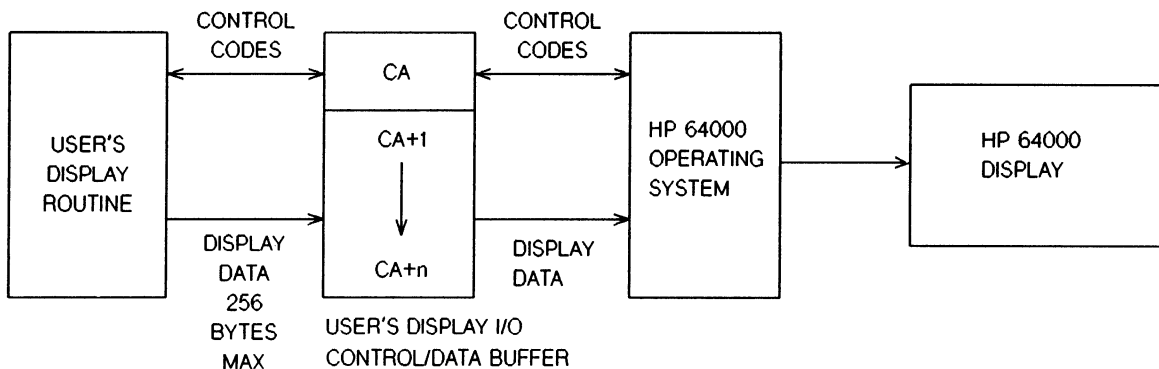


Figure 11-1. Display Interface Diagram

The address for the display CA is entered during simulated I/O configuration. The emulator must not be restricted to real time runs when using the display simulated I/O. The display CA location must be located in memory space assigned as user RAM or emulation RAM. It is recommended that the display CA location be in emulation RAM since this allows your programs to run faster.

Three display I/O codes include supplemental information. This supplemental information is contained in the locations following the display CA, i.e., CA+1 thru CA+n. The supplemental information must be placed in locations CA+1 thru CA+n BEFORE the control code is placed in the display CA. When this is not done, the HP 64000 may respond to the control code in the display CA before the supplemental data is set into locations CA+1 thru CA+n.

OPENING THE DISPLAY FILE (80H)

Before writing to the display, your program must request that the display file be opened. This is done by placing code 80H into the display CA.

NOTE

During the time the simulated I/O display file is open, the development station keyboard has no control over the display.

To regain control, press the simulate softkey to close the file. If the keyboard file is open, it, too, is closed when the softkey is pressed.

The HP 64000 responds by opening the display file, and returning a 00 to the display CA. When the file cannot be opened, error codes are returned as shown in table 11-1.

After the file is opened, your program may write on the display as described in the following paragraphs.

SCROLLING

Roll To/Write Line 18 (82H)

This is one of two methods to write to the display. It allows you to write to the bottom of the display. Sequential 82H commands cause the previously written line to roll to line 17. Thus, with this command, writing is always done on the bottom line and the previously written lines are shifted up.

To roll and write on line 18, your program must first place the line length (in bytes) into the display CA+1. The line length must be a minimum of two bytes and may be a maximum of 80 bytes, in two byte increments. That is, the line must always contain an even number of bytes. When you write an odd number of bytes, the HP 64000 pads the line with a null.

Next, place the ASCII codes of the characters to be displayed on line 18, into locations CA+2 through (CA+2)+n. Finally, place the roll to/write line 18 control code, (82H) into the display CA.

NOTE

The display characters must be formatted in ASCII codes. The development stations does not accept ASCII codes above 0F0H.

The HP 64000 responds by storing the characters in a display buffer and returning a 00 in the display CA. Although the HP 64000 responds almost immediately with 00, the actual scrolling of a line can take up to 200 msec. Thus, you may want to use a wait-routine in your program. When the display cannot be scrolled, the development station returns an error code as listed in table 11-1.

During a scrolling delay, the development station still accepts other commands. Subsequent scrolls are buffered and performed in sequence.

LINE & COLUMN

You can write to a specified line and column by using control codes 83H and 84H. First, use 83H to set up the starting position for the text, then use 84H to transfer the text. (See figure 11-2.)

Selecting The Starting Line/Column (83H)

You may specify the line number and column number at which writing will start. To do this, your program places the line number (1 thru 18) into the display CA+1, the column number (1 thru 80) into display CA+2, and then 83H into the display CA.

The HP 64000 responds by storing the line and column number and returning code 00 to the display CA. The line and column numbers are stored until writing is initiated (code 84H) or the display file is closed.

When the line and column numbers are not accepted by the HP 64000, an error code is returned to location CA as listed in table 11-1. Figure 11-2 shows the display techniques.

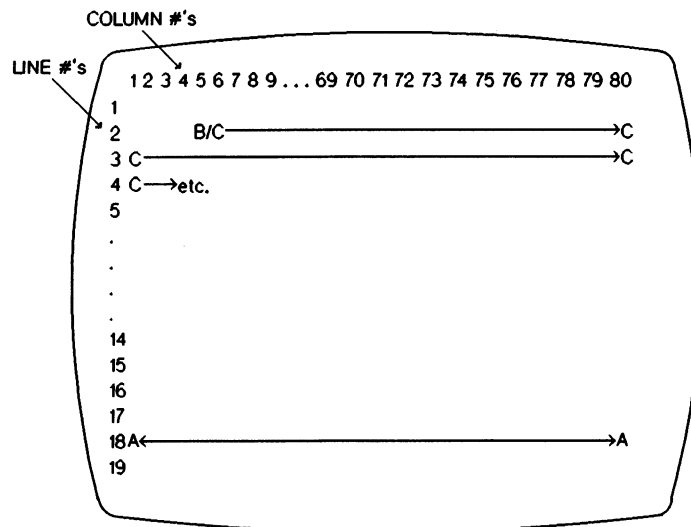
Writing From The Starting Line/Column (84H)

To write to the selected line and column, your program must first place the record length (in bytes) into the display CA+1. The record length must not exceed 255 bytes.

Next, place the ASCII codes of the characters to be displayed into locations CA+2 through (CA+2)+n. Finally, place the write from starting line/column code (84H) into the display CA.

NOTE

The display characters must be formatted in ASCII codes. The development station does not accept ASCII codes above 0F0H.



Development Station Display

Scrolling

A Code 82H causes the display to roll to line 18. Up to 80 characters may be written on a line. Sequential Roll To/Write line 18 commands cause the previous line 18 to roll to line 17, line 17 to roll to line 16, etc.

Line & Column

B/C B is the point controlled by code 83H, at which writing begins. C is the character string which is controlled by code 84H.

Figure 11-2. Display Techniques

The HP 64000 responds by displaying the record beginning at the starting line and column specified by code 83H. When the record exceeds the length of the starting line, writing continues at column one of the next line.

When the HP 64000 cannot initiate writing as requested, an error code is returned to the display CA as shown in table 11-1.

CLOSING THE DISPLAY FILE (81H)

Your program closes the display file by placing code 81H into the display CA. The HP 64000 responds by closing the file and returning code 00 to the display CA.

When the close file is not accepted, an error code is returned to the display CA as shown in table 11-1.

Pressing the simulate softkey or performing a reset-reset closes the open display file. The closing display also closes an open keyboard file.

Table 11-1. Display Control Codes

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
			VALID USER REQUEST	INVALID REQUEST	
	Address	Contents	Address	Contents	Error Code
OPEN DISPLAY FILE	CA	80H	CA	00	01 thru 08 & 14 09: code >84H or file is open 10 thru 13: NA
CLOSE DISPLAY FILE	CA	81H	CA	00	01 thru 08 & 14 09: File is already closed. 10 thru 13: NA
ROLL TO/ WRITE LINE 18	CA	82H	CA	00	01 thru 08 & 14
	CA+1	Line Length in bytes (80 max.)	The HP 64000 program stores this data in a display buffer. A delay may occur before rolling to and writing on line 18 actually occurs. A program wait may be required. If successive line 18's are written, then the preceding line 18 is rolled to line 17, 17 to 16, etc.		09: File is not open 10, 11 & 13: NA
	CA+2	Line byte 1*			12: Invalid record length
(CA+2)+n	Line byte n*				

Table 11-1. Display Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
	Address	Contents	Address	Contents	Error Code
SELECT STARTING LINE/ COLUMN	CA	83H	CA	00	01 thru 08 & 14
	CA+1	Line # (1-18)	The HP 64000 program stores the line and column numbers until a write line/column request is issued or the file is closed.	00	09: File is not open.
	CA+2	Column Number (1-80)			10, 12 & 13: NA
		11: Invalid line or column number.			
WRITE FROM STARTING LINE/ COLUMN	CA	84H	CA	00	01 thru 08, 13 & 14
	CA+1	Record length in bytes (255 Max)	The HP 64000 program displays the record starting at line/column selected by code 83H. If record exceeds one line, writing continues at column 1 of next line, etc. See Figure 11-2.	00	09: File is not open.
	CA+2	Record byte#1			10 & 12: NA
↓ (CA+2) +n	↓ Record byte n*	11: line/column not specified by 83H.			

*All display characters must be formatted in ASCII code. A code greater than 0F0H will not be accepted by the HP 64000 program.

NA = Not Applicable.

See Appendix E for complete error code listing.

DISPLAY EXAMPLE PROGRAM

The Pascal program listing given in figure 11-3 is included as an example. It contains routines for display simulated I/O and keyboard simulated I/O.

The program establishes the display CA at address 100H and the keyboard CA at address 120H.

"680XX"

```
  { This program will display a sine wave (using Simulated I/O)}
  { on the screen.  Each time that you press the space bar,   }
  { which demonstrates the keyboard Simulated Input/Output,  }
  { the next harmonic will be added to the original wave, and }
  { the display will be updated.  The program will continue to }
  { run until you press either the 'simulate' softkey, or the  }
  { 'e' key.  As you continue to press the spacebar, you will }
  { see that the wave displayed comes closer and closer to a  }
  { squarewave.  NOTE: The equation to build a sawtooth wave  }
  { is also provided; to display the sawtooth, just uncomment }
  { the equation labeled 'sawtooth' and put the 'squarewave'  }
  { equation in comments.  HAVE FUN!                          }
}

PROGRAM DEMO;
$EXTENSIONS ON$

TYPE
  BITS= (BIT0,BIT1,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7);
  DISP_TYPE = ARRAY [0..31] OF BYTE;
  BUF_TYPE = ARRAY [0..255] OF BYTE;
  SOB      = SET OF BITS;

$GLOBVAR+$
  VAR
$ORG = 100H$
  DISP_CA : DISP_TYPE;
$ORG = 120H$
  KEY_CA : DISP_TYPE;
$END_ORG$
  T,S,F,X1,Y1 : BYTE;
  CO: BYTE;
  MORE_CHAR : BYTE;
  I,J,L,K : INTEGER;
  CH : CHAR;
  A,B,C,D : REAL;
$GLOBPROC ON$
```

Figure 11-3 Display/Keyboard Simulated I/O Program

Emulator/Analyzer 68000/68008
Display Simulated I/O

```
PROCEDURE DELAY;
  VAR B :INTEGER ;
BEGIN
  B := 0000FFFFH;
  WHILE B<>0 DO B := B - 1;
END;

PROCEDURE OPEN_DISPLAY;
{ This procedure opens the display by writing an 80H to the }
{ display control address (DISP_CA). }
BEGIN
  DISP_CA[0] := 80H;
  WHILE DISP_CA[0] = BYTE(80H) DO;
END;

PROCEDURE POSITION(X,Y:BYTE);
{ This procedure first checks to make sure that the next }
{ location is within valid display area. Then sets the row }
{ and column by writing 83H into DISP_CA +2 (DISP_CA[2]). }
BEGIN
  IF (X<BYTE(80)) AND (Y<BYTE(19)) THEN
  BEGIN
    DISP_CA[1] := Y;
    DISP_CA[2] := X;
    DISP_CA[0] := 83H;
    WHILE DISP_CA[0] = BYTE(83H) DO;
  END;
END;

PROCEDURE ROLL_DISPLAY;
{ This procedure first loads the important data such as: }
{ the line length and the ASCII codes to be placed on the }
{ line that is written. Then the code 82H is written into }
{ the control address to cause the 64000 to write the data }
{ to the display. }
BEGIN
  POSITION(BYTE(3),BYTE(18));
  DISP_CA[1] := 2;
  DISP_CA[2] := BYTE(90H);
  DISP_CA[3] := BYTE(90H);
  DISP_CA[0] := 82H;
  WHILE DISP_CA[0] = BYTE(82H) DO;
  DELAY;
END;
```

Figure 11-3 Display/Keyboard Simulated I/O Program (Cont'd)

```

PROCEDURE WRITE_CHAR(C:CHAR);
{ This procedure first checks to make sure that the ASCII code}
{ to be written to the screen is less than 0FH. The starting }
{ line number is written to the CA + 1 and the column number }
{ is written to the CA+2. Then the code 84H is written to the}
{ CA to cause the write to the display to be executed.      }
BEGIN
  IF C<>'
    BEGIN
      DISP_CA[1] := 1H;
      DISP_CA[2] := BYTE(C);
      DISP_CA[0] := 84H;
      WHILE DISP_CA[0] <> BYTE(0) DO;
    END
  ELSE ROLL_DISPLAY;
END;

PROCEDURE CLOSE_DISPLAY;
{ This procedure closes the display by writing the control code }
{ 81H to the CA.                                           }
BEGIN
  DISP_CA[0] := 81H;
  WHILE DISP_CA[0] = BYTE(81H) DO;
END;

PROCEDURE OPEN_KEYBOARD;
{ This procedure opens the keyboard for simulated I/O. This }
{ is accomplished by writing the corresponding control code }
{ (80H) the control address. Note that the control address }
{ for the keyboard MUST be different than the CA for the }
{ display.                                               }
BEGIN
  KEY_CA[1] := BYTE(-2);
  KEY_CA[2] := 1H;
  KEY_CA[0] := 80H;
END;

FUNCTION GET_CHAR:CHAR;
{ This procedure first goes to the OPEN_KEYBOARD routine to }
{ open the keyboard for SIM. I/O. Then the value of the key}
{ pressed is retrieved. The procedure then checks for the }
{ value of the key.                                       }
BEGIN
  OPEN_KEYBOARD;
  WHILE KEY_CA[0] <> BYTE(0) DO;
  GET_CHAR := CHAR(KEY_CA[4]);
  IF KEY_CA[1]=BYTE(13) THEN GET_CHAR:='
END;

```

Figure 11-3 Display/ Keyboard Simulated I/O Program (Cont'd)

Emulator/Analyzer 68000/68008
Display Simulated I/O

```
BEGIN
{ This is the main procedure of the program.
K := 0;
F := 1;
S := 1;
T := S;
WHILE F = BYTE(1) DO
  BEGIN
    IF S=1 THEN OPEN_DISPLAY;
    K := K+1;
    FOR I := 4 TO 75 DO
      BEGIN
        A := 0.0;
        B := -1.0;
        FOR L := 1 TO K DO
          BEGIN
            B := -B;
            { square wave }
            A := A + B*COS(REAL(2*L-1)*6.28*REAL(I-4)/56.0)/REAL(2*L-1);
            { sawtooth wave }
            { A := A + SIN(REAL(L)*6.28*REAL(I-4)/45.0)/(1.4*REAL(L)); }
          END;
            X1 := BYTE(I);
            Y1 := BYTE(5.0*A+9.0);
            IF S=1 THEN POSITION(X1,Y1);
            IF S=1 THEN WRITE_CHAR('.');
          END;
        IF S= 1 THEN
          BEGIN
            CH := GET_CHAR;
            IF CH<>CHAR(0) THEN IF CH='E' THEN F:=0;
            IF CH<>CHAR(0) THEN IF CH='S' THEN T:=0;
            CLOSE_DISPLAY;
          END;
        S := T;
      END;
    END.
  END.
```

Figure 11-3 Display/ Keyboard Simulated I/O Program

Chapter 12

KEYBOARD SIMULATED I/O

OVERVIEW

This chapter will:

- Describe keyboard simulated I/O.
- Explain how to establish a keyboard interface.
- List the keyboard control codes.

NOTE

A sample Pascal program that uses keyboard simulated I/O is provided in chapter 11, Display Simulated I/O.

The keyboard simulated I/O feature allows the emulator to control the keyboard in the development station. You can open the keyboard interface, accept keystrokes from the keyboard, and close the keyboard interface.

To do this, you must write a program that passes keyboard control codes to, and accepts keystrokes from, the development station. You assemble or compile the program and link it with your other programs in the emulation environment. The following paragraphs describe how to establish the keyboard interface with your program.

The keyboard I/O interface (see figure 12-1) requires a memory location, called the keyboard control address (CA), to which I/O handshaking codes are sent by your program and the HP 64000 system. The HP 64000 samples the keyboard CA periodically looking for commands. The keyboard CA must be initially defined in your program and in the simulated I/O configuration. When more than one simulated I/O interface is to be implemented, you must make sure that each I/O program assigns a unique address for the CA.

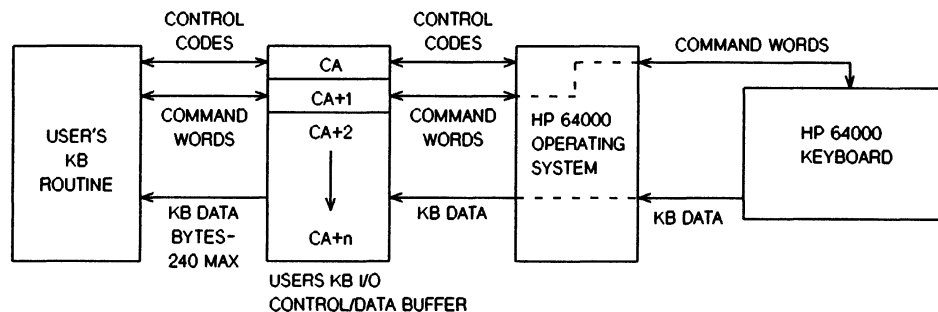


Figure 12-1. Keyboard Interface Diagram

The address for the keyboard CA is entered during simulated I/O configuration. The emulator must not be restricted to real time runs when using the keyboard simulated I/O. The keyboard CA location must be located in memory space assigned as user RAM or emulation RAM. It is recommended that the keyboard CA location be in emulation RAM since this allows your programs to run faster.

Two keyboard I/O codes include supplemental information. This supplemental information is contained in the locations following the keyboard CA, i.e., CA+1 thru CA+n. The supplemental information must be placed in locations CA+1 thru CA+n BEFORE the control code is placed in the keyboard CA. When this is not done, the HP 64000 may respond to the control code in the keyboard CA before the supplemental data is set into locations CA+1 thru CA+n.

The keyboard control codes are summarized in table 12-1.

OPENING THE KEYBOARD FILE (80H)

Before any keyboard operation can be initiated, your program must request that the keyboard (KB) I/O interface be opened (see figure 12-2, phase 1).

NOTE

During the time the simulated I/O keyboard file is open, the development station does not respond to the keyboard in the standard manner.

To close the simulated I/O keyboard file and return the keyboard to standard operation, press the *simulate* softkey. If the display file is also open, it, too, is closed when the softkey is pressed.

The first step in opening the keyboard interface is to place a -1 or -2 into location CA+1. The -1 input command word causes the current line not to be cleared on the first character (i.e., the current keyboard characters are appended to any characters already displayed on the same line). A "-2" causes the current line to be cleared on the first character (i.e., previously displayed characters are erased from the line and only the current keyboard characters are displayed).

Next, your program must place the record length into location CA+2. This is the maximum record length (i.e., number of keyboard characters) that your program will accept from the keyboard. The record length may be up to 240 characters (3 lines on the HP 64000 display). However, the keyboard may transmit more or less characters than this specification. When the number of characters transmitted exceeds the record length, your program is informed by a KB-output-command word in location CA+1 (see table 12-2).

READ IN PROCESS (82H) - HP 64000 RESPONSE

The HP 64000 responds to the open keyboard file request by storing the input command word and record length, and by placing code 82H into the keyboard CA (see figure 12-2, phase 2).

The HP 64000 sets the output command word, in CA+1, to the same code specified in the KB-input-command word (-1 or -2). The HP 64000 then monitors the keyboard until an output command word is detected. The result of this detection is described in the following paragraphs.

OUTPUT AVAILABLE (00)-HP 64000 RESPONSE

When an output command word is detected, the HP 64000 places the word, and if applicable, other data into locations CA+1 through CA+n (see figure 12-2, phase 3). The output word, which is always sent, is placed in buffer location CA+1.

The HP 64000 places a 00 in the keyboard CA to indicate that either a command word and/or data is available.

When keyboard characters are sent and if a "lost character" was generated then the lost character is placed into location CA+2. Also, when keyboard characters are sent, the actual number of characters in the string (i.e., actual record length) is placed into location CA+3. The keyboard characters (ASCII coded bytes) are placed into locations CA+4 thru (CA+4)+n.

The output command word in location CA+1 may be any one of the codes shown in table 12-2. Two of these codes, 8 and 24, occur only when the record length from the keyboard exceeds the record length specification. When either of these codes is generated, location CA+2 contains the ASCII code of the surplus or lost character. A lost character may be generated in two ways:

- a. When characters are entered as a continuous string and the string exceeds the specified record length - For this case, the first character to exceed the specified record length is placed in "lost character" location CA+2. When typing continues, each individual surplus character is placed into the "lost character" location CA+2 replacing the previous character. Thus, the last "lost character" entered remains in location CA+2.
- b. When a character is inserted into a full record - For this case, the character at the end of the already full record is placed into "lost character" location CA+2. If additional characters are inserted, each succeeding end character is placed into CA+2, replacing the previous character.

CLOSING THE KEYBOARD FILE (81H)

Your program closes the keyboard file by placing code 81H into the keyboard CA. The HP 64000 responds by closing the file and returning code 00 to the keyboard CA.

When the close file is not accepted, an error code is returned to the keyboard CA as shown in table 12-1.

Pressing the simulate softkey or performing a reset-closes closes the open keyboard file. Closing the open keyboard file also closes an open display file.

Table 12-1. Keyboard Control Codes

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
			VALID USER REQUEST	INVALID REQUEST	
	Address	Contents	Address	Contents	Error Code
OPEN KB INTER FACE	CA	80H		SEE 82H, below	08, 12, or 14
	CA+1	KB Input Command Word			Other codes do not apply
	CA+2	Max. Record Length Specification (up to 240 bytes)			
READ IN PROCESS		Initiated by HP 64000 program in response to 80H above	CA HP 64000 stores KB-input-command word & max. record length spec. It then monitors KB-output-command word until positive word is detected and then responds as follows:	82H	
OUTPUT AVAILABLE		Initiated by HP 64000 after 82H, above	CA CA+1	00 KB out-put command word	

Table 12-1. Keyboard Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
	Address	Contents	VALID USER REQUEST		INVALID REQUEST
			Address	Contents	Error Code
CLOSE KB I/O	CA	User program may then respond to 00 with 80H or 81H as shown below.	CA+2	Reserved for Lost Character	08 or 14 Other codes do not apply.
			CA+3	Actual record length (#of KB bytes)	
			CA+4	KB Byte 0	
			↓	↓	
			(CA+4)+n	KB Byte n	
			CA	00	

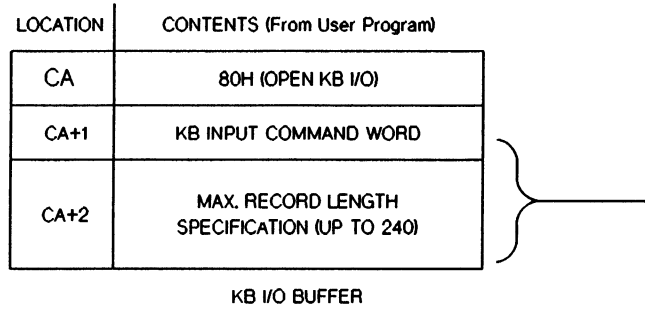
See Appendix E for complete error code listing.

Table 12-2. Command Word Codes

Part A.		KB - Input - Command Word
Code		Meaning
-1		Current line not cleared. Characters appended to previously displayed characters.
-2		Current line cleared. Previously displayed characters erased.

Part B.		KB - Output - Command Word
Code		Meaning
8		Insert character in full line (lost character placed in CA+2)
9		Tab Key
10		Down arrow key
11		Up arrow key
12		Display next page
13		Carriage return
14		Attempting to move cursor right past last allowed screen location
15		Attempting to move cursor left past first allowed screen location
16		Delete character from full line
17		Shift key
18		Display previous page
19		Roll display down
20		Roll display up
21		Shift right arrow key
22		Shift left arrow key
23		Clear line key
24		Actual record length exceeded record length specification (lost character placed in CA+2)

Phase I - User Requests Interface Opening



The actual address for location "CA" is defined by the user during configuration of the emulation "CMDFILE".

Phase II - HP 64000 Response to Open-Interface Request

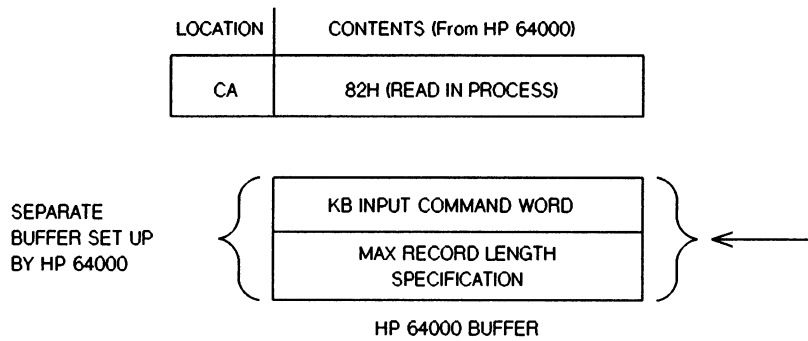
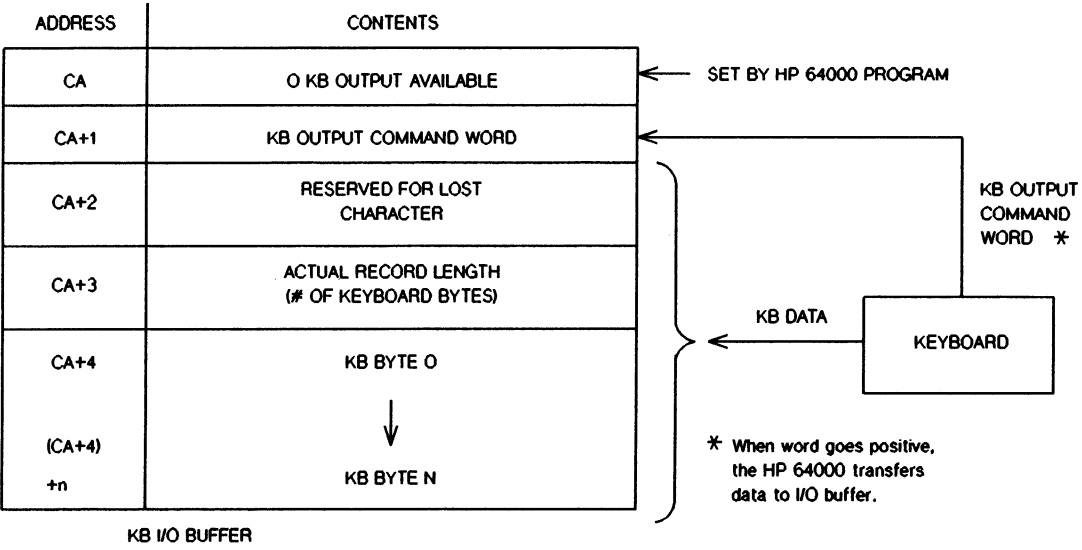


Figure 12-2. Keyboard Interface Sequence

Phase III - HP 64000 Detects Positive KB Output Command Code



Phase IV - The user program may respond with either an 80H code as shown for phase I or an 81H code which closes the simulated keyboard I/O interface.

Figure 12-2. Keyboard Interface Sequence (Cont'd)

Chapter 13

DISC FILE SIMULATED I/O

OVERVIEW

This chapter will:

- Describe disc file simulated I/O.
- Explain how to establish a disc interface.
- List the disc control codes.
- Provide a sample Pascal disc I/O program.

The disc file simulated I/O feature allows the emulator to control up to six files on a disc drive. You can create disc files, open and close files, delete files, and rename files. You can position to record one, write a record, advance n records, backup n records, position to record n, and read a record within a file.

To do this, you must write a program that passes disc control codes and records to the development station. You assemble or compile the program and link it with your other programs in the emulation environment. The following paragraphs describe how to establish the disc drive interface with your program.

The disc I/O interface (see figure 13-1) requires memory locations, called the disc file control addresses (CA), to which I/O handshaking codes are sent by your program and the HP 64000 system. One CA is required for each file you access. The maximum number of files that can be opened simultaneously is six. The HP 64000 samples each disc file CA periodically looking for commands. The disc file CA must be initially defined in your program and in the simulated I/O configuration. When more than one simulated I/O interface is to be implemented, you must make sure that each I/O program assigns a unique address for the CA.

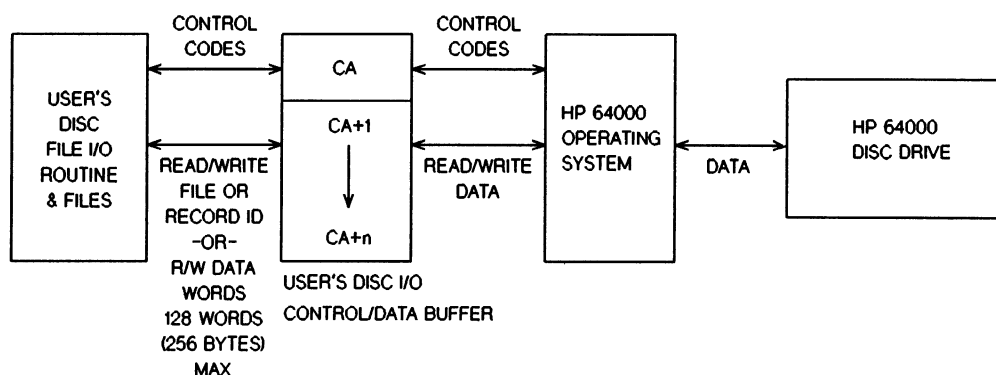
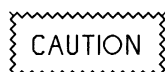


Figure 13-1. Disc Drive Interface Diagram

The address for each disc file CA is entered during simulated I/O configuration. The emulator must not be restricted to real time runs when using the disc drive simulated I/O. The disc file CA location must be located in memory space assigned as either user RAM or emulation RAM. It is recommended that the disc file CA locations be in emulation RAM since this allows your programs to run faster.

Most disc file I/O codes include supplemental information. This supplemental information is contained in the locations following the disc file CA, i.e., CA+1 thru CA+n. The supplemental information must be placed in locations CA+1 thru CA+n BEFORE the control code is placed in the disc file CA. When this is not done, the HP 64000 may respond to the control code in the disc file CA before the supplemental data is set into locations CA+1 thru CA+n.



The disc file simulated I/O can be used to access critical system files. Use extreme care when accessing disc files. It is recommended that you do not access the following file types:

- Emulation Command Files (Type 6)
- Linker Command Files (Type 7)
- Linker Configuration Files (Type 8)

Incorrectly accessing these file types may destroy them and cause serious system problems! It is recommended that you use the data (type 10) file only.

Detailed file descriptions and record layouts are contained in the file format manual. A partial list of files types is included in table 13-1.

Table 13-1. Disc File Type Numbers and Names

FILE TYPE NUMBER	FILE NAME
2	Source
3	Relocatable
4	Absolute
5	Listing
6	Emulation Command
7	Linker Command
8	Trace
10	Data
12	Assembler Symbols
13	Linker Symbol
*14 thru 255	Types are defined and numbers assigned by user program.

* HP may required some unassigned numbers for future use. It is, therefore, strongly recommended that the DATA (type 10) file be employed for the user defined type file.

Predefined types of files, identified by numbers 2 thru 13, may be used by your programs.

You may assign file type numbers 14 thru 255 to files used by your programs. It should be noted, however, that HP may require some unassigned numbers for future use. It is recommended that you use file type numbers well beyond number 14.

NOTE

Once created, file types 14 thru 255 can only be deleted by using the simulated I/O delete command.

The complete file name is assigned during emulation configuration. For a given file name, only one of each file type may be created. For example, a file named USA may only have one type 3 file; it cannot have two type 3 files.

CREATING A NEW FILE (80H)

To create a new file, your program places the file type number into location CA+1, the disc number into location CA+2, and then places code 80H into the disc file CA. The disc number is the disc upon which the file will reside.

The HP 64000 responds by creating the file type requested and returning a 00 to the disc file CA.

When the file cannot be created, an error code as shown in table 13-2 is returned to the disc file CA. General definitions for the error codes are listed in appendix E.

After the file is created, your program may either write records immediately into it, or close it, and then reopen it and write records into it later.

WRITING A RECORD (89H)

A new record may be written into a file in one of two ways. The record may be added to the end of the file or it may be written over an existing record in the file. However, when an existing record is written over, then the newly written record becomes the last record in the file.

To add a record to the end of the file, the record selected must be one greater than the last record in the file. For example, if a file contains five records, then record 6 must be selected before writing is initiated. (If record 5 is selected, it will be written over by the new record.) After writing record 6, record 7 may be written by issuing another write code, etc.

To write over an existing record, first select the record and then initiate writing. Again, remember that all following records in the file are erased. For example, if a file contains 10 records, and record three is written over, then records four thru ten are erased.

Writing First Record

After a file is created the first record is written into it as follows. Your program places parameters, as described below, into locations CA+1 thru CA+n, and then places code 89H into the disc file CA.

The number of words in the record is placed into location CA+1. A write record may contain up to a maximum of 128 words (256 bytes). Thus, an even number of bytes (whole words) must always be written.

Locations CA+2 thru (CA+2)+n contain the words of the write record.

The HP 64000 responds by writing the record into the file as record number 1. After the record is successfully written, the HP 64000 returns a 00 to the disc file CA. When the record cannot be written, an error code, as listed in table 13-2, is returned to the disc file CA.

Writing Additional Records

When a newly created file is still open (i.e., has never been closed), additional records are written into the file as described for record one with the following difference. Each succeeding record is automatically written with the next corresponding record number. Thus, the second record written becomes record number 2, the third record written becomes record number 3, etc.

CLOSING THE FILE (82H)

To close a file, your program places code 82H into the disc file CA. The HP 64000 responds by closing the file and returning a 00 to the disc file CA. When the file cannot be closed, an error code, as listed in table 13-2, is returned to the disc file CA.

ACCESSING EXISTING FILES - OPENING THE FILE (81H)

To open an existing file, your program places the file type number into location CA+1, the disc number into location CA+2, and then places code 81H into the disc file CA.

The HP 64000 responds by opening the file and returning a 00 to the disc file CA. When the file cannot be opened, an error code, as shown in table 13-2, is returned to the disc file CA.



When a record is written into a disc file, it always becomes the last record in the file. Thus, writing a record into any location other than at the end of the file erases all the following records in the file. When accomplishing the following paragraph, choose record positions with care!

After the file is opened, your program may either: (1) immediately read/write record 1, (2) select any record for reading, or (3) select a position within the file to begin writing.

SELECTING A RECORD

Records are selected automatically upon entering and using a file, advancing N records, backing up N records, positioning to record N, and rewinding to record 1. These options are discussed in the following paragraphs.

Automatic Selection Of Record 1,2,3...Etc.

When an existing file is first accessed, record 1 is automatically selected. Thus, it may be immediately written into, or read from, without first selecting it with an "advance", "position", or "rewind" code. After reading or writing record 1, record 2 is automatically selected and may be read from or written into. This process may be continued for records 3, 4, 5, ..., etc.

Advancing N Records (84H)

Records located ahead of the currently selected record (i.e., those records with higher numbers) may be selected as follows. Your program places the number of records into locations CA+1 and CA+2, and then places code 84H into the disc file CA. The number of records is selected with a 15-bit word. The eight least significant bits are located in CA+1. The seven most significant bits are located in CA+2. The most significant bit in CA+2 is not used.

The HP 64000 responds by advancing the specified number and returning a 00 to the disc file CA. When the record cannot be selected, an error code, as shown in table 13-2, is returned to the disc file CA.

After the record is selected, your program may then either read from or write into it.

Backup N Records (85H)

Records located behind the currently selected record (i.e., those records with smaller numbers than the current record) are selected in a way very similar to "advance "N" records." The only difference is that backup code 85H is placed into the disc file CA. Location CA+1 and CA+2 contain the number of records as defined above in Advancing N Records (84H). The HP 64000 also responds as described above.

Position To Record N (86H)

Any record within the file may also be selected without knowing its location relative to the current record. This method is also similar to the "advance" or "backup" methods. The difference is that position code 86H is placed into the disc file CA. Locations CA+1 and CA+2 contain the record number as defined above in Backup N Records (85H). The HP 64000 responds as described above.

Rewind To Record One (88H)

This is a fast way to select record 1. Only record 1 can be selected using this method. Your program places code 88H into the disc file CA, there are no entries required in locations CA+1 and CA+2. The HP 64000 program responds by positioning at record one and returning 00 to the disc CA. When the record cannot be selected, an error code, as shown in table 13-2, is returned to the disc CA.

READING THE RECORD (87H)

Once a record has been selected, it may be read as follows. Your program places the maximum number of 16-bit words it will accept from the record into location CA+1. Up to 128 words may be accepted. The recommended technique is always set CA+1 to 128. Then, after reading is complete, throw away those words not wanted, if any. After specifying location CA+1, code 87H is placed into the disc file CA.

When the record is read successfully, the HP 64000 responds as follows: code 00 is returned to the disc file CA. The actual number of 16-bit words read from the buffer is placed in location CA+1. Location CA+2 thru (CA+2)+n contains bytes 0 thru n.

When the record cannot be read, an error code, as shown in table 13-2, is returned to the disc file CA.

CHANGING THE FILE NAME (8AH)

The file name associated with a given disc file CA may be changed. This does not rename any files on the disc, but simply changes the name in the emulation command file associated with a given disc file CA. To do this you must first make sure that the present file associated with the disc file CA is closed.

To change the file name in the emulation configuration file, your program places the new name record into locations CA+1 thru CA+16, and then places code 8AH into the disc file CA. The name record is a fixed length record consisting of eight, 16-bit words. This record contains the record name, USERID, and specifies the length of both of these items.

The name must contain at least one character and may be up to nine characters long. The ID may be up to six characters long. However, the name and ID lengths are specified in a unique way. Also, the words containing these characters must be packed in the name record. Name, character lengths and packing are described in the File Format Manual and in table 13-2.

To actually change the name of an existing file, you must copy the contents of the file under the old file name into the file with the new file name. Either one or both of these file names may be specified by your program at run time and accessed after "change file name" has been issued to the appropriate disc file CA.

DELETING THE FILE (83H)

To delete a file, your program places the file type into location CA+1, the disc number into location CA+2, and then places code 83H into the disc file CA. The HP 64000 responds by deleting the file. When the file cannot be deleted, an error code is returned to the disc file CA as shown in table 13-2. This delete is similar to a "purge" command in the general operating system. The purged file does go into the recoverable file list.

Table 13-2. Disc File Control Codes

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:			
	Address	Contents	VALID USER REQUEST		INVALID REQUEST	
			Address	Contents	Error Code	
CREATE FILE	CA	80H	CA	00	01 thru 08, 10	
	CA+1	File Type Number				09: File is not open
	CA+2	Disc #				11 thru 14: NA
OPEN FILE	CA	81H	CA	00	01 thru 08, 10	
	CA+1	File Type Number				09: File is already open
	CA+2	Disc #				11 thru 14: NA
CLOSE FILE	CA	82H	CA	00	01 thru 08	
						09: File is already closed
						10 thru 14: NA
DELETE FILE	CA	83H	CA	00	01 thru 08,10	
	CA+1	File Type Number				09: File is not open
	CA+2	Disc #				11 thru 14:NA

Table 13-2. Disc File Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
	Address	Contents	VALID USER REQUEST		INVALID REQUEST
			Address	Contents	Error Code
ADVANCE "N" RECORDS	CA	84H	CA	00	01 thru 08
	CA+1	LSB 15-bit* record			09: File not open
	CA+2	MSB number (*bit 16 not used)			10 thru 14: NA
BACKUP "N" RECORDS	CA	85H	CA	00	01 thru 08
	CA+1	LSB 15-bit* record			09: File not open.
	CA+2	MSB number (*bit 16 not used)			10 thru 14: NA
POSITION TO RECORD "N"	CA	86H	CA	00	01 thru 08
	CA+1	LSB 15-bit* record			09: File not open
	CA+2	MSB number (*bit 16 not used)			10 thru 14: NA
READ RECORD	CA	87H	CA	00	01 thru 08
	CA+1	Max. number of words user can accept. (128 words/ 256 bytes max.)	CA+1	Actual # of words read from buffer.	09: File is not open 12

Table 13-2. Disc File Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
	Address	Contents	VALID USER REQUEST		INVALID REQUEST
			Address	Contents	Error Code
REWIND TO RECORD ONE	CA	88H	CA+2 ↓ (CA+2)+n (*256 bytes/128 words is max. record length.)	Read Byte 1 ↓ Read Byte n *	10, 11, 13, 14: NA
			CA	00	01 thru 08 09: File is not open 10 thru 14: NA
WRITE RECORD	CA	89H	CA	00	01 thru 08, 12
	CA+1	Number of words to be written. (128 words/256 bytes maximum.)			09: File is not open.
	CA+2 ↓ (CA+2)+n	Write byte 1 ↓ Write byte n			10, 11, 13, 14: NA

Table 13-2. Disc File Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
			VALID USER REQUEST		INVALID REQUEST
	Address	Contents	Address	Contents	Error Code
CHANGE FILE NAME SEE NOTE BELOW	CA	8AH	CA	00	01 thru 08 12 & 15 09: File not open 10, 11, 13, 14: NA
	CA+2	Bits 7-5 specify length of file name in 16-bit words-1. Bits 4 & 3 specify ID length in 16-bit words. Bits 2-0 contain all zeros. (See note below.)			
	CA+3	First character of file name. Limited to capital letters A thru Z.			
		Second and following file name characters may be small or capital letters.			

Table 13-2. Disc File Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
	Address	Contents	VALID USER REQUEST	INVALID REQUEST	Error Code
		numerals 0 thru 9, underlines, and only if required one blank may be used to fill in last character in last word of name.			
	CA+4 thru CA+n. Where n 10	Up to 9 name characters may be used.			
	CA+ (n+1)	First USERID character.			
	CA+ (n+2) ↓ thru ↓ CA+16	Up to 6 USERID characters may be used. See note below.			

Table 13-2. Disc File Control Codes (Cont'd)

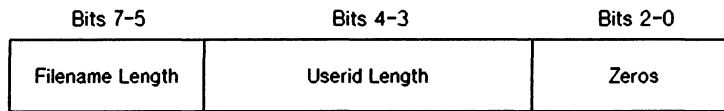
NOTE:

Bits 7-5 of address location (CA+1) are used to specify the length of the new filename. The table below gives the correct values for all filename lengths. The filename characters, follow CA+1 in succeeding bytes. If a filename contains an even number of characters, a terminating blank must be used. See the example below.

The userid length is specified as the number of words required in bits 4-3 of (CA+1). The userid must immediately follow the filename words in memory. For userids with an odd number of characters, the last word must also be terminated with a blank (see table below).

Bits 2-0 of address (CA+1) are set to 0.

Once the filename and address CA+1 have been set up in memory, the value 8AH should be written to the Control Address (CA).



CA+1

Bits 7-5	Filename Length (In Characters)
000	1
001	2 or 3
010	4 or 5
011	6 or 7
100	8 or 9

Bits 4-3	Userid Length (In Characters)
00	0
01	1 or 2
10	3 or 4
11	5 or 6

Example: Sim I/O Disc file #1 with Control Address (CA) 4000H will be given the new name of FILE:UID.

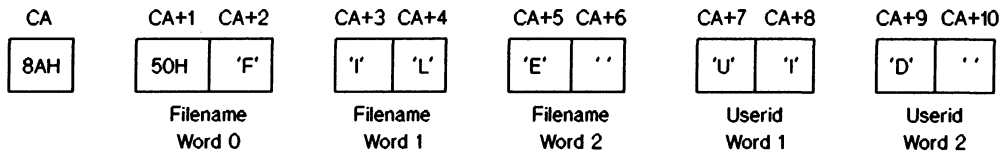


Table 13-2. Disc File Control Codes (Cont'd)

CA+1:
;--; Filename Length (bits 7-5)
 ;-; Userid Length (bits 4-3)
 ;-; Zeros (bits 2-0)

50H = 010 10 000B
 Binary 2 for userid words
 Binary 2 for 2 ADDITIONAL filename words

The name and USERID characters must be packed into a fixed length record. This record consists of 8, 16-bit words. Thus, the name record will always require a user buffer consisting of 17 bytes (byte CA through byte CA+16). All unused 16-bit words must be at the end of the record. No intervening unused words or bytes are allowed.

DISC FILE EXAMPLE PROGRAM

The Pascal program listing given in figure 13-2 is included as an example. It contains a routine that writes to a single disc file.

The program establishes the disc CA at address 1000H. The file is a type 10 data file located on disc 0.

Emulator/Analyzer 68000/68008
Disc File Simulated I/O

"680XX"

PROGRAM DISC_1;

{The purpose of this program is to demonstrate how to use disc sim I/O. The program takes a block of data (presumably generated by the target system) and creates a file named DATA:data so that it can be used later (for example, a HOST PASCAL program might analyze it to see if the data is correct).

A record in the file DATA:data will be made 122 bytes long, consisting of data taken consecutively beginning at address ACCUMULATED_DATA. The data will be put in the file in the same order, all in one record. Because the disc file is organized in words, an even number of bytes must be written to the file.

This program:

- 1) Opens the file. If an error occurs, the error code is placed at the variable ERROR, and the procedure OPEN_ERROR is executed.
- 2) Transfers the data from ACCUMULATED_DATA to the disc Control Address area.
- 3) Writes the file. If an error occurs, the error code is placed at the variable ERROR, and the procedure WRITE_ERROR is executed.
- 4) Closes the file.

Note that the name of the file is defined in the emulation command file (during emulation configuration).}

\$EXTENSIONS ON\$ {Extensions to PASCAL will be used in this program.}

TYPE

DISC_CA_TYPE = ARRAY [0..257] OF BYTE;

{Since 68010 data is organized in bytes, the type BYTE is appropriate for most variables.}

VAR

\$GLOBVAR ON\$

ERROR : BYTE; {This will contain the error code information.}

\$ORG = 1000H\$

DISC_CA : DISC_CA_TYPE;

\$END_ORG\$

{Org'd to keep control address at the same location whenever the program is compiled, even though program length may change due to program revision.}

Figure 13-2. Program Disc_1

```
ACCUMULATED_DATA : ARRAY [1..122] OF BYTE;

$GLOBPROC ON$ {So routines can easily be viewed in emulation.}

PROCEDURE WAIT_FOR_DISC;

    {This procedure waits until DISC_CA[0] does not have a 1 in bit 7, which means the last
    disc access request was acted on by the HP 64000.}

BEGIN

    REPEAT
        UNTIL (DISC_CA[0] = 0); {Wait until bit 7 is zero.}

END; {Procedure wait_for_disc.}

PROCEDURE OPEN_ERROR;

    {This procedure places the error code returned by the open request in the variable "ERROR"
    and waits for operator action. The nature of the error, either open error or write error,
    can be discerned from the address at which the repeat loop is acting.}

BEGIN

    ERROR := DISC_CA[0];
    REPEAT
        UNTIL (1 = 0);

END; {Procedure open_error.}

PROCEDURE WRITE_ERROR;

    {This procedure places the error code returned by the write request in the variable "ERROR"
    and waits for operator action. The nature of the error, either write error or open error,
    can be discerned from the address at which the repeat loop is acting.}

BEGIN

    ERROR := DISC_CA[0];
    REPEAT
        UNTIL (1 = 0);

END; {Procedure write_error.}
```

Figure 13-2. Program Disc_1 (Cont'd)

Emulator/Analyzer 68000/68008
Disc File Simulated I/O

```
PROCEDURE TRANSFER_DATA;

    {This procedure transfers 122 bytes of data from the array ACCUMULATED_DATA to DISC_CA
    starting at DISC_CA[2].}

VAR
    INDEX : BYTE;

BEGIN

    FOR INDEX := 1 TO 122 DO
        DISC_CA[ 1 + INDEX ] := ACCUMULATED_DATA[INDEX ];
    END;    {Procedure transfer_data.}

BEGIN    {Main program: to begin here, insert the command, "run from DISC_1".}

    DISC_CA[1] := 10;    {Type is data}
    DISC_CA[2] := 0;    {on disc 0.*}
    DISC_CA[0] := 80H;  {Tell HP 64000 that use of disc sim I/O is requested.}

    WAIT_FOR_DISC;

    IF (DISC_CA[0] <> 0) THEN
        OPEN_ERROR;    {If there was an error, notify the operator.}

    TRANSFER_DATA;

    DISC_CA[1] := 61;    {Place number of WORDS to write here.}
    DISC_CA[0] := 89H;  {Request write to disc.}

    WAIT_FOR_DISC;

    IF (DISC_CA[0] <> 0) THEN
        WRITE_ERROR;

    DISC_CA[0] := 82H;    {Close disc file.}

END.    {Program Disc_1.}
```

Figure 13-2. Program Disc_1 (Cont'd)

Chapter 14

RS-232 SIMULATED I/O

OVERVIEW

This chapter will:

- Describe RS-232 simulated I/O.
- Explain how to establish an RS-232 interface.
- List the RS-232 control codes.

In this chapter, whenever 68000 is mentioned (even as part of a filename) 68008 is also assumed unless there are differences between the two processors. In that case, the differences will be noted.

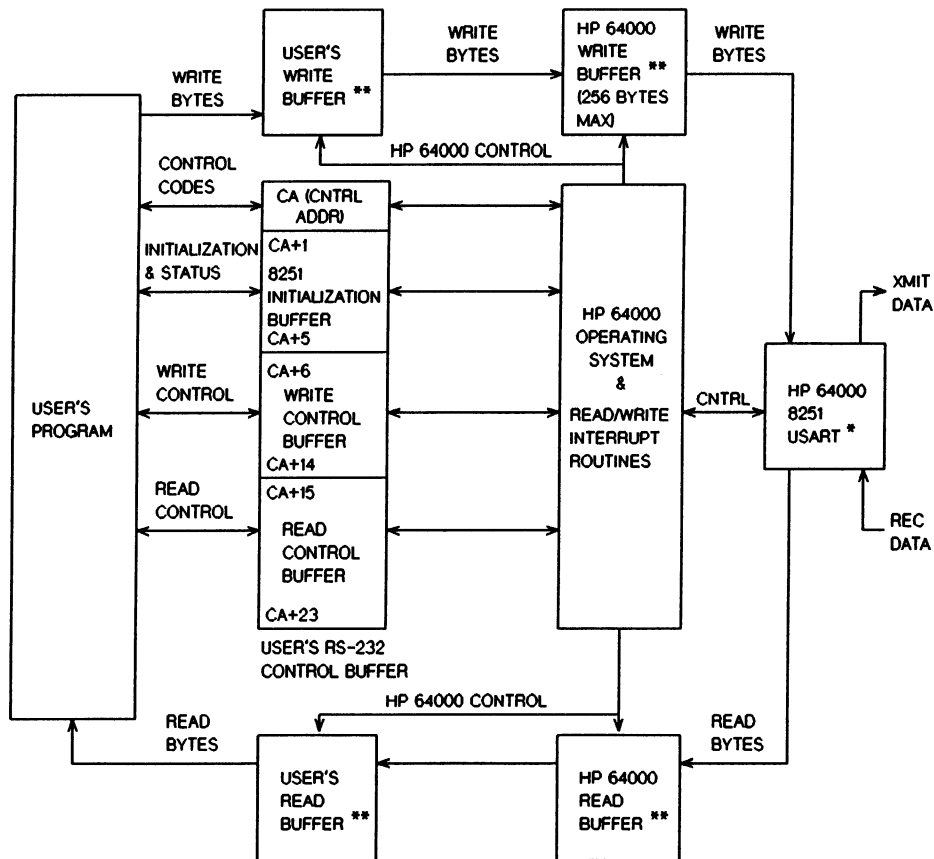
The RS-232 simulated I/O feature allows the emulator to control the RS-232 port in the development station. You can open and close the RS-232 file; open, close, and update the write buffer; open, close, and update the read buffer; and control the universal synchronous/asynchronous receiver/transmitter (8251 USART).

To do this, you must write a program that passes RS-232 control codes and bytes to the development station. You assemble or compile the program and link it with your other programs in the emulation environment. The following paragraphs describe how to establish the RS-232 interface with your program.

The RS-232 I/O interface (see figure 14-1) requires a memory location, called the RS-232 control address (CA), to which I/O handshaking codes are sent by your program and the HP 64000 system. The HP 64000 samples the RS-232 CA periodically looking for commands. The RS-232 CA must be initially defined in your program and in the simulated I/O configuration. When more than one simulated I/O interface is to be implemented, then you must make sure that each I/O program assigns a unique address for the CA.

The address for the RS-232 CA is entered during simulated I/O configuration. The emulator must not be restricted to real time runs when using the RS-232 simulated I/O. The RS-232 CA location must be located in memory space assigned as user RAM or emulation RAM. It is recommended that the RS-232 CA location be in emulation RAM since this allows your programs to run faster.

Most RS-232 I/O codes include supplemental information. This supplemental information is contained in the locations following RS-232 CA, i.e., CA+1 thru CA+n. The supplemental information must be placed in locations CA+1 thru CA+n BEFORE the control code is placed in the RS-232 CA. When this is not done, the HP 64000 may respond to the control code in the RS-232 CA before the supplemental data is set into locations CA+1 thru CA+n.



* USART = Universal Synchronous/Asynchronous Receiver/Transmitter.
** Buffers are required only if records are to be read or written. Single bytes do not require these buffers.

Figure 14-1. RS-232 Interface Diagram

OPENING THE RS-232 FILE (80H)

Before any RS-232 operations can be initiated, your program must request that the RS-232 File be opened. This is done by placing code 80H into the RS-232 CA.

The HP 64000 responds by opening the RS-232 file and returning a 00 to the RS-232 CA. When the file cannot be opened, error code 08 or 09 is returned to the RS-232 CA.

After the file is opened, the 8251 must be initialized as described in the next paragraph.

INITIALIZING THE 8251 USART (82H)

In general, 8251 USART initialization consists of resetting the 8251 USART and then selecting one of the following three operating modes: (1) asynchronous, (2) synchronous with one sync character, or (3) synchronous with two sync characters. See figure 14-2 below.

Your program requests initialization by first setting up buffer locations CA+1 thru CA+5 and then placing code 82H into the RS-232 CA.

After the 8251 is initialized, the HP 64000 returns a 00 to the RS-232 CA. When the 8251 USART cannot be initialized, error code 08 or 09 is returned as shown in table 14-1.

ADDRESS	ASYNCHRONOUS MODE - INITIALIZATION FORMAT	SYNCHRONOUS MODE- SINGLES SYNC CHARACTER INITIALIZATION FORMAT	SYNCHRONOUS MODE- DOUBLE SYNC CHARACTER INITIALIZATION FORMAT	ADDRESS
CA	82H - INITIALIZE 8251	82H - INITIALIZE 8251	82H - INITIALIZE 8251	CA
CA+1	COMMAND INSTRUCTION (Internal Reset 8251)	COMMAND INSTRUCTION (Internal Reset 8251)	COMMAND INSTRUCTION (Internal Reset 8251)	CA+1
CA+2	ASYNCHRONOUS MODE INSTRUCTION	SYNCHRONOUS MODE INSTRUCTION	SYNCHRONOUS MODE INSTRUCTION	CA+2
CA+3	SYNC OPTION WORD 0=No sync characters	SYNC OPTION WORD 1=1 sync character	SYNC OPTION WORD 2=2 sync characters	CA+3
CA+4	Not Used	SYNC CHARACTER 1	SYNC CHARACTER 1	CA+4
CA+5	Not Used	Not Used	SYNC CHARACTER 2	CA+5
CA+6 ↓ CA+22	RESERVED FOR WRITE CONTROL	RESERVED FOR WRITE CONTROL	RESERVED FOR WRITE CONTROL	CA+6 ↓ CA+22
CA+23 ↓ CA+39	RESERVED FOR READ CONTROL	RESERVED FOR READ CONTROL	RESERVED FOR READ CONTROL	CA+23 ↓ CA+39

Figure 14-2. 8251 Initialization Formats

A command instruction with Internal Reset (IR) bit D6 set is placed into location CA+1. See figure 14-3 below. The contents placed into locations CA+2 thru CA+5 depend upon the operating mode selected as described in the following paragraphs.

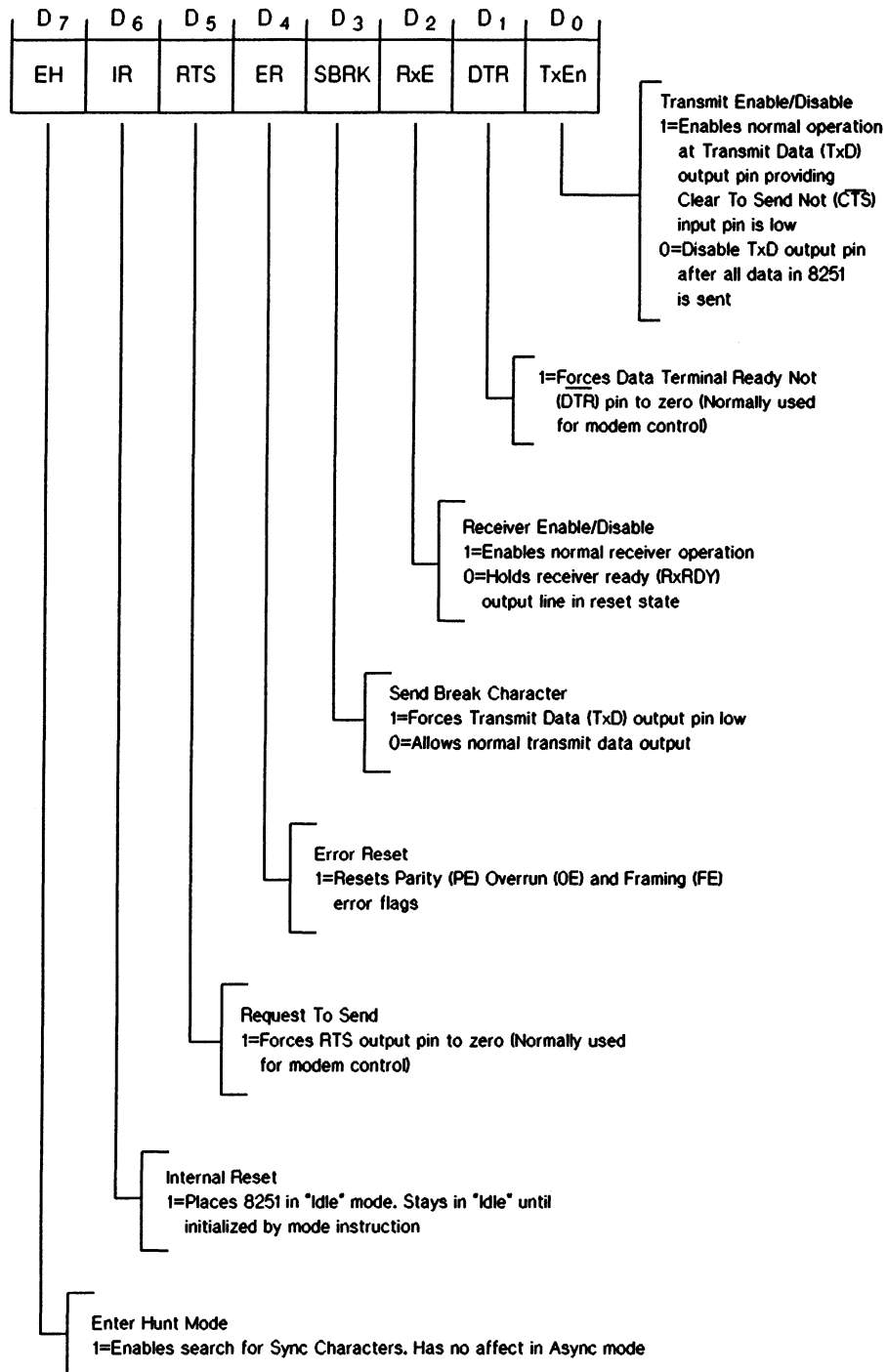


Figure 14-3. Command Mode Instruction Format

Asynchronous Mode

For this mode, the asynchronous mode instruction is placed into location CA+2 and a sync option word specifying 0 must be placed into location CA+3. Locations CA+4 and CA+5 contain no meaningful data.

The asynchronous mode instruction is used to select the baud rate, the character length, the parity parameters, and the number of stop bits. See figure 14-4 below.

The only baud rates which may be used with the HP 64000 are the transmitter clock frequency, $1 \times T_{xc}$, or $1/16 \times T_{xc}$. The baud rate factor of $1/64 \times T_{xc}$ cannot be used with the HP 64000. The basic frequency of T_{xc} is selected by switches on the modem I/O card. Thus, the basic frequency (T_{xc}) may be changed by the I/O card switches.

You must format this instruction so that the appropriate parameters are specified. $1/16 \times T_{xc}$ must be programmed if the baud rate is to match the baud rate table in the System Overview manual.

The sync option specifies 0 since there are no sync characters for the asynchronous mode.

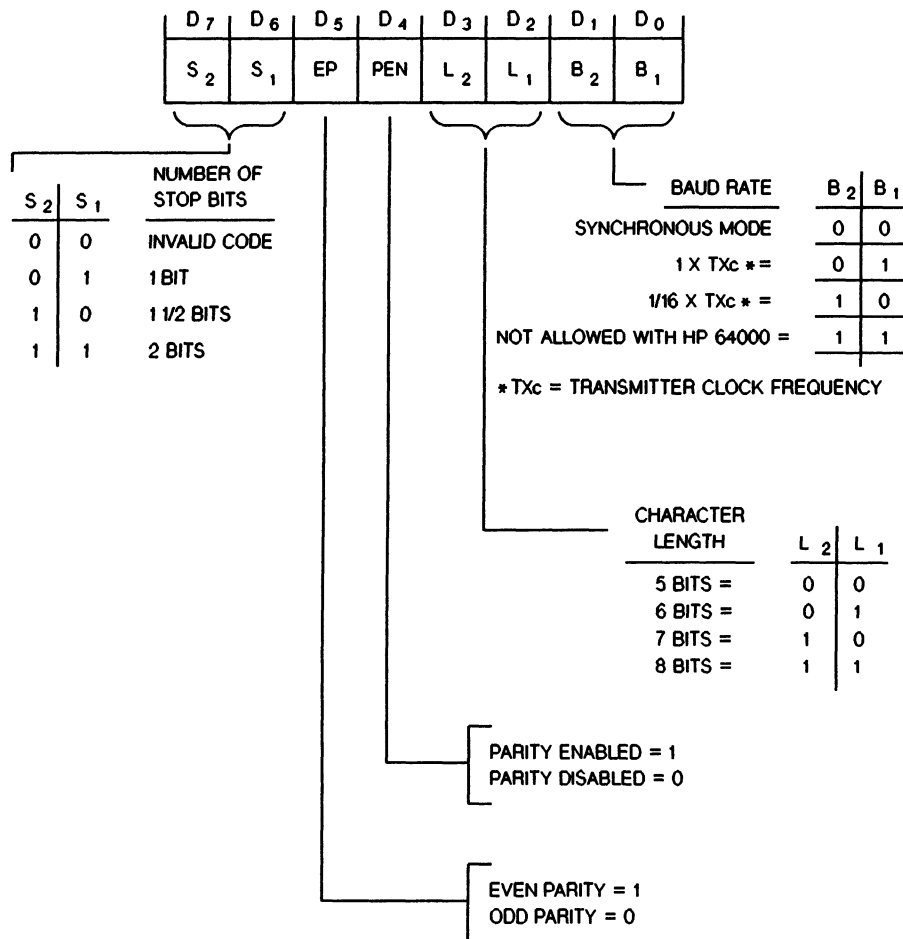


Figure 14-4. Asynchronous Mode Instruction Format

Synchronous Mode/Single Sync Character

For this mode, the synchronous mode instruction is placed into location CA+2, the sync option word specifying "1" is placed into location CA+3, and the sync character is placed into location CA+4. Location CA+5 contains no meaningful data.

The synchronous mode instruction is used to select the character length, and the parity and synchronization parameters. See figure 14-5 below. Bit D7 (SCS) of this word must specify a single sync character. You must format this instruction so that the other appropriate parameters are specified.

The sync option word specifies "1" for a single sync character.

You must specify the format of the sync character.

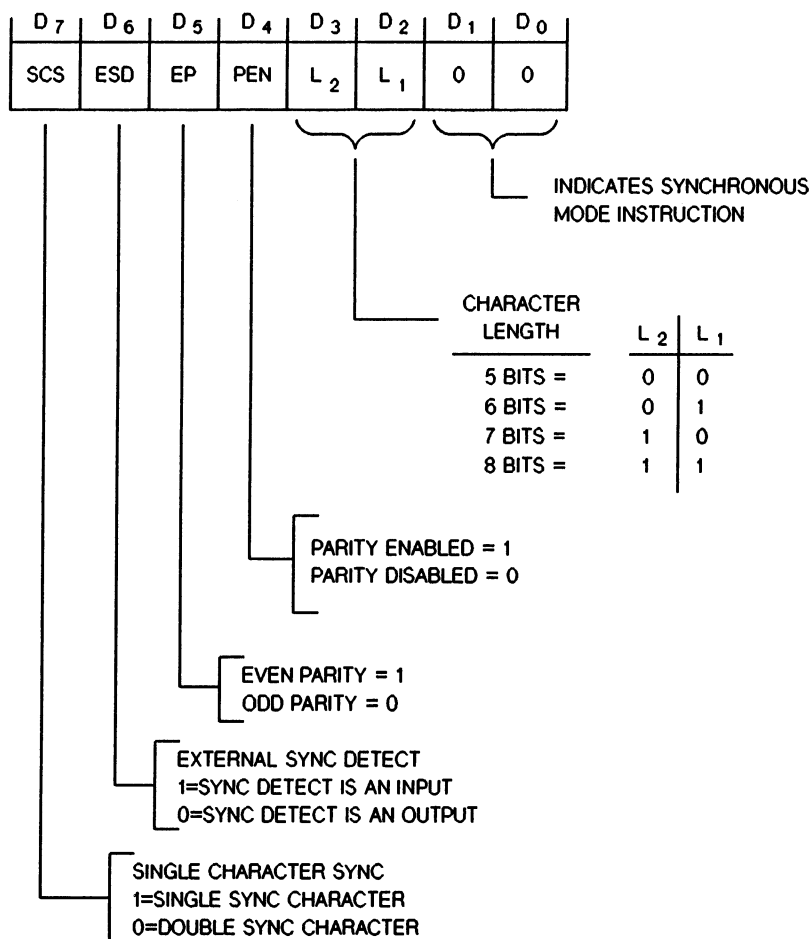


Figure 14-5. Synchronous Mode Instruction Format

Synchronous Mode/Double Sync Character

For this mode, the synchronous mode instruction is placed into location CA+2, the sync option word specifying "2" is placed into location CA+3 and sync characters 1 and 2 are placed into locations CA+4 and CA+5, respectively.

The synchronous mode instruction is used to select the character length, and the parity and synchronization parameters. See figure 14-5. Bit D7 (SCS) of this word must specify a double sync character. You must format this instruction so that the other appropriate parameters are specified.

The sync option word specifies "2" for double sync characters.

You must define the format of both sync characters.

COMMAND TO 8251 USART (83H)

After the 8251 USART is initialized (i.e., reset and asynchronous or synchronous operation selected), it must be placed in the appropriate mode - transmit, receive, or combination transmit/receive etc. To do this, your program first places the appropriately formatted command word into location CA+1 and then places code 83H into the RS-232 CA. You must format the command word to select the applicable operation as shown in figure 14-3.

The HP 64000 responds by supplying the command word to the 8251 USART and returning a 00 to the RS-232 CA. When this cannot be done, code 08 or 09 is returned to the RS-232 CA as shown in table 14-1.

STATUS FROM 8251 (84H)

Your program may check the status of the 8251 at any time. To do this, place code 84H into the RS-232 CA. The HP 64000 responds by returning a 00 to the RS-232 CA and placing the 8251 USART status word in location CA+1.

The status word format is shown in figure 14-6 below.

The status bits D0, D1, and D2 may be cleared or set by the HP 64000 program when operating in any of the buffered modes. If the user desires these bits to control operation, it is necessary to close the appropriate Tx or Rx buffers first.

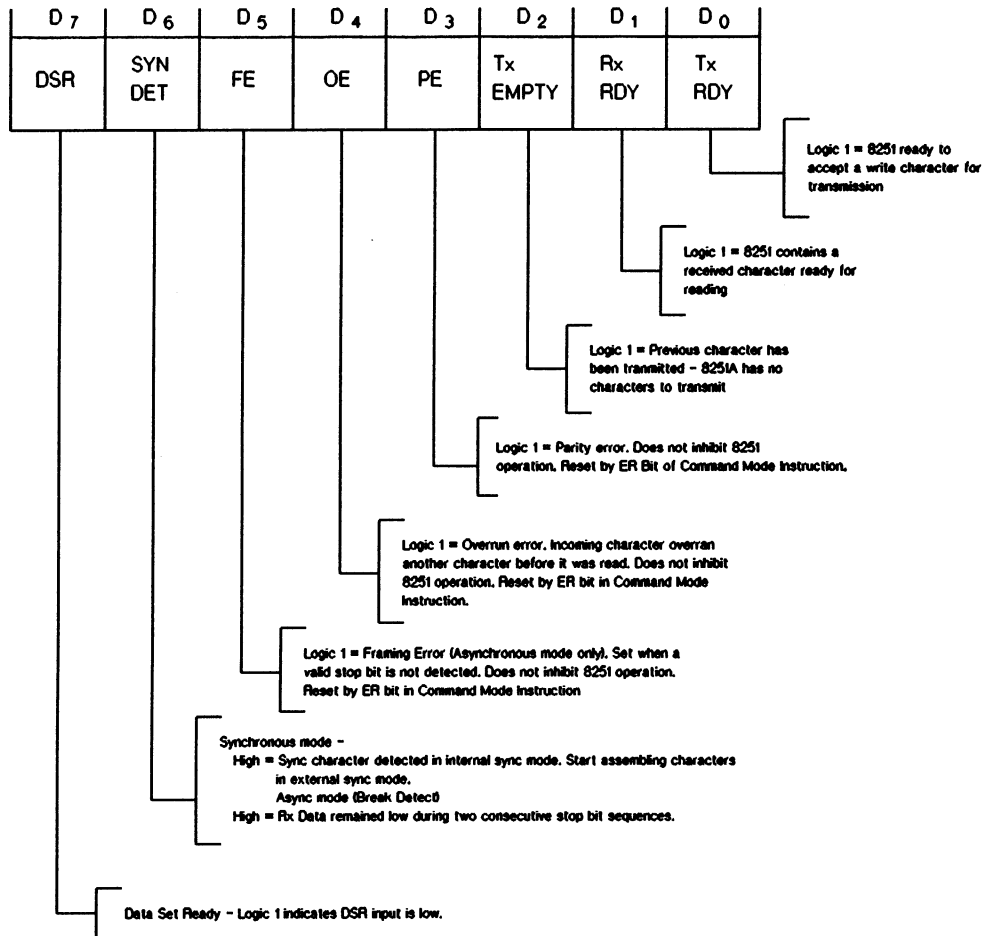


Figure 14-6. 8251 Status Word Format

WRITING TO THE 8251 USART

Your program may write to the 8251 in two ways. It may write a byte at a time, or a write buffer may set up and data written continuously. Both methods are described.

NOTE

Before attempting to write data, the 8251 USART must be initialized and the command word, in the appropriate format, sent to it as described in the previous paragraphs.

When any of the write buffer requests cannot be done, the HP 64000 returns the appropriate error code to the RS-232 CA as shown in table 14-1.

Writing A Single Byte (86H)

To write a single byte to the 8251 USART, your program first places the write byte into location CA+1 and then places code 86H into the RS-232 CA. The HP 64000 responds by supplying the byte to the 8251 USART and returning a 00 to the RS-232 CA. When writing cannot be done, error code 08 or 09 is returned to the RS-232 CA. When more data is to be sent, it is recommended that your program poll the 8251 USART status to determine when it is ready to receive more data.

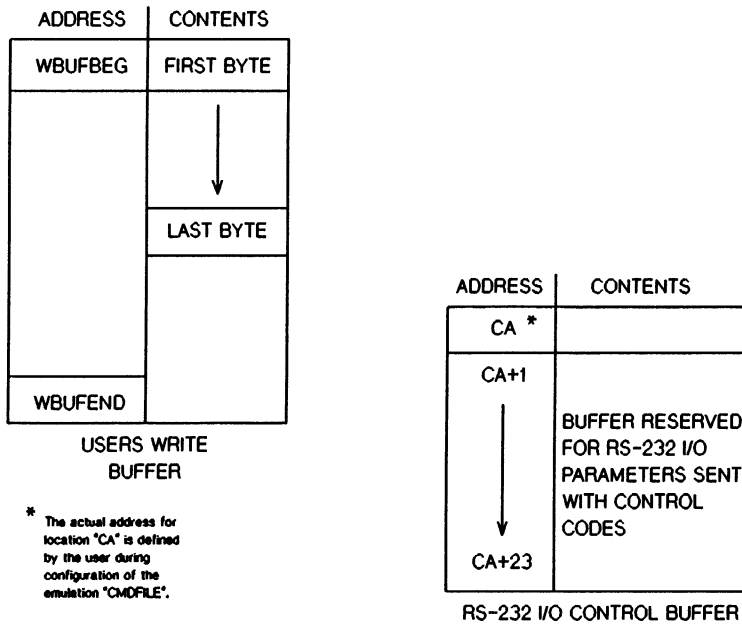
Using A Buffer To Write Multiple Bytes Write Record (87H) Update Write Buffer (89H)

To write a record to the 8251 USART, your program must first set up a write buffer and identify the beginning and ending locations in the buffer. The corresponding HP 64000 write buffer holds a maximum of 256 bytes (see figure 14-7, phase 1). Your program then writes a record into the buffer and identifies the buffer locations into which the first and last bytes of the record are written.

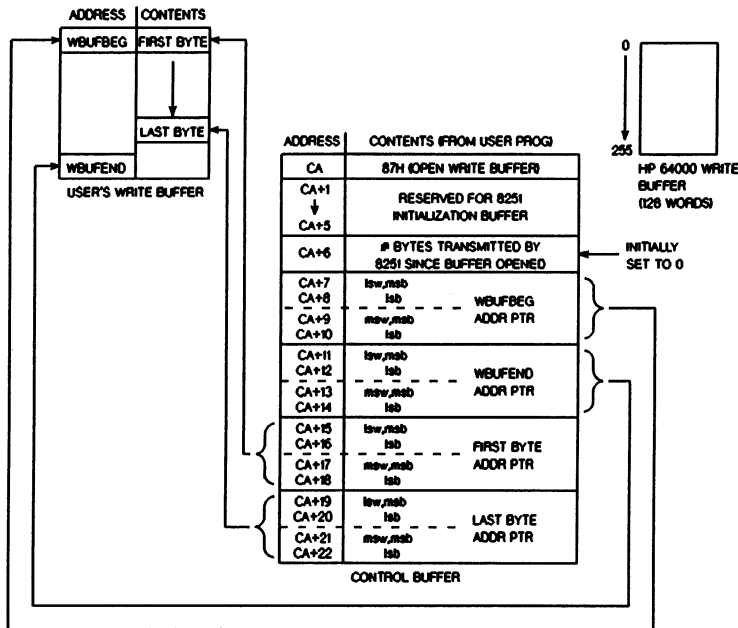
Your program must then request that the record be transferred to the 8251 USART (see figure 14-7, phase 2). This is done by first placing the user write buffer's beginning/ending and first/last byte address pointers into locations CA+7 thru CA+22 and then placing code 87H into the RS-232 CA.

The HP 64000 responds by transferring data from the user's write buffer into a HP 64000 write buffer (see figure 14-7, phase 3). For each byte transferred to the HP 64000 buffer, the first byte address pointer (in locations CA+15 through CA+18) is incremented by one. Data transfer continues until all data in the user's write buffer is transferred or the HP 64000 write buffer becomes full. The HP 64000 write buffer holds a maximum of 256 bytes, or 128 words. When a write buffer is set up and update code 8DH or 89H is used, the number of bytes actually transmitted by the 8251 USART is also entered into location CA+6 by the HP 64000 system. The number of bytes transmitted refers to the number of bytes transmitted from the HP 64000 buffer.

Your program should periodically examine the first and last address byte pointers (if using update code 8DH or 89H, examine the number of bytes transmitted by the 8251 USART) to determine the status of the buffer. When the first and last byte pointers are equal, all data has been transferred to the HP 64000 buffer.

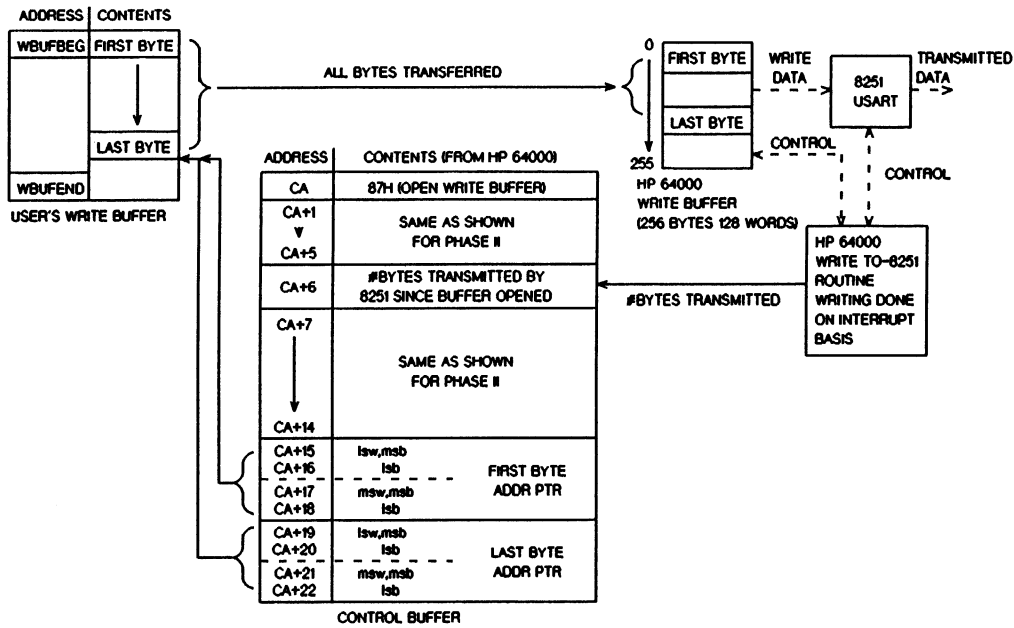


Phase 1

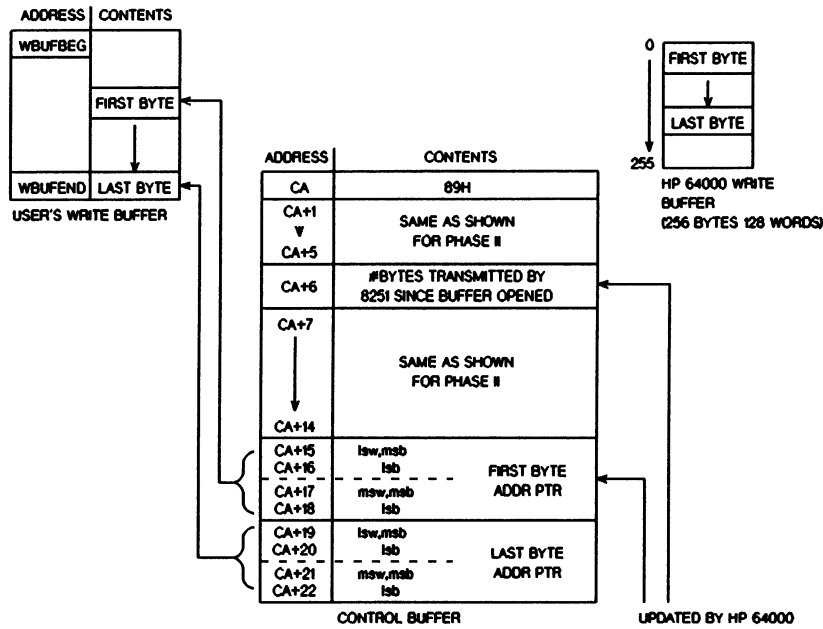


Phase 2

Figure 14-7. Writing RS-232 Record Interface Sequence



Phase 3



Phase 4

Figure 14-7. Writing RS-232 Record Interface Sequence (Cont'd)

When all data has been transferred, your program may either supply another write record, or close the write buffer. When all data has not been transferred, your program may wait until the remaining data is transferred, add more data to the buffer and update the last byte pointer, or close the write buffer. Each of these options is described in the following paragraphs.

Additional data may be added to, or a new record written into the buffer and the last byte address pointer updated as follows: when the first and last byte address pointers are pointing to the same location, the first new byte goes into the location pointed to by both pointers. When the first and last byte address pointers are not pointing to the same location, then the first new byte goes into the location just ahead of the one pointed to by the last byte address pointer (i.e., last byte address pointer + 1). Then the following bytes are entered to succeeding locations (see figure 14-7, phase 3).

After entering data into the buffer, your program requests a write data transfer. This is done by first placing the updated last byte address pointer into locations CA+19 thru CA+22 and then placing code 89H into location CA (see figure 14-7, phase 4).

The HP 64000 responds by transferring data from the user's write buffer to the HP 64000 write buffer, increments the first byte address pointer for each byte transferred, and if update code 8DH or 89H is being used, the number of bytes sent by the 8251 USART is also updated.

Once your program has placed code 8DH or 89H (update buffer) into the RS-232 CA, the HP 64000 routinely monitors the last byte address pointer to determine when more data has been loaded into the user's write buffer. When the HP 64000 detects that the last byte address pointer has been incremented, it transfers the data and increments the first byte address pointer to indicate the number of bytes written. It also updates the number of bytes sent by the 8251 USART.

To write another record, your program updates the last address pointer. The HP 64000 responds as described above.

Data may be stored in the user's write buffer using a "wrap around" method. That is, once the last location in the buffer is filled, the next byte is placed into the first location of the buffer. Thus, it is possible for the last byte address pointer to be pointing to an address that is less than (i.e., ahead of) the first byte address.

Closing The Write Buffer (88H)

To close the write buffer, your program places code 88H in the RS-232 CA. The HP 64000 closes the write buffer and returns a 00 to the RS-232 CA.

READING FROM THE 8251 USART

Reading data from the 8251 USART is similar to writing data to it. Your program may read data in two ways. It may read a byte at a time or it may set up a read buffer and read a record at a time. Both methods are described.

NOTE

Before attempting to read data, the 8251 USART must have been initialized and the command word, in the applicable format, sent to it as described in the previous paragraphs.

When any of the read buffer requests cannot be done, the HP 64000 returns the appropriate error code to the RS-232 CA as shown in table 14-1.

Reading A Single Byte (85H)

To read a single byte from the 8251 USART, your program places code 85H into the RS-232 CA.

The HP 64000 responds by returning a 00 to the RS-232 CA and the read byte to location CA+1. When reading cannot be done, error code 08 or 09 is returned to the RS-232 CA.

The HP 64000 returns whatever character is in the Rx buffer of the 8251 USART. It is recommended that your program check the status of the 8251 USART to see when Rx RDY is true before performing the single byte read. Any read operation clears Rx RDY, indicating that the character in the buffer has been read.

Using A Buffer To Read Multiple Bytes Read Record (8AH) Update Read Buffer (8CH)

To read a record from the 8251 USART, your program must first set up a read buffer and identify the beginning and ending locations in the buffer (see figure 14-8, phase 1).

This is done by first placing the address pointers into locations CA+24 thru CA+39 and then placing code 8AH into the RS-232 CA (see figure 14-8, phase 2). Locations CA+24 thru CA+31 contain the address pointers for the beginning and ending locations of the user's read buffer.

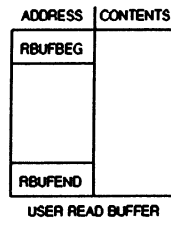
Locations CA+32 thru CA+39 contain the address pointers for the first and last bytes written into the buffer. These pointers are both initially set to point to the first location in the user's read buffer. This indicates that the buffer is empty. The HP 64000 forces the first data pointer to always point to the beginning of the buffer.

The HP 64000 responds by continuously transferring data from the 8251 USART to the HP 64000 read buffer (see figure 14-8, phase 3). Your program must then issue an 8CH or 8DH to transfer the data to the user's buffer. For each byte transferred into the user's read buffer, the last byte address pointer is incremented by one (see figure 14-8, phase 4). In addition, when update code 8DH or 8CH is being used, the number of bytes received by the 8251 USART and transferred into the HP 64000 is entered into location CA+23.

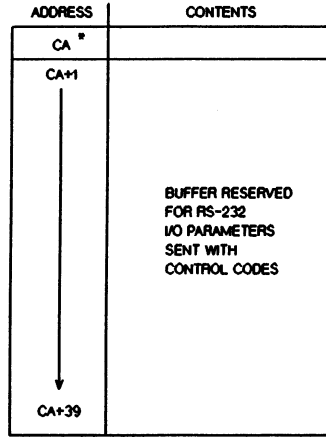
To determine when and how much data is available, your program must monitor the last byte address pointer and the number of bytes received. When data is found in the buffer, your program should process the data. When all data expected is received, your program may then close the read buffer.

Once your program has placed code 8CH or 8DH into the RS-232 CA, the HP 64000 periodically monitors the output of the 8251 USART, transfers data into the user read buffer, and updates the last byte address as required. Your program in turn monitors the last byte address pointer to determine when more data is available. This process continues until your program closes the read buffer.

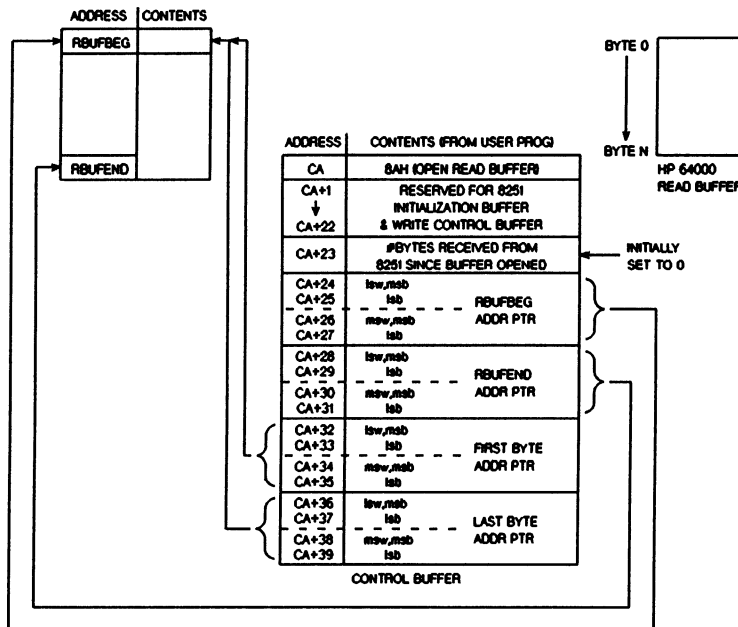
When code 8CH or 8DH is being used, and your program issues an 8AH again, the buffer is frozen, yet the HP 64000 continues to receive data into its buffer.



* The actual address for location "CA" is defined by the user during configuration of the emulation "CMDFILE".

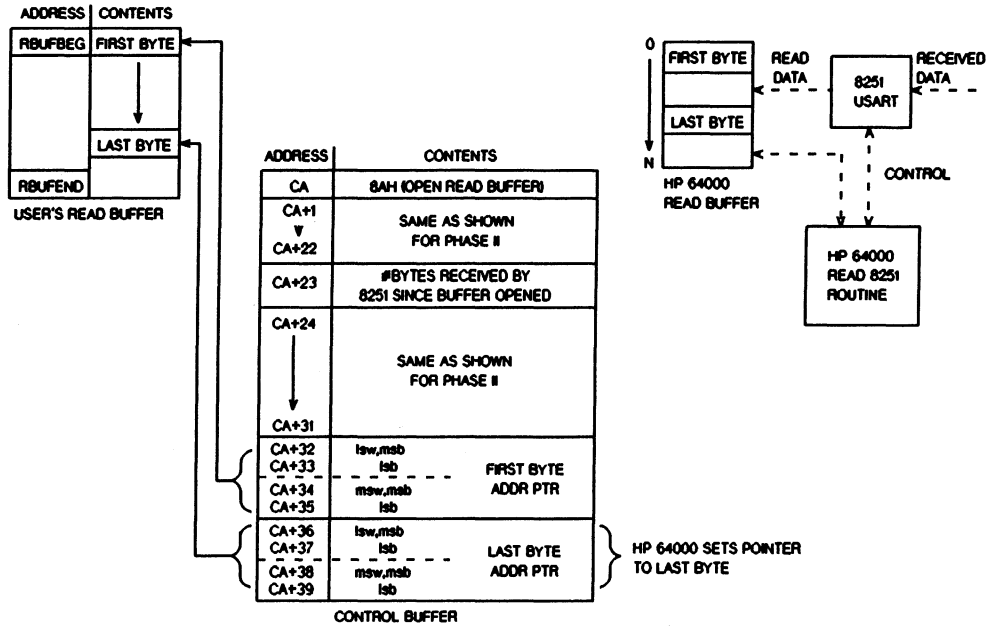


Phase 1

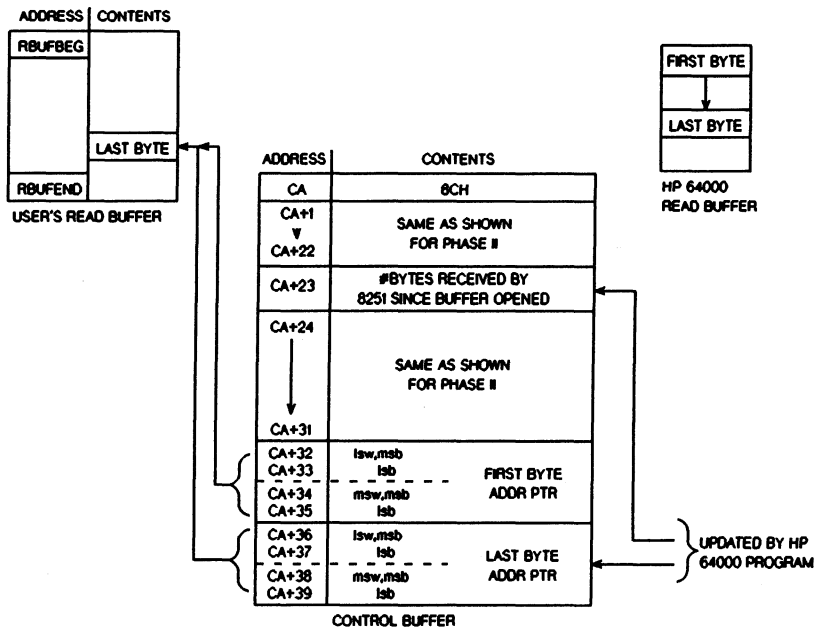


Phase 2

Figure 14-8. Reading RS-232 Record Interface Sequence



Phase 3



Phase 4

Figure 14-8. Reading RS-232 Record Interface Sequence (Cont'd)

Data may be stored in the user's read buffer using a "wrap-around" method. That is, once the last location in the buffer is filled, the next byte is placed into the first location of the buffer. Thus, it is possible for the last byte address pointer to be pointing to an address that is less than (i.e., ahead of) the first byte address.

Closing The Read Buffer (8BH)

To close the read buffer, your program places code 8BH into the RS-232 CA. The HP 64000 closes the buffer and returns a 00 to the RS-232 CA.

UPDATING READ/WRITE BUFFERS (8DH)

Once the read and write buffers have been set up and opened, the buffers may both be updated by using one code. To do this, your program places the updated first and last byte address pointers for both the read and write buffers into the corresponding locations and then places code 8DH into the RS-232 CA.

The HP 64000 responds to the update request as described previously. However, in addition to setting, monitoring, and updating the first and last byte address pointers, the number of bytes received and transmitted by the 8251 USART is also set, updated, and monitored. This provides an additional indication of how much data is sent and received.

CLOSING THE RS-232 FILE (81H)

To close the RS-232 file, your program must place code 81H into the RS-232 CA.

The HP 64000 responds by closing the RS-232 file and returning a 00 to the RS-232 CA. When the file cannot be closed, error code 08 or 09 is returned to the RS-232 CA.

Table 14-1. RS-232 Control Codes

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
	Address	Contents	VALID USER REQUEST		INVALID REQUEST
			Address	Contents	Error Code
OPEN RS-232 FILE	CA	80H	CA	00	01-07: NA 08 09: File already open. 10-14: NA
CLOSE RS-232 FILE	CA	81H	CA	00	01-07: NA 08 09: File not open. 10-14: NA
INITIALIZE 8251	CA	82H	CA	00	Same as 81H, above
	CA+1	Command Instruction			
	CA+2	Mode Instruction			
	CA+3	Sync Option word			
	CA+4	Sync Character, one			
	CA+5	Sync Character, two			

Table 14-1. RS-232 Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
			VALID USER REQUEST	INVALID REQUEST	
	Address	Contents	Address	Contents	Error Code
COMMAND TO 8251	CA	83H	CA	00	Same as 81H, above
	CA+1	Command Word			
STATUS FROM 8251	CA	84H	CA	00	Same as 81H, above
			CA+1	Status Word	
READ SINGLE BYTE FROM 8251	CA	85H	CA	00	Same as 81H, above
			CA+1	Byte Read	
WRITE SINGLE BYTE TO 8251	CA	86H	CA	00	Same as 81H, above
	CA+1	Write Byte			
OPEN WRITE BUFFER	CA	87H	CA	87H	
	CA+1	Reserved for Initialization buffer	The HP 64000 transfers write data from the users buffer to the HP 64000 buffer.		
	↓ CA+5				

Table 14-1. RS-232 Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
	Address	Contents	VALID USER REQUEST	INVALID REQUEST	Error Code
	CA+6	#Bytes sent by 8251. Cleared by open (87H). Updated by HP 64000 when update code 89H or 8DH is used.	For each byte transferred to the HP 64000 buffer, first byte address pointer is incremented by one.		
	CA+7 (lsw, msb)	Buffer Begin Address pointer			
	CA+8 (lsw, lsb)				
	CA+9 (msw, msb)				
	CA+10 (msw, lsb)				
	CA+11 (lsw, msb)	Buffer End Address pointer			
	CA+12 (lsw, lsb)				
	CA+13 (msw, msb)				
	CA+14 (msw, lsb)				

Table 14-1. RS-232 Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
	Address	Contents	VALID USER REQUEST		INVALID REQUEST
			Address	Contents	Error Code
	CA+15 (lsw, msb)	First Byte Address pointer			
	CA+16 (lsw, lsb)				
	CA+17 (msw, msb)				
	CA+18 (msw, lsb)				
	CA+19 (lsw, msb)	Last Byte Address pointer			
	CA+20 (lsw, lsb)				
	CA+21 (msw, msb)				
	CA+22 (msw, lsb)				

Table 14-1. RS-232 Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
	Address	Contents	Address	Contents	Error Code
CLOSE WRITE BUFFER	CA	88H	CA	00	Same as 81H, above.
UPDATE WRITE BUFFER	CA	89H	CA	89H	Same as 81H, above.
	CA+1	Reserved for Initialization Buffer	The user updates the last byte address Pointer to indicate how much new write data is in the buffer. The HP 64000 processes the write data, increments the first byte addr. pointer, and updates # bytes sent by 8251 as required.		
	CA+5	# Bytes sent by 8251.			
	CA+6				
	CA+7	Not changed by user.			
	↓				
	CA+14	First Byte Address pointer			
	CA+15 (lsw, msb)				
	CA+16 (lsw, lsb)				
	CA+17 (msw, msb)				
	CA+18 (msw, lsb)				
	CA+19 (lsw, msb)			Updated Last Byte Address pointer	
CA+20 (lsw, lsb)					
CA+21 (msw, msb)					
CA+22 (msw, lsb)					

Table 14-1. RS-232 Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
			VALID USER REQUEST	INVALID REQUEST	
	Address	Contents	Address	Contents	Error Code
OPEN READ BUFFER	CA	8AH	CA	8AH	Same as 81H, above
	CA+1	Reserved for Initialization and write buffers. # Bytes received by 8251. Cleared by open (8AH). Updated by HP 64000 when update code 8CH or 8DH is used.	The user sets first and last address pointers to point to buffer beginning address. The HP 64000 will transfer data from the 8251 to the HP 64000 buffer. The user must use the commands 8CH or 8DH to transfer the data to the users buffer.		
	↓				
	CA+22				
	CA+23				
	CA+24 (lsw, msb)				
	CA+25 (lsw, lsb)				
	CA+26 (msw, msb)				
	CA+27 (msw, lsb)				
CA+28 (lsw, msb)	Buffer End Address pointer				
CA+29 (lsw, lsb)					

Table 14-1. RS-232 Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
	Address	Contents	VALID USER REQUEST		INVALID REQUEST
	Address	Contents	Address	Contents	Error Code
	CA+30 (msw, msb)				
	CA+31 (msw, lsb)				
	CA+32 (lsw, msb)	First Byte Address pointer			
	CA+33 (lsw, lsb)				
	CA+34 (msw, msb)				
	CA+35 (msw, lsb)				
	CA+36 (lsw, msb)	Last Byte Address pointer			
	CA+37 (lsw, lsb)				
	CA+38 (msw, msb)				
	CA+39 (msw, lsb)				

Table 14-1. RS-232 Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
			VALID USER REQUEST	INVALID REQUEST	
	Address	Contents	Address	Contents	Error Code
CLOSE READ BUFFER	CA	8BH	CA	00	Same as 81H, above
UPDATE READ BUFFER	CA	8CH	CA	80H	Same as 81H, above
	CA+1 ↓ CA+22	Reserved for initialization and write buffers.			
	CA+23	#Bytes received by 8251.			
	CA+24 ↓ CA+31	Not changed by user.			
	CA+32 (lsw, msb)	First Byte Address pointer			
	CA+33 (lsw, lsb)				
	CA+34 (msw, msb)				
	CA+35 (msw, lsb)				
				The HP 64000 continues to transfer data, increments last byte address pointer, (updates #Bytes received by 8251) as required. User program monitors these parameters to determine how much data is received. (HP 64000 forces first byte address pointer to always point to the beginning of the buffers.)	

Table 14-1. RS-232 Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
			VALID USER REQUEST		INVALID REQUEST
	Address	Contents	Address	Contents	Error Code
	CA+36 (lsw, msb)	Last Byte Address pointer			
	CA+37 (lsw, lsb)				
	CA+38 (msw, msb)				
	CA+39 (msw, lsb)				

Table 14-1. RS-232 Control Codes (Cont'd)

REQUEST NAME	USER PROGRAM REQUEST		HP 64000 RESPONSE TO:		
	Address	Contents	Address	Contents	Error Code
UPDATE WRITE/ READ BUFFERS	CA	8DH	CA	00H	Same as 81H, above
	CA+1 ↓ CA+5	Reserved for Initialization Buffer			
	CA+6 ↓ CA+22	Same as shown for update Write, Buffer, above.	Write and read buffers are both updated as described above.		
	CA+23 ↓ CA+39	Same as shown for update Read Buffer, above.			

NOTES

Appendix A

SYNTACTICAL VARIABLE DEFINITIONS

The syntactical variables used throughout this manual are described in this appendix.

<ABSFILE>

The <ABSFILE> is the file identifier of an absolute file that contains the emulation program. The emulation program is placed into the file by assembling and linking to the file before application to the target microprocessor. <ABSFILE> has the same format requirements as the <FILE> variable which is described later in this appendix.

<ADDRESS>

The <ADDRESS> variable defines a bit pattern of up to 32 bits which specifies a particular location in mapped memory. That bit pattern can be represented by a binary, octal, hexadecimal, or decimal number; a local or global symbol; or a mathematical combination of numbers or symbols. <ADDRESS> has the same format requirements as the <VALUE> variable which is described later in this appendix.

<ADR_LST>

The variable <ADR_LST> contains a list of addresses, separated by commas, where the addresses are within the address space defined by the processor.

<CMDFILE>

The <CMDFILE> variable is the file identifier for an existing emulation configuration file. This command file contains the organizational commands for the processor to be emulated. The command file can be retained or modified for further use. <CMDFILE> has the same requirements as the <FILE> variable which is described later in this appendix.

<FILE>

The <FILE> variable is used to identify files generated or accessed by the development system commands. <FILE> consists of the following parameters:

<FILE NAME>[:<USERID>][:<DISC#>]

where:

<FILE NAME> is the identifier given to a particular file. <FILE NAME> must begin with an upper case alphabetic character and can have a total length of nine characters. After the first character, any upper or lower case alphanumeric character or an underscore can be used. If more than nine characters are specified, the name is truncated to the first nine characters.

<USERID> is the identifier assumed by a particular system user. <USERID> must begin with an upper case alphabetic character and can have a total length of six characters. The characters following the first character can be any upper or lower case alphanumeric characters, including the underscore. If more than six characters are specified, the userid is truncated to the first six characters. If a userid is not entered, the current userid is used as the default.

<DISC#> specifies the disc on which the file is stored. <DISC#> can be any digit from 0 through 7, but it must correspond to the Logical Unit number assigned to one of the discs at system power up. The default is to search the discs for the file specified, or to create the file on disc zero.

<REAL>

The <REAL> variable is an alphanumeric representation of a short real or a long real number value. The syntax is:

$$\left\{ \begin{array}{l} [+] \\ [-] \end{array} \right\} \langle \text{integer} \rangle \quad . \langle \text{integer} \rangle \quad [\text{ E } \left\{ \begin{array}{l} [+] \\ [-] \end{array} \right\} \langle \text{integer} \rangle]$$

Where <integer> is an unsigned decimal integer.

A short real number consists of 32 bits, with 6-7 significant digits. It has an approximate absolute value decimal range of from less than or equal to 8.34×10^{-37} through an absolute value of less than or equal to 3.37×10^{38} .

A long real number consists of 64 bits, with 15-16 significant digits. It has an approximate absolute value decimal range of from less than or equal to 4.19×10^{-307} through an absolute value of less than or equal to 1.67×10^{308} .

<STATE>

The <STATE> variable specifies a particular state on the emulation bus. The <STATE> expression consists of an address, a data, and a status specification.

<VALUE>

<VALUE> is a syntactical variable that allows specification of symbols (labels), numbers, parentheses, and math operators (+, -, /, (), *) following standard algebraic rules to produce a value. Legal operands are defined as follows:

<NUMBER> is an alphanumeric representation of a 16 bit pattern of ones, zeros, and don't cares (X's). The bit pattern can be represented in binary, octal, hexadecimal, or decimal where binary is indicated by a "B", octal by a "Q", hexadecimal by an "H", and decimal by a "D". Decimal is the default value and the use of "D" is optional.

Examples:

```
(A+B)*C  
10101011XXXXXXB  
145XXXQ  
2563
```

The <LOCAL SYMBOL> variable represents the name of a symbol which can only be used by the program module in which it is defined. The <GLOBAL SYMBOL> variable represents the name of a symbol which can be called by program modules other than the one in which it is defined. The global symbol must be declared as such by a GLB statement in the source file.

<LOCAL SYMBOL> is specified as: SYMBOL_NAME [:<MODULE>] or: #<LINE #> [:<MODULE>] where <MODULE> is the same as <FILE>. For PASCAL programs, lines which generate object code produce local line # symbols corresponding to the source line.

<GLOBAL SYMBOL> is specified as <SYMBOL_NAME> or @:<MODULE> which produces the starting address of the specified <MODULE>.

<MODULE> specifies the file in which the local symbol is defined. If no <MODULE> is specified, the global symbol table associated with the absolute program file loaded by the emulator is searched for the <SYMBOL_NAME>. If the symbol name is not found in the global symbol table, a search is made of the last referenced local symbol table. If the symbol name is not found in the local symbol table, an error message is displayed on the status line. For more information, refer to the description of <FILE> which is included in this appendix.

<STRING> is an ASCII string delimited by ", ', ^, and produces a 32 bit code.

Examples:

```
'A'      00000041H  
"AB"     00004142H  
^ABC^    00414243H  
'ABCD'   41424344H
```

NOTES

Appendix B

680XX REGISTER FORMAT AND NAMES

GENERAL

In this chapter, whenever 680XX is mentioned (even as part of a filename) both the 68000 and the 68008 are assumed unless there are differences between the two processors. In that case, the differences will be noted. When the name 680XX appears in any program listings, just insert the appropriate name of your processor.

M680XX Registers

```
Next PC  000000  STATUS 1EBE < xnzv >    USPT  0000761A  SSPT  00007622
D0-D7  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
A0-A7  00000000  00000000  00000000  00000000  00000000  00000000  00000000  0000761A
```

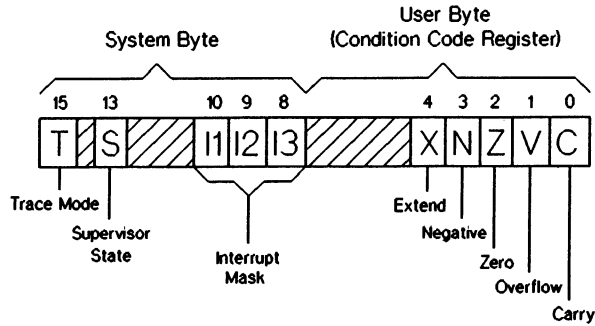
NOTE

The offset is applied to the PC register only
(Next PC = (3900H - 0100H = 3800H)).

To modify a register, use one of the following names:

D0, D1, D2, D3, D4, D5, D6, D7	-Data Registers
A0, A1, A2, A3, A4, A5, A6, A7	-Address Registers
USPT	-User Stack Pointer
SSPT	-System Stack Pointer
STATUS	-Status

The format for the status register is as follows:



Courtesy Motorola Inc.

Figure B-1. Status Register Format

Polarities for the following bits are: 0 = clear; 1 = set

Bit 15 <t>	-Trace Mode
Bit 13 <s>	-Supervisor Mode
Bits 10,9,8	-Interrupt Mask
Bits 4,3,2,1,0	-Condition Codes
Bit 4 <x>	-Extend
Bit 3 <n>	-Negative
Bit 2 <z>	-Zero
Bit 1 <v>	-Overflow
Bit 0 <c>	-Carry

NOTE

The t,s,n,z,v,c status conditions are displayed only if set

Appendix C

GLOSSARY OF SOFTKEY LABELS

INTRODUCTION

This appendix contains a list of the softkey labels provided in the emulation software. The corresponding command line message is given for each softkey label and an explanation of the softkey label follows. An example is also given which shows the message as it would appear on the command line of the display.

Command Line Message	Softkey Label
<i>about</i>	<i>about</i>
Used with the <i>trace</i> command to cause the trigger point to occur in the center of the trace buffer. <i>trace about <ADDRESS></i>	
<i>absolute</i>	<i>absolute</i>
Used with the <i>display memory</i> and <i>display trace</i> commands to cause the information being displayed to be displayed in hex format. <i>display memory absolute</i> <i>display trace absolute</i>	
<i>access_size_to</i>	<i>acc_size</i> ***(68000 only)***
Used with the <i>modify</i> when changing the amount of memory to be moved when using "MOVE_MEM" to either byte or word. <i>modify access_size_to byte</i> <i>modify access_size_to word</i>	

Command Line Message	Softkey Label
<i>address</i>	<i>address</i>
Used with the <i>run</i> and <i>trace</i> commands to specify an address or address range, a "not" address or address range, a local or global symbol, or a source program line number. Implied if no keyword is given. May be skipped if a comma (,) is entered.	
<i>trace after address <ADDRESS></i> <i>run until address range <ADDRESS> thru <G_L_SYM></i> <i>trace after , , <VALUE></i> <i>run until , , <VALUE></i>	
<i>after</i>	<i>after</i>
Used with the <i>trace</i> command to cause all trace data to be captured after the trigger point. The trigger point will be in the beginning of the trace buffer.	
<i>trace after <ADDRESS></i>	
<i>again</i>	<i>again</i>
Used with the <i>trace</i> command to repeat the previous measurement.	
<i>trace again</i>	
<i>analysis_mode_to</i>	<i>anly_mode</i>
Used to change analysis mode from bus_cycle_data to execution mode and vice versa.	
<i>modify analysis_mode_to execution_data</i>	
<i>and</i>	<i>and</i>
Used with the <i>trace</i> command to "and" statuses in a trace specification.	
<i>trace after status int_ack and read occurs 2</i>	
<i>before</i>	<i>before</i>
Used with the <i>trace</i> command to cause all trace data to be captured before the specified trigger point. The trigger point will be at the end of the trace buffer.	
<i>trace before <ADDRESS></i>	

Command Line Message	Softkey Label
----------------------	---------------

binary

binary

Used with the *display trace status* commands to cause data to be displayed in binary format.

display trace status binary

blocked

blocked

Used with the *display memory* commands to cause memory to be displayed in a blocked rather than linear mode.

display memory blocked byte

break

break

Pulls on the NMI line to break the processor into the emulation monitor program.

break

break_on

break_on

Used with the *trace* command. At the end of the measurement, or at the trigger point, will cause the processor to break into the emulation monitor program.

trace break_on trigger

bus_cycle_data

bus_cycle

Used with the *modify anly_mode* commands to set analysis mode to *bus_cycle_data*. This mode will cause the analyzer to capture all cycles occurring on the emulation bus.

modify analysis_mode_to bus_cycle_data

byte

byte

Used with the *display memory* commands to specify a display in byte rather than word mode.

display memory blocked byte

Command Line Message	Softkey Label
----------------------	---------------

clear

clear

Used with the *modify sftw_bpts* commands to clear breakpoints that are no longer desired. If no address or symbol is specified, all breakpoints will be cleared.

modify software_breakpoints clear
modify software_breakpoints clear <G_L_SYM>

configuration

config

Used with the *modify* command to reenter a configuration previously set up in order to make modifications.

modify configuration

count

count

Used with the *display trace* commands to specify whether count is to be displayed absolute or relative.

display trace count absolute

counting

counting

Used with the *trace* command to specify whether count is to be based on time or on states.

trace counting status user

data

data

Used within a trace specification. Specifies tracing on a data value. May be omitted by using a comma (,) in place of the *data* softkey.

trace after data <VALUE> or status <VALUE>
trace after , , data_acc

display

display

Used to specify what type of display is desired (e.g.; display memory, display trace, display registers, display glob_sym, display loc_sym, sftw_bpts, io_port).

display registers

Command Line Message

Softkey Label

emul_memory

emul_mem

Used with the *load memory* commands, to load only the emulation portion of memory with an absolute file you specify.

load emul_memory <FILE>

end

end

Used to end out of emulation into the meas_sys or the system monitor level of softkeys.

end

execute

execute

Used with multi-module systems to execute run or trace specifications that have been set up using the *specify* command.

execute

execution_data

exec_data

Used with the *modify only_mode* commands to set analysis mode to execution_data mode. In this mode, only executed cycles will be captured by the analyzer.

modify analysis_mode_to execution_data

from

from

Used with the *run* command to specify a transfer address.

run from <ADDRESS>

global_symbols

glob_sym

Used with the *display* command to display global symbols. Used with the *list printer* and *list <FILE>* commands to print the global symbols or to keep them in a file that you name.

display global_symbols

Command Line Message

Softkey Label

halt

halt

Used to halt execution of analysis and anything going on over the IMB. This softkey is available at the *meas_sys* level softkeys when you end out of emulation with a trace measurement in progress. It will also appear during an emulation session in place of the *execute* softkey when a trace measurement is in progress.

end **RETURN** *halt*

hex

hex

Used with the *display trace status* commands to specify that status be displayed in hex format.

display trace status hex

io_port

io_port

Used with the *display* and *modify* commands to display and modify io ports.

display io_port <ADDRESS> *blocked byte*
modify io_port <ADDRESS> *to* <VALLIST>

list

list

Used to list information to a file that you name or to the printer. Can be used anywhere that *display* is used.

list <FILE> *memory blocked byte*
list printer trace mnemonic count relative

load

load

Used to load an absolute file into memory or to load a trace file that was previously saved (stored).

load <FILE>
load trace <FILE>

local_symbols_in

loc_sym

Used with the *display* command to display local symbols in an *asmb_sym* file. Also used with the *list printer* and *list* <FILE> commands to print the local symbols or to keep the local symbols list to a file that you name.

display local_symbols_in <FILE>

Command Line Message

Softkey Label

long

long

Used with the *display memory real* and *modify memory real* commands to specify that the display or modification of an address or symbol be made in longreal numbers format (refer to appendix A for the definition of the longreal numbers format).

display memory <ADDRESS> real long
modify memory real long <ADDRESS> to <REAL>

measurement_complete

meas_comp

Used with the *wait* and *trace break_on* commands. The *wait* command will cause a delay until a measurement completes and then continue a command file. The *trace break_on* command will cause a break into the monitor at measurement complete.

wait measurement_complete
trace break_on measurement_complete

memory

memory

Used with the *modify* and *display* commands to modify and display the contents of specified memory locations.

modify memory <ADDRESS> to <VALLIST> , <VALLIST> , <VALLIST>
display memory <G_L_SYM> real long

mnemonic

mnemonic

Used with the *display* command to specify a mnemonic memory or a mnemonic trace display.

display memory mnemonic
display trace mnemonic count relative

modify

modify

Used with *memory*, *sftw_bpts*, *register*, *config*, *io_port*, and *anly_mode* to specify a new setup of the selected item.

modify register <REGNAME> to <VALUE>
modify software_breakpoints set <ADDRESS>

not

not

Used with *run until* and *trace* commands to negate a condition.

trace after address not range <ADDRESS> thru <ADDRESS>

Command Line Message	Softkey Label
<i>occurs</i>	<i>occurs</i>
Used to specify the number of times an event must occur before the trigger is armed.	
<i>run until status data_acc occurs <#_TIMES></i>	
<i>offset_by</i>	<i>offset_by</i>
Used with the <i>display</i> and <i>list</i> commands to specify the offset to be used when displaying memory, trace, registers, or software breakpoints. In the case of memory, trace, and software breakpoints, the addresses displayed will have the offset subtracted from them. In the case of registers, the CS/IP will have the offset subtracted from it.	
<i>display registers offset by <ADDRESS></i>	
<i>only</i>	<i>only</i>
Used with the <i>trace</i> command to specify that only certain specified data will be captured.	
<i>trace after status int_ack only address range <ADDRESS> thru <ADDRESS></i>	
<i>or</i>	<i>or</i>
Used with the <i>trace</i> command to add "or" specifications to the trace setup.	
<i>trace after status supervis or address <VALUE></i>	
<i>printer</i>	<i>printer</i>
Used with the <i>list</i> command to specify that selected data is to be listed to the printer.	
<i>list printer global_symbols</i>	
<i>range</i>	<i>range</i>
Used with the <i>trace</i> command to specify a range to be included in the trace specification.	
<i>trace after address range <ADDRESS> thru <ADDRESS></i>	

Command Line Message

Softkey Label

real

real

Used with the *modify* and *display* commands to modify or display memory in real number format (refer to appendix A for the definition of real number formats).

display memory <ADDRESS> real short

register

register

Used with the *modify* command to modify the contents of a register (see appendix B for a listing of the 680XX registers).

modify register <REGNAME> to <VALUE>

registers

registers

Used with the *display* command to display the registers and their contents. Also used with the *list* command to list the registers and their contents to the printer or to a file that you name (see appendix B for a listing of the 680XX registers).

display registers
list printer registers

relative

relative

Used with the *display trace* commands to specify that the count time shown for a particular state is relative to the state shown directly above it on the display (as opposed to absolute, which is time count from the very beginning of the trace).

display trace count relative

repetitively

rept

Used with the *trace* and *display* commands to cause the specified action to be performed repetitively.

display memory repetitively

reset

reset

Used to reset the emulator processor. Causes the emulator processor to jump to the emulation monitor program.

reset

Command Line Message

Softkey Label

run

run

Used to cause the processor to run. Initially, the *run* command will initialize the emulation monitor program and cause the processor to run in the monitor. Pressed again, it starts your program at the point designated in your program by the ORG statement, or at the point you specified when you linked your program(s). It can also be used to start the run from a particular location, with the end of the run based on specified parameters.

run

run from <ADDRESS> until status read occurs <#_TIMES>

set

set

Used with the *modify sftw_btps* commands to set a software breakpoint at a specific address, global symbol, or local symbol.

modify software_breakpoints set <ADDRESS>

short

short

Used with the *display memory real* and *modify memory real* commands to specify that the display or modification of an address or symbol be made in shortreal numbers format (refer to appendix A for the definition of the shortreal numbers format).

modify memory real short <ADDRESS> to <REAL>

software_breakpoints

sftw_btps

Used with the *display*, *modify*, and *list* commands to allow the software breakpoints to be displayed, modified, or listed to the printer or to a file that you name.

modify software_breakpoints clear <ADDRESS>

specify

specify

Used with multi-module systems to set up a run or trace for execution when the *execute* softkey is pressed.

specify run from <ADDRESS>

state

state

Used with *trace counting* commands to specify counting of states as opposed to counting time.

trace after data <VALUE> counting state status read

Command Line Message

Softkey Label

status

status

Used with *run*, *trace*, and *display* commands to reference status.

run from <G_L_SYM> until status <VALUE> occurs <#_TIMES>
trace after status data_acc
display trace status binary

step

step

Used to specify a single step. Can also specify a from condition, or a number of steps from an address or a global or local symbol.

step
step from <ADDRESS>
step <#_STEPS> from <ADDRESS>

stop_trace

stop_trc

Used to stop execution of a trace in process.

stop_trace

store

store

Used to store either a portion of memory or a trace buffer to a file that you name.

store memory <ADDRESS> thru <G_L_SYM> to <FILE>

thru

thru

Used with the *run*, *trace*, *display*, *modify*, *store*, and *list* commands to specify a particular range of information.

run until address range <ADDRESS> thru <ADDRESS> status data_acc
trace after address range <ADDRESS> thru <ADDRESS> status data_acc
display memory <ADDRESS> thru <ADDRESS>
modify memory <ADDRESS> thru <ADDRESS> to <VALLIST>

Command Line Message

Softkey Label

time

time

Used with *trace counting* commands to specify counting of time as opposed to counting states.

trace after status data_acc counting time

to

to

Used with the *modify* command to specify a value of information to which a memory location, register, or io_port is to be changed. Used with the *store* command to specify a destination for the information to be stored (to the printer or to a file).

modify <REGNAME> to <VALUE>

store memory <G_L_SYM> thru <ADDRESS> to <FILE>

trace

trace

Used to set up the analyzer trace specification. Also used with the *display* command to display the specified trace.

trace after <ADDRESS> counting time

display trace mnemonic count relative

trigger

trigger

Used with *trace break_on* commands to specify that at the trigger point the processor will break into the emulation monitor program.

trace break_on trigger

until

until

Used with the *run* command to establish an end of run condition. At the end of the run, the processor jumps to the emulation monitor program. This command is equivalent to a *trace <specification> break_on trigger* followed by *run*.

run until data <VALUE>

Command Line Message

Softkey Label

wait

wait

Used to specify a delay of a specific number of seconds, to specify a wait until measurement completes, or (default) until any key is pressed.

wait

wait <TIMER>

wait measurement_complete

word

word

Used with the *display memory* commands to specify a display in word mode rather than byte mode.

display memory absolute word

NOTES

Appendix D

STATUS, ERROR, AND SOFTKEY PROMPT MESSAGES

INTRODUCTION

This appendix contains a list of the status and error messages, and the softkey prompts and their corresponding messages. All these messages are displayed on the CRT as a result of the emulation software. An explanation of each message is also given. Table D-1 provides a list of the status messages, table D-2 provides a list of error messages, and table D-3 provides a list of the softkey prompts. Status messages are displayed on screen to provide an indication of operating status. Error messages are displayed on screen to indicate an improper operating condition or invalid entry on the command line. The softkey prompts are provided on the softkey label line to prompt you to input the required information.

Table D-1. Status Messages

Status Message	Meaning
Cannot halt processor	Processor cannot be halted. Hold acknowledge was never received after a halt instruction.
Execution aborted	The execution was stopped by using the <i>halt</i> softkey.
Execution in process	Execution is in process (for use with EBPP).
Halt	Displayed when a double bus fault occurs or when an external device strobes the halt line.
Bus grant	Displayed when the inactivity is caused by granting the bus to another device.
Memory load complete	Displayed when the absolute file you commanded to be loaded, with the <i>load <FILE></i> commands, has completed the loading process.
No memory cycles	Displayed when the hardware status cannot be used to determine the exact cause of the absence of bus activity. Possible cause may be: clock to processor has stopped or, during configuration of the emulator, an external clock was specified but no external clock is connected.
No monitor program	A command has been entered that requires the use of the monitor, but no monitor has been loaded. Load the monitor program absolute code and try again.

Table D-1. Status Messages (Cont'd)

Status Message	Meaning
reset	Displayed when the the target system is reset.
Reset	Displayed when the inactivity is caused by the emulation system or by the user setting the RESET line.
Running	Processor is running.
Running in monitor	Processor is running in the monitor.
Step complete	Single stepping completed - processor is back running in the monitor. This message will also be displayed if an analysis break or a memory error occurs before the specified number of steps has been completed.
Step in process	Processor is single stepping.
Trace complete	A trace has completed.
Trace in process	A trace is in process.
Trace stopped	The trace has been stopped, either by using the <i>stop_trc</i> softkey or by initiating a new trace while a trace is already in process.
wait	The wait message indicates that the 680XX is waiting for a DTACK memory response or other memory response. The condition may be terminated by asserting DTACK, VPA, or BERR; or if this isn't convenient, the 680XX can be reset by depressing the reset softkey.

Table D-2. Error Messages

Error Messages	Meaning
Access guarded mem. NNNNNNH (where NNNNNNH is address)	<p>You have entered a command which caused the processor to try to access address NNNNNNH, which is in guarded memory. This caused the processor to be forced into the emulation monitor program. If your command was legitimate, check your program and/or the memory map you set up during emulation configuration (refer to chapter 4) to find the problem.</p> <p>This message could also have been caused by your program trying to access guarded memory, such as a stack overflow into guarded memory. In this case, you will need to relocate your stack area.</p>
Break not allowed from reset	<p>You have attempted to force a break while the processor is in a reset condition. This is not allowed. Refer to the paragraph entitled "The Break Function and the Emulation Monitor" in chapter 7 for a further discussion of the break function.</p>
Command causes break- runs restricted to real time	<p>When you configured your emulator, you restricted runs to real time. To be executable, the command that you just entered requires a break to the emulation monitor program. Implied breaks are not allowed when runs are restricted to real time. You may wish to modify your emulation configuration at this point.</p>
Command not allowed- processor not in monitor	<p>In order to be executed, the command you have just entered requires the emulation monitor program. The emulation monitor program has been loaded, but was not accessible using the command just entered. This message may occur if the processor is reset.</p>
Global symbol <SYMBOL> in <PROGRAM:YOURID> <i>not found</i>	<p>The global symbol name you typed in, in answer to the softkey prompt <G_L_SYM>, was not found in the global symbols file created when you assembled your program. Press the <i>display glob_sym</i> softkeys and press RETURN. The global symbols available in your absolute file will be displayed on the screen. If you want to use the symbol you had typed in as a global symbol, go back to your program and identify it as such. Reassemble and relink your program(s) after this is done.</p>
Module <PROGRAM: YOURID> not found	<p>The emulation software cannot locate the program module you have identified in answer to the <G_L_SYM> softkey prompt. Type in the correct program module name.</p>

Table D-2. Error Messages (Cont'd)

Error Messages	Meaning
Local symbol <SYMBOL: PROGRAM:YOURID> not found	The local symbol name you typed in, in answer to the softkey prompt <G_L_SYM>, was not found in the program file you named. Press the <i>display loc_sym</i> softkeys, type in your program name, and press (RETURN) . The local symbols available in that program will be displayed on the screen.
Symbols not accessible, absolute file not loaded	After you entered the emulation session, you did not load your absolute file into memory. Press the <i>load</i> softkey, type in the name of your absolute file, and press the key.
Syntax error	Displayed whenever you attempt to enter an invalid command and press the (RETURN) key.
Write to ROM NNNNNNH (where NNNNNNH is address)	When you configured your emulator, you chose to break the processor on an attempt to write to ROM. The command you just entered, for the program you are running, caused the processor to try to write to ROM. This caused the processor to be forced into the emulation monitor program. If your command was legitimate, check your program and/or the memory map you set up during emulation configuration (refer to chapter 4) to find the problem.

Table D-3. Softkey Prompt Messages

Softkey Prompt Messages	Status Line Message and Meaning
<ADDRESS>	<p><i>Address expression</i></p> <p>Any valid address within the absolute file loaded into emulation or user memory.</p>
<ADDRESS>	<p><i>Address expression w/ optional don't cares (X)</i></p> <p>Any valid address (with optional don't cares "X's") within the absolute file loaded into emulation or user memory.</p>
<ADR_LST>	<p><i>Addresses or ranges separated by commas</i></p> <p>Any valid addresses, or address ranges separated by commas, within the absolute file loaded into emulation or user memory.</p>
<CMDFILE>	<p><i>System command file name</i></p> <p>Prompts you to enter the name of a command file containing valid HP 64000 operating system or emulation commands. Used to link files, configure the emulator, run measurements automatically, etc.</p>
<FILE>	<p><i>Absolute file name</i></p> <p>The name of the absolute file to be loaded from the HP 64000 system to emulation or user memory.</p>
<FILE>	<p><i>File name</i></p> <p>When used with the <i>list</i> command, <FILE> is the name you give to a file to which you are keeping information (i.e., memory, trace, registers, global symbols, local symbols, software breakpoints, and i/o ports).</p>
<FILE>	<p><i>Relocatable file name</i></p> <p>Used with the <i>library</i> command to identify a relocatable file to be added to a relocatable library file.</p>

Table D-3. Softkey Prompt Messages (Cont'd)

Softkey Prompt Messages	Status Line Message and Meaning
<FILE>	<p><i>Trace store file</i></p> <p>Used with the <i>store</i> command to specify the name of a file in which you want to store trace information.</p>
<G_L_SYM>	<p><i>Global or local symbol</i></p> <p>Prompts you for a global or local symbol. If a local symbol is supplied in answer to the prompt, it is necessary to insert a colon (:) and the module name containing the local symbol.</p>
<LINE_#>	<p><i>"#" followed by source program line number</i></p> <p><LINE_#> represents the line number of a Pascal or C statement in a source program. If the specified <LINE> contains only comments (no executable code), emulation software will associate the line number with the first line containing executable code following it. Any comment lines preceding the first line of executable code in a procedure or function are not recognized. All lines in the specified range must be contained within a single module. This module may be a procedure or function in Pascal or a function in C, or the main program block.</p>
<NUMBER>	<p><i>Number value</i></p> <p>Prompts you for the source line <NUMBER> to be inserted in answer to the <LINE_#> prompt given above.</p>
<PARMS>	<p><i>Parameter(s)</i></p> <p>The parameters passed to a command file.</p>
<REAL>	<p><i>Real number</i></p> <p>Used with the <i>modify memory real</i> commands to change or add a real number in a memory location. The real number consists of an interger plus a decimal fraction and an exponent (refer to appendix A for the definition of real numbers).</p>
<REGNAME>	<p><i>Name of processor register</i></p> <p>The register name (i.e., A0, D2, D0, etc.). Refer to appendix B for the register names and format).</p>

Table D-3. Softkey Prompt Messages (Cont'd)

Softkey Prompt Messages	Status Line Message and Meaning
<TIMER>	<p><i># seconds for delay. Default: wait on any keystroke</i></p> <p>Used with the <i>wait</i> command to define the number of seconds to delay before the next operation is executed. If no delay time is specified, a keyboard key must be pressed to cause action to resume.</p>
<VALLIST>	<p><i>List of values</i></p> <p>Used with <i>modify memory word</i> or <i>byte</i> to modify the contents of a selected memory location(s).</p>
<VALUE>	<p><i>Expression, w/ optional don't cares (X)</i></p> <p>Prompts you for a value to be inserted to identify an address, status, or data value with optional "don't cares (X's)".</p>
<#_TIMES>	<p><i>Number of occurrences of trigger event</i></p> <p>Used with the <i>occurs</i> command to define the number of occurrences that are to happen before the specified trigger event will be executed.</p>
<#_STEPS>	<p><i>Number of processor steps</i></p> <p>Used with the <i>step</i> command to define the number of processor steps to be executed. The result is then displayed.</p>

NOTES

Appendix E

SIMULATED I/O ERROR CODES

The general definitions for the simulated I/O error codes are listed in table E-1.

When a request by the user program cannot be executed, the applicable error code is returned by the HP 64000 system to the appropriate control address.

Table E-1. Simulated I/O Error Codes - General Definitions

DECIMAL CODE #	(Hex)	MEANING
00		No error - successful operation
01		End of file
02		Invalid disc
03		File not found
04		File already exists
05		No disc space available
06		No directory space available
07		File is Corrupt (bad linkage)
08		Cannot read/write assigned memory
09		Request not allowed
10	(A)	Invalid file type
11	(B)	Invalid row or column no.
12	(C)	Invalid record length
13	(D)	Invalid display character >OFOH
14	(E)	While in simulated display I/O or simulated keyboard I/O, the HP 64000 "simulate" soft key was pressed to exit simulate I/O. All open files are closed.
15	(F)	Error in new disc file name when attempting to change a disc file name. First character in file name limited to capital letters A through Z. Second and following characters may contain capital and lower case letters, numerals 0 through 9, underlines, and only if required to fill in the last byte of the last word, a blank is used.

NOTES

Appendix F

RESTRICTIONS WHEN USING THE EXECUTION MODE OF ANALYSIS

EFFECTS OF EXECUTION_DATA MODE

The emulator holds states in a queue for up to two bus cycles after they were executed before placing them on the analysis bus. The emulator uses this delay period to see if an interrupt flushes an instruction from the microprocessor pipeline. Any instruction that is flushed from the pipeline will not be placed on the analysis bus.

Because of the queue depth, the emulator may not be able to present certain trace information on the analysis bus in the most logical order. The following analysis order may occur:

```
Opcode 1
Postword   (associated with Opcode 1)
Data       (associated with Opcode 1)
Data       (associated with Opcode 1)
Postword   (associated with Opcode 1)
Opcode 2
```

Note the location of the second Postword.

Certain run-time conditions make it impossible to determine whether a branch was taken or not. Under these conditions, the emulator places the code on the analysis bus without knowing whether that code was executed or not. The conditions that cause this uncertainty are as follows:

When the next program address is "present address + 2", this usually indicates "branch not taken". There are cases where a branch is taken to "present address + 2":

1. A one-word branch-around instruction. (i.e. RTE, RTD, RTS, TRAP, JSR, JMP, BSR, BRA, BCC)
2. A trap to an address which is "present address + 2",

The program steps listed below show an example of a one-word branch-around. The "divide by zero" trap is taken at "L1", and the trap vector points to "L2". The instruction "MOVE.L D0,D1" is placed on the analysis bus even though the trap was taken.

```
                MOVE.L    #0,D1
L1              DIVS      D1,D0
                MOVE.L    D0,D1
L2              CLR.L     [A0]
```

PROGRAM COUNTER RELATIVE INSTRUCTIONS

When processing Program Counter Relative instructions while in the `bus_cycle_data` mode, the 680XX processor tags opcodes and data as program words. To differentiate between opcodes and data, the emulator uses the following algorithm:

If the opcode is Program Counter Relative, then:

1. If the next program address is equal to the "previous address + 2", the word is a program word.
2. If next program word address is not equal to the "previous address + 2", the word is a data operation.

To insure that this works, make sure that your PC-relative data is at least four bytes from the end of the instruction that accesses the data. See the example in figure F-1.

L2	MOVE	#5E30H,D0		L2	MOVE	#5E30H,D0	
	MOVE	[A7]+,TEMP1			MOVE	[A7]+,TEMP1	
	MOVE.L	L1[PC],D0			MOVE.L	L1[PC],D0	
	BEQ	L2			BEQ	L2	
L1	DATA	<-----	NOTE THAT THE DATA	L1	DATA	<----	NOTE THAT THE
			IS ONLY 2 BYTES				NOP ADDS THE
			AWAY FROM THE INSTRUCTION.				ADDITIONAL 2 BYTES
							THAT ARE NEEDED.

Figure F-1. Example One Word Branch-Around Code

If the PC-relative data is not at least four bytes away, the data will be interpreted as program code by the de-pipelining hardware and the trace data will be invalid for an undetermined amount of time.

USING THE "TRACE FUNCTION" OF THE 680XX

The 680XX processor allows instruction-by-instruction tracing. While in the trace mode, the processor asserts the "T" bit in its status register, but it does not set any status bits on its output pins. There is no direct way to differentiate between program cycles and stack operations during the trace mode. Therefore, if the processor trace mode is enabled, the trace data (execution_mode only) will not be valid.

USING UNIMPLEMENTED INSTRUCTIONS

The 680XX processor has unimplemented instruction traps which you can use to build your own instructions. The execution mode opcode synchronizer cannot detect your unimplemented instructions so it will place these instructions on the analysis bus. If you create an unimplemented instruction that has the lower 6 bits equal to 11101X, the synchronizer may detect this instruction as a Program Counter Relative opcode and analysis may interpret your program words as data words when the trap is taken.

NOTE

There are two 'illegal opcodes' that are specifically designed by the manufacturer for you to customize. These are :

AXXXH and FXXXH

These two instructions are recognized by the emulation system and will not cause any problems.


DMA CYCLES USING EXECUTION_MODE

DMA cycles will appear in a trace list ahead of the bus cycle in which the bus was granted. This is because DMA cycles bypass the de-pipeline hardware of the emulator.


NOTES

SERVICE INFORMATION CONTENTS

APPENDIX G: SERVICE INFORMATION

Section G1: Installing Your Emulation Hardware and Software	G1-1
OVERVIEW	G1-1
INTRODUCTION	G1-1
PRE-INSTALLATION INSPECTION	G1-4
REQUIRED EQUIPMENT, RECOMMENDED ADDITIONAL EQUIPMENT, AND ADDITIONAL OPTIONAL EQUIPMENT	G1-4
INSTALLING EMULATION SYSTEM HARDWARE	
INTO AN HP 64100 LOGIC DEVELOPMENT STATION	G1-6
CONFIGURATION OF THE BOARDS IN THE STATION	G1-6
CONFIGURATION OF THE MEMORY BOARDS	G1-7
INSTALLATION INSTRUCTIONS	G1-8
INSTALLING EMULATION SYSTEM HARDWARE INTO	
AN HP 64110 LOGIC DEVELOPMENT STATION	G1-15
CONFIGURATION OF THE BOARDS IN THE STATION	G1-15
CONFIGURATION OF THE MEMORY BOARDS	G1-15
INSTALLATION INSTRUCTIONS	G1-15
 INSTALLING THE EMULATION PROBE CABLE	
CONNECTOR INTO THE TARGET SYSTEM SOCKET	G1-20
Target System Microprocessor Connector Compatibility	G1-21
Installing into a DIP Socket	G1-21
Installing into a PGA Socket	G1-22
TURNING ON THE DEVELOPMENT STATION	G1-23
LOADING EMULATION SYSTEM SOFTWARE	G1-24
Backing Up Your Software	G1-24
Loading Software in Clustered Stations Configuration	G1-24
Configuring A Flexible Disc For Stand-Alone Operation	G1-25
REMOVING EMULATION SOFTWARE	
FROM THE OPERATING SYSTEM	G1-29
PERFORMING OPERATION VERIFICATION	G1-29
OPERATING THE EMULATION SYSTEM IN STAND-ALONE CONFIGURATION ..	G1-30
REMOVING THE EMULATOR CONTROL BOARD	G1-31
REPACKAGING FOR SHIPMENT	G1-32
Original Packaging Materials	G1-32
Other Packaging Materials	G1-32
 Section G2: Performance Verification and Troubleshooting	 G2-1
INTRODUCTION	G2-1
PERFORMANCE VERIFICATION	G2-1
TROUBLESHOOTING	G2-8
WHAT IS AN EMULATION SYSTEM?	G2-8
Scope	G2-8
Emulation Overview	G2-8

SERVICE INFORMATION CONTENTS (CONT'D)

Emulation Subsystem Overview	G2-10
General	G2-10
Emulator Pod	G2-10
Emulation Control Card	G2-10
Internal Analysis	G2-10
Emulation Memory Controller	G2-10
Emulation Memory	G2-10
Emulator Block Diagrams	G2-10
Emulator Control Card	G2-10
Emulator Pod	G2-14
TROUBLESHOOTING FLOW CHART	G2-17
Section G3: Adjustments	G3-1
Section G4: Replaceable Parts	G4-1
INTRODUCTION	G4-1
EXCHANGE ASSEMBLIES	G4-1
ABBREVIATIONS	G4-1
REPLACEABLE PARTS LIST	G4-1
ORDERING INFORMATION	G4-2
DIRECT MAIL ORDER SYSTEM	G4-2
Section G5: Removal and Replacement Procedures	G5-1
EMULATOR POD DISASSEMBLY	G5-1
EMULATOR POD REASSEMBLY	G5-5
Section G6: Electrical and Environmental Characteristics	G6-1
USER INTERFACE	G6-1
POWER SUPPLY REQUIREMENTS	G6-1
 EMULATION PROBE INSTALLATION CAUTION STATEMENT	G6-2
OPERATING ENVIRONMENT	G6-3
STORAGE AND SHIPMENT	G6-3
PERFORMANCE CHARACTERISTICS	G6-3
Section G7: Electromagnetic Compatibility	G7-1

SERVICE INFORMATION CONTENTS (Cont'd)

Section G8: Schematic Diagrams	G8-1
INTRODUCTION	G8-1
THEORY OF OPERATION	G8-1
68000 PROCESSOR BOARD A1	G8-1
68008 PROCESSOR BOARD A1	G8-4
TIMING BOARD A2	G8-7
Data Bus Transmission and Control	G8-7
Pod Registers	G8-9
Pod Index Register (U1L)	G8-9
Pod Control Register (U2N, U1P)	G8-10
Break Address Register (U3K, U4K, U3L, U4L)	G8-11
Break Method	G8-12
DTACK Circuitry	G8-13
Wait Detection Circuitry	G8-13
Generation of LDTACK	G8-13
Generation of XDTACK	G8-14
DMA Circuitry	G8-14
DMA Arbitration	G8-14
Emulation Halt (EHALT)	G8-14
DMA Tagging	G8-14
Tristating the Bus	G8-15
MNEMONICS	G8-15
Index	G1-1

SERVICE INFORMATION ILLUSTRATIONS

Figure G1-1. HP 64100 Key Features	G1-2
Figure G1-2. HP 64110 Key Features	G1-3
Figure G1-3. RFI Ground Bracket Assembly Parts	G1-10
Figure G1-4. RFI Ground Bracket Installation (64100A)	G1-10
Figure G1-5. Emulation Control Board Connections	G1-12
Figure G1-6. Ground Bar Clamp Installation (64100A)	G1-14
Figure G1-7. HP 64110 Installation Details	G1-17
Figure G1-8. Ground Bar Clamp Installation (64110A)	G1-19
Figure G1-9. HP 64110 Memory, Emulation, and Intermodule Bus Cabling	G1-19
Figure G1-10. Installing the Emulation Probe into a DIP Socket	G1-21
Figure G1-10. Installing the Emulation Probe into a PGA Socket	G1-22
Figure G1-12. I/O Bus Configuration Display	G1-22
Figure G1-13. Option Test Display	G1-30
Figure G2-1. Awaiting Command Status	G2-4
Figure G2-2. Option_Test Card Slot Listing	G2-4
Figure G2-3. Performance Verification	G2-5
Figure G2-4. Pod Register Tests (Static)	G2-5
Figure G2-5. Pod Functionality Test	G2-6

SERVICE INFORMATION ILLUSTRATIONS(Cont'd)

Figure G2-6. Analysis Stimulus Test	G2-6
Figure G2-7. Analysis Tests	G2-7
Figure G2-8. Emulation Subsystem	G2-9
Figure G2-9. Emulator Control Card Block Diagram	G2-13
Figure G2-10. Emulator Pod Block Diagram	G2-15
Figure G2-11. Emulator Troubleshooting Flowchart	G2-19
Figure G4-1. Emulator Control Card Parts Locator	G4-3
Figure G4-2. Emulator Pod Illustrated Parts Breakdown	G4-4
Figure G5-1. Emulator Pod Side View	G5-2
Figure G5-2. Emulator Pod Top View	G5-3
Figure G5-3. Emulator Pod Bottom View	G5-3
Figure G5-4. Emulator Pod - Inside Top Cover	G5-4
Figure G5-5. Pod Bus and Board Interconnect Cables	G5-4
Figure G6-1. Read Cycle Timing Diagram (68000)	G6-10
Figure G6-2. Write Cycle Timing Diagram (68000)	G6-11
Figure G6-3. Bus Arbitration Timing Diagram - Idle Bus Case (68000) sssssssssssss	G6-12
Figure G6-4. Bus Arbitration Timing Diagram - Active Bus Case (68000) sssssssssss	G6-13
Figure G6-5. Bus Arbitration Timing Diagram - Multiple Bus Requests (68000) sssssss	G6-14
Figure G6-6. Read Cycle Timing Diagram (68008)	G6-17
Figure G6-7. Write Cycle Timing Diagram (68008)	G6-18
Figure G6-8. MC68008 to M6800 Peripheral Timing Diagram Best Case (68008)	G6-19
Figure G6-9. MC68008 to M6800 Peripheral Timing Diagram Worst Case (68008)	G6-20
Figure G6-10. Bus Arbitration Timing Diagram - Idle Bus Case (68008) sssssssssssss	G6-21
Figure G6-11. Bus Arbitration Timing Diagram - Active Bus Case (68008) sssssssssss	G6-22
Figure G6-12. Bus Arbitration Timing Diagram - Multiple Bus Requests (68008; 52-Pin Version	
Figure G8-1. 68000 Emulator Pod Processor Board Schematic	G8-35
Figure G8-2. 68008 Emulator Pod Processor Board Schematic	G8-41
Figure G8-3. Emulator Pod Timing Board Schematic	G8-47

SERVICE INFORMATION TABLES

Table G1-1. RFI Ground Bracket Assembly Parts List	G1-9
Table G4-1. Reference Designators and Abbreviations	G4-7
Table G4-2. Control Card Replaceable Parts	G4-8
Table G4-3. 68000 Emulator Pod Replaceable Parts	G4-9
Table G4-4. 68008 Emulator Pod Replaceable Parts	G4-10
Table G6-1. Timing Comparison - MC68000R12 vs 12.5 Mhz Emulator	G6-4
Table G6-2. Timing Comparison - MC68008L10 vs 10.0 Mhz Emulator	G6-6
Table G6-3. Electrical Specifications - Read and Write Cycles (68000)	G6-8
Table G6-4. Electrical Specifications - Read and Write Cycles (68008)	G6-15
Table G8-1. Mnemonics	G8-16

SAFETY SUMMARY

The following general safety precautions must be observed during all phases of operation, service, and repair of this instrument. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the instrument. Hewlett-Packard Company assumes no liability for the customer's failure to comply with these requirements.

GROUND THE INSTRUMENT.

To minimize shock hazard, the instrument chassis and cabinet must be connected to an electrical ground. The instrument is equipped with a three-conductor ac power cable. The power cable must either be plugged into an approved three-contact electrical outlet or used with a three-contact to two-contact adapter with the grounding wire (green) firmly connected to an electrical ground (safety ground) at the power outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE.

Do not operate the instrument in the presence of flammable gases or fumes. Operation of any electrical instrument in such an environment constitutes a definite safety hazard.

KEEP AWAY FROM LIVE CIRCUITS.

Operating personnel must not remove instrument covers. Component replacement and internal adjustments must be made by qualified maintenance personnel. Do not replace components with the power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

DO NOT SERVICE OR ADJUST ALONE.

Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

DO NOT SUBSTITUTE PARTS OR MODIFY INSTRUMENT.

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification of the instrument. Return the instrument to a Hewlett-Packard Sales and Service Office for service and repair to ensure that safety features are maintained.

DANGEROUS PROCEDURE WARNINGS.

Warnings, such as the example below, precede potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

WARNING

Dangerous voltages, capable of causing death, are present in this instrument. Use extreme caution when handling, testing, and adjusting.

SAFETY SYMBOLS

General Definitions of Safety Symbols Used on Equipment or in Manuals.



Instruction manual symbol: the product is marked with this symbol when it is necessary for the user to refer to the instruction manual in order to protect against damage to the instrument.



Indicates dangerous voltage (terminals fed from the interior by voltage exceeding 1000 volts must be so marked).



OR



Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.



Low-noise or noiseless, clean ground (earth) terminal. Used for a signal common, as well as providing protection against electrical shock in case of a fault. A terminal marked with this symbol must be connected to ground in the manner described in the installation (operating) manual, and before operating the equipment.



OR



Frame or chassis terminal. A connection to the frame (chassis) of the equipment which normally includes all exposed metal structures.



Alternating current (power line).



Direct current (power line).



Alternating or direct current (power line).

WARNING

The **WARNING** sign denotes a hazard. It calls attention to a procedure, practice, condition or the like, which, if not correctly performed, could result in injury or death to personnel.

CAUTION

The **CAUTION** sign denotes a hazard. It calls attention to an operating procedure, practice, condition or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product.

NOTE :

The **NOTE** sign denotes important information. It calls attention to procedure, practice, condition or the like, which is essential to highlight.

Appendix G

INSTALLATION AND SERVICE INFORMATION

Section G1

INSTALLING YOUR EMULATION HARDWARE AND SOFTWARE

OVERVIEW

Section G1 will:

- o Provide pre-installation inspection instructions.
- o Provide a complete list of emulation system hardware, software, and manuals.
- o Show you how to install an RFI ground bracket.
- o Show you how to install the emulation system hardware.
- o Show you how to install the emulation probe cable.
- o Show you how to turn on the development station.
- o Show you how to load the emulation system software.
- o Explain how to customize a flexible disc for an HP 64000 station in stand alone configuration.
- o Show you how to remove emulation system software modules.
- o Provide reminders for stand alone operating configuration.

INTRODUCTION

If you are installing your HP 64000 Logic Development System, including peripherals, as a new installation, refer to the HP 64000 Installation and Configuration Manual for instructions concerning the installation of the system work stations, disc, and printer. Also, refer to "Pre-installation Inspection" given in this section. After you have done these, install the emulation system as instructed later in this section.

NOTE

All references to the 68000 microprocessor and the HP Model 64243 Emulator in this service appendix are equally applicable to the 68008 microprocessor and the HP Model 64244 Emulator respectively, unless otherwise noted.

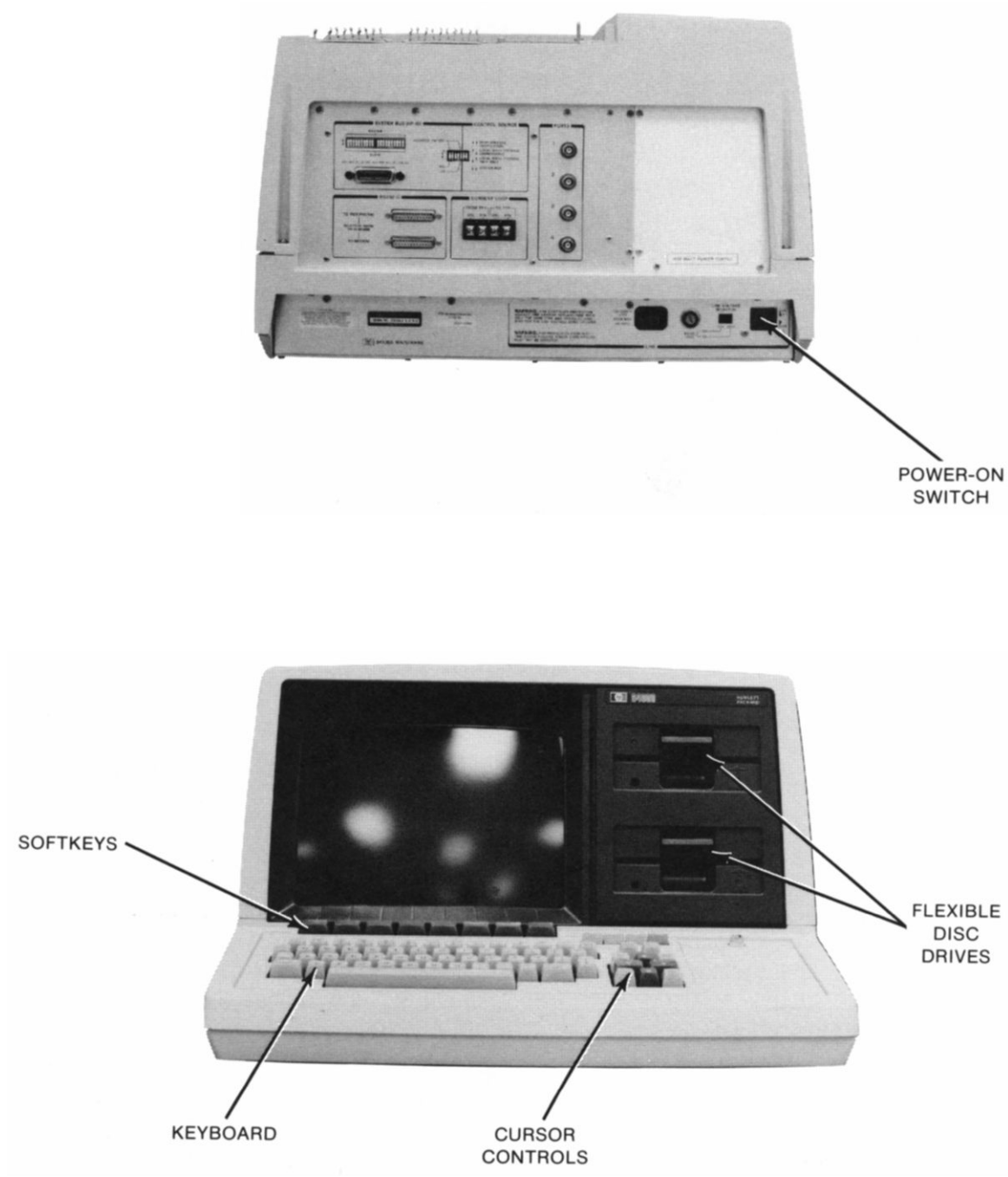


Figure G1-1. HP 64100 Key Features

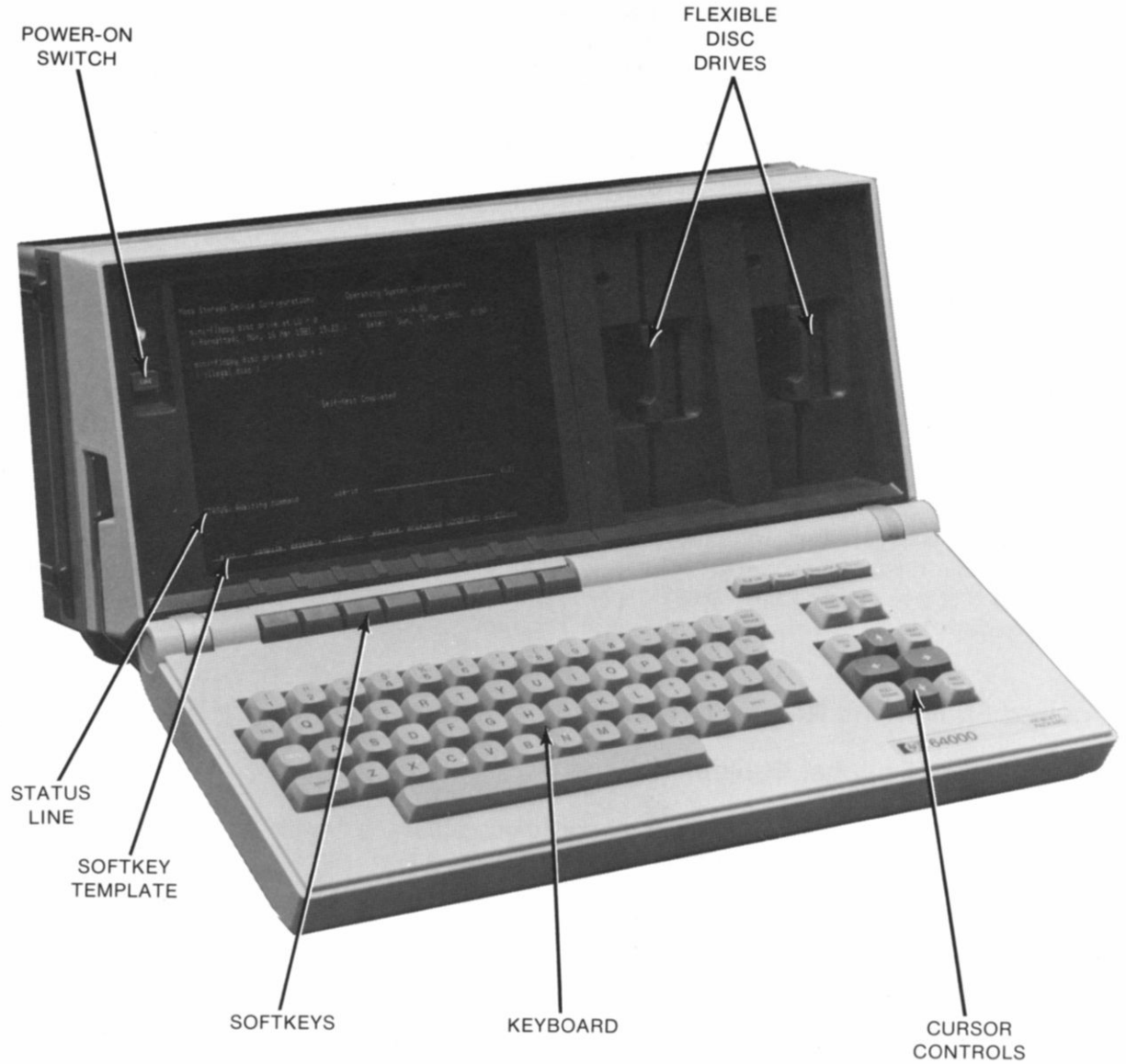


Figure G1-2. HP 64110 Key Features

If you have just received your 68000 Emulation system, check to see that you have all of the hardware, software, and manuals that are listed in this section. The hardware must be installed in the HP 64000 development station. You have a choice of loading the software to hard disc or using the software from a flexible disc. In either case, it is recommended that you generate back-up software as soon as your system is operational.

Figures G1-1 and G1-2 identify some key features of the HP 64100 and HP 64110 Stations, respectively. The identifying labels used in these figures are used throughout this manual. Note the location of the power switches. For more information on the hardware configuration, refer to the Installation and Configuration Manual.

If you are using the emulator for the first time, you should perform the checks listed in this section under the heading "Performing Operation Verification".

PRE-INSTALLATION INSPECTION

Unpack all of the emulation system circuit boards, cables, pod, and related equipment. Carefully inspect the equipment for damage that may have occurred during shipping. If any damage is found, please contact your nearest Hewlett-Packard Sales/Service Office as soon as possible.

Compare the parts received with the list of items that you ordered from the list of required equipment given below (shown in Chapter 1, figure 1-1 of the Operating Section) to assure that all of the items that you ordered have been shipped. If any equipment is missing, please contact your nearest Hewlett-Packard Sales/Service Office as soon as possible.

Required Equipment, Recommended Additional Equipment, and Additional Optional Equipment

The emulation system consists of four hardware modules plus the flexible disc emulation software package and technical manuals. The four hardware modules are:

- o Emulation Subsystem
- o Emulation Memory Subsystem
- o Internal Analyzer
- o Emulation/Memory/Analysis Bus Cable(s)

The emulation system hardware listed above consists of circuit boards which are installed in the development station card cage, the emulation probe (pod) which is connected by cabling to the development station, and bus cables to connect the boards. These parts, along with the software and technical manuals required, are shown in Chapter 1, figure 1-1 of the Operating Section and are also listed below.

NOTE

You may already have some of the equipment listed if you have purchased other Hewlett-Packard equipment. For example, if you have previously purchased other emulation systems, you may already have an emulation memory subsystem, and possibly, the optional internal analysis board. In those cases, disregard the parts listed below that do not concern your situation, but do follow the installation instructions for the equipment that you have since the instructions are tailored to the 68000 emulation system.

Make sure you have the parts listed below that you have ordered and that no damage has occurred to any of the parts.

REQUIRED EQUIPMENT

Emulation Control Board (HP Part No. 64243-66501 or 64244-66501)

Emulation Probe (68000 Or 68008 Pod) with Microprocessor Connector (HP Model 64243 or 64244)

Emulation Memory Control Board (HP 64155A)

Emulation Memory Board (HP 64152/153/154A/B or HP 64161/162/163A: 1 required; 2-8 optional)

Emulation/Memory/Analysis Bus Cable (HP 64960A: 1 required for Memory Control Board)

Flexible Disc -68000 or 68008 Emulation System Software (HP Part No. 64243-12000 or 64244-12000)

Operating Manual with Service Appendix (HP Part No. 64243-90901)

RECOMMENDED ADDITIONAL EQUIPMENT

Wide Internal Analysis Board (64302A)

Emulation/Memory/Analysis Bus Cable (64960A: 2 required for Wide Internal Analysis Board)

Flexible Disc 68000/08/10 Assembler/Linker Software (64845AF)

ADDITIONAL OPTIONAL EQUIPMENT

Flexible Disc 68000/08/10 Pascal Compiler Software (64815AF)

Flexible Disc 68000/08/10 C Compiler Software (64819AF)

Software Performance Analyzer (64310A)

Logic State/Software Analyzer (64620) with Emulation Bus Preprocessor
(64304A)

68000 Dual In-Line Pin (DIP) Probe Cable (64243-61601)

68000 Pin Grid Array (PGA) Probe Cable (64243-61602)

INSTALLING EMULATION SYSTEM HARDWARE INTO AN HP 64100 LOGIC DEVELOPMENT STATION

WARNING

Any installation, servicing, adjustment, maintenance, or repair of this product must be performed only by qualified personnel. Make sure power is off prior to performing any of the installation instructions given below.

NOTE

The following installation steps assume the installation of a complete system (maximum memory and internal analysis). Disregard procedural steps for equipment you do not require or have.

Configuration of the Boards in the Station

While the emulation and analysis circuit boards may be installed in any card slot in the station chassis, mechanical considerations (i.e.; cabling) make the following card groupings most convenient:

For single module systems:

slot 9 Internal Analysis board
slot 8 Emulation Control board
slot 7 Memory Control board
slot 6 Memory board
slot 5 Memory board (if used)
slot 4 Memory board (if used)
slot 3 Memory board (if used)
slot 2 Memory board (if used)
slot 1 Memory board (if used)
slot 0 Memory board (if used)

For two emulation module systems:

slot 9 Internal Analysis board
slot 8 Emulation Control board
slot 7 Memory Control board
slot 6 Memory board
slot 5 Memory board (if used)
slot 4 Internal Analysis board
slot 3 Emulation Control board
slot 2 Memory Control board
slot 1 Memory board
slot 0 Memory board (if used)

When an emulator is used in a system with state or timing analyzers, either half of the above ordering may be used. As noted before, the boards may be installed in any order. It is important, however, that the boards be positioned so that the emulation control and intermodule bus cables do not cross.

Configuration of the Memory Boards

Each memory board has space for four rows of memory chips. The HP 6416XA memory boards may contain from 16K to 64K words (32K to 128K bytes) of random access memory (RAM). The HP 6415XA memory boards may contain from 4K to 16K words (8K to 32K bytes) of random access memory (RAM). The number of rows loaded on the board determines the amount of memory as follows:

NOTE

The HP 6415XA memory boards can no longer be purchased. However, they are mentioned here to cover the possibility that you may have purchased them previously to support earlier HP equipment purchases.

- HP 64161A - 128K bytes - all four rows loaded
- HP 64162A - 64K bytes - bottom two rows loaded
- HP 64163A - 32K bytes - bottom row only loaded
- HP 64152A - 32K bytes - all four rows loaded
- HP 64153A - 16K bytes - bottom two rows loaded
- HP 64154A - 8K bytes - bottom row only loaded

The number of bytes available on the board is also indicated on the ejector tab located on the top edge of the board.

6416XA MEMORY BOARDS. There are five 16-pin sockets located near the top edge (on the component side) of each memory board. Each half of the first four sockets is labeled, respectively, in 16K word ranges. Note that the numbers indicate words of memory (not bytes, as indicated on the ejector tab). The fifth socket contains jumpers which are used to prevent memory overlap by defining the memory partition of each memory board and to also allow mixing of Model HP 6416X and earlier Model HP 6415X series memory boards in the same emulation memory system. For further information regarding jumper configuration, refer to the Model HP 64161A/162A/163A Emulation Memory Service Manual. The factory-shipped position of the jumper in the fifth socket is as follows:

- HP 64161A - 8-pin jumper in the lower half of the fifth socket connecting R1 0-16K, R2 16-32K, R3 32-48K, and R4 48-64K.
- HP 64162A - Two 2-pin jumpers in the lower half of the fifth socket connecting R1 0-16K and R2 16-32K.
- HP 64163A - One 2-pin jumper in the lower half of the fifth socket connecting R1 0-16K.

If memory board types are not being mixed, the jumper in the fifth socket must be left in the factory-shipped position. If both HP 6416X and HP 6415X memory board types are to be used in your system, refer to the HP 64161A/162A/163A Emulation Memory Service Manual for configuration details.

6415XA MEMORY BOARDS. The HP 6415XA Emulation Memory boards are no longer available. If you have these boards available from a previous purchase, and wish to use them for supporting your 68000 emulation system you will need to refer to the HP 64152A/64153A/64154A Emulation Memory Service Manual for configuration details concerning these memory boards.

Installation Instructions

WARNING

Read the safety summary at the front of this manual before installation or removal of the Emulation Subsystem.

CAUTION

Power to the HP 64000 Development Station must be removed before installation or removal of option cards (emulation, etc.) to avoid damage to the option cards and the development station.

GENERAL. Installation of the circuit boards is accomplished by sliding each circuit board into the circuit board guide slots, with the component side of the board facing the front of the chassis. Align the connector at the bottom of the board with the motherboard connector at the bottom of the card cage, then apply a downward pressure until the board is seated in the motherboard connector. Be sure the ejector handles are in their full horizontal position when the board has reached its full downward travel.

Install the RFI Ground Bracket Assembly first. Because of cable restrictions, it is recommended that the internal analysis board be installed next, followed by the Emulation Control board, the Memory Controller, and then the memory boards.

To install the Emulation System and related equipment, proceed as follows:

- a. Turn OFF power to the HP 64100 Development Station. (See figure G1-1 for the location of the power switch.)

- b. Loosen the two hold-down screws on the top, rear of the card cage access cover, and remove the cover.
- c. Install the RFI Ground Bracket Assembly.

An RFI Ground Bracket must be installed in the HP 64100A development station before installing the Emulator Control Card and the Emulator Pod. The purpose of this ground bracket is to connect the emulator pod cable shield braid to earth ground. This effectively prevents the emission of radio frequency interference components from the pod cables.

The RFI ground bracket parts are included with the Emulator Pod. For convenience, a photograph depicting all the parts is reproduced here (see figure G1-3); Table G1-1 gives a listing of parts included in the ground bracket assembly (A4 for the 68000 Emulator Pod).

NOTE

If no ground bracket is installed in the development station, then the old U-shaped bracket on the rightmost side of the development station top cover (as viewed from the rear) must be removed. Loosen the two screws; remove the bracket; discard the bracket and the screws (they will not be needed for the RFI bracket).

Table G1-1. RFI Ground Bracket Assembly Parts List

REF DES	DESCRIPTION	PART NO.
A4	GROUND BRACKET ASSY	64100-62102
A4MP1	GROUND BAR CLAMP	1531-0273
A4MP2	SCREW 4-40 3/4"	2200-0151
A4MP3	SCREW 6-32 3/8"	2360-0117
A4MP4	SCREW 6-32 1"	2360-0129
A4MP5	NUT 6-32	2420-0001
A4MP6	WASHER	3050-0235
A4MP7	RFI GROUND BRACKET	64100-01205

Install the ground bracket on the HP 64100A Development Station, proceed as follows:

Place the RFI bracket on the top rear cover of the development station in the rightmost cable position (as viewed from the rear of the station) with the threaded stud on the bracket pointing upwards and the U-shaped portion of the bracket to the inside of the card cage. See figure G1-4 for details.

Thread two screws (A4MP3) into the RFI bracket from the outside of the development station rear panel sheet metal. Tighten the screws firmly.

To remove the bracket, reverse the installation procedure.

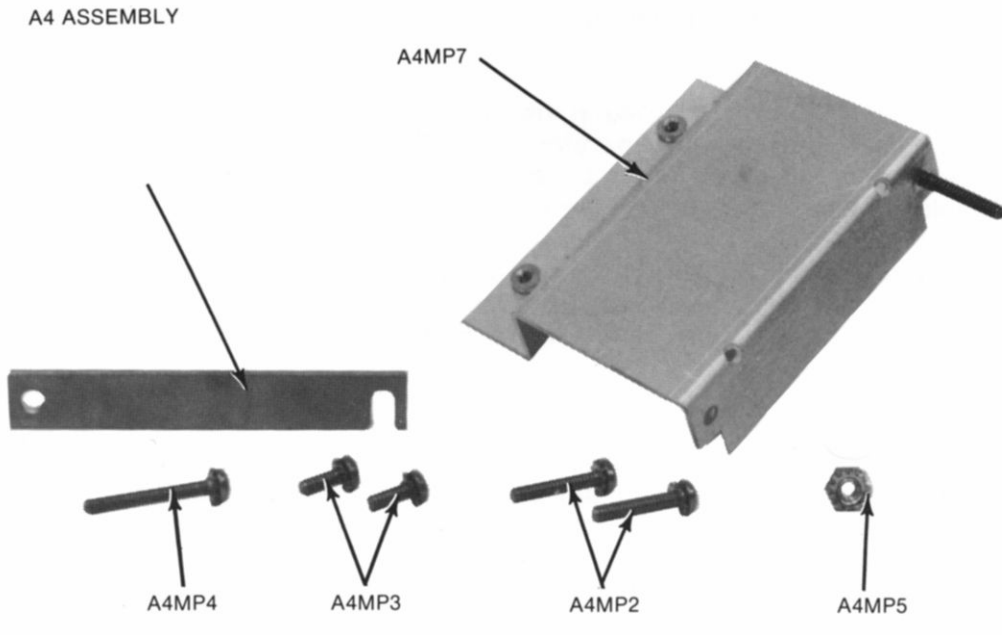


Figure G1-3. RFI Ground Bracket Assembly Parts

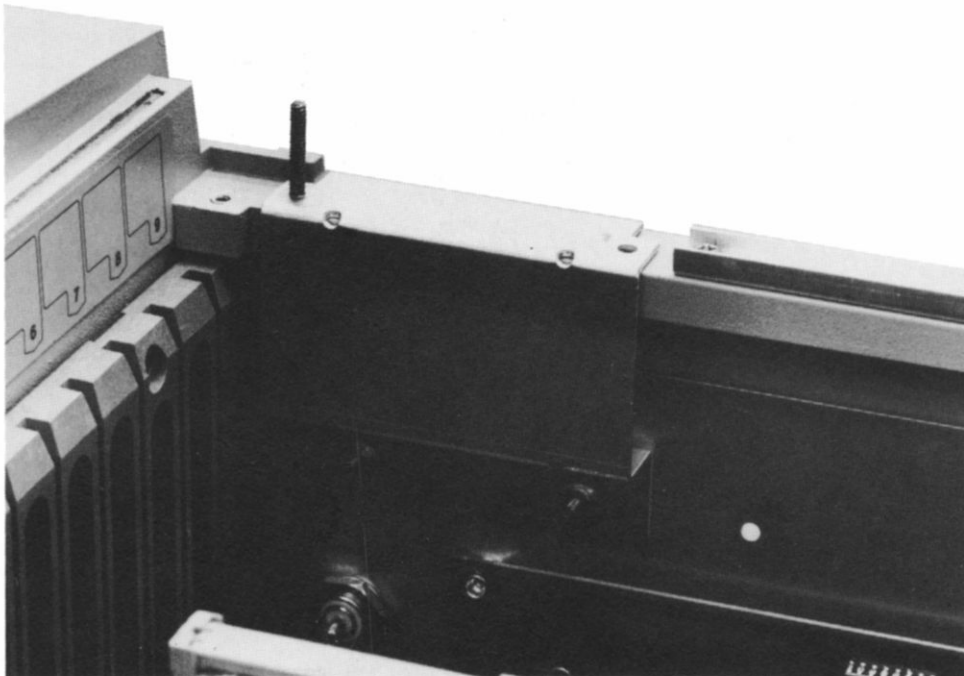


Figure G1-4. RFI Ground Bracket Installation (HP 64100A)

d. Install the Internal Analysis board

SINGLE MODULE SYSTEMS--- Install the Analysis board in the next rearmost slot (toward the rear of the HP 64000) from where the Emulation Control board will be installed. For example, if the Emulation Control board is to be installed in slot 8, the Analysis board should be installed in slot 9.

MULTIPLE MODULE SYSTEM--- Install the Internal Analysis boards between the locations where the Emulation Control boards will be installed.

e. Install the Memory Control board

SINGLE AND MULTIPLE MODULE SYSTEMS--- Install the Memory Control board in the next slot forward (one slot number less than) from the location where the Emulation Control board will be installed.

f. Install The Memory Boards

SINGLE AND MULTIPLE MODULE SYSTEMS--- Install the Memory board(s) in the slot(s) in front of the associated Memory Control board.

g. Install the Emulation Pod And Emulation Control board

Three multicolored ribbon cables are attached to the Emulation Probe (pod). These cables are used to connect the pod to the Emulation Control board (see figure G1-5). The cables have connectors which will only fit one way, described as follows: The brown and the red color-coded connectors on two of the cables plug in to the brown and red coded surface-mounted connectors located in the upper-right portion on the component side of the Emulation Control board. The yellow-coded connector on the other cable mates with the yellow-coded connector on the top edge of the Emulation Control board. Pin 1 on each cable connector is indicated by a triangle molded into the connector. Proper connection is also facilitated by keying of the connectors. Connect the pod to the Control Board by joining the connectors.

SINGLE MODULE SYSTEMS--- Install the Emulation Control board in slot 8 of the card cage to maximize the free cable length outside the development station chassis. Before fully seating the Emulation Control Board, connect the two Emulation Memory Control Split-bus cables from the Emulation Control Board to the Memory Control board located in the slot directly in front of the Emulation Control board.

Emulator/Analyzer 68000/68008
 Installation and Service Information - Installation

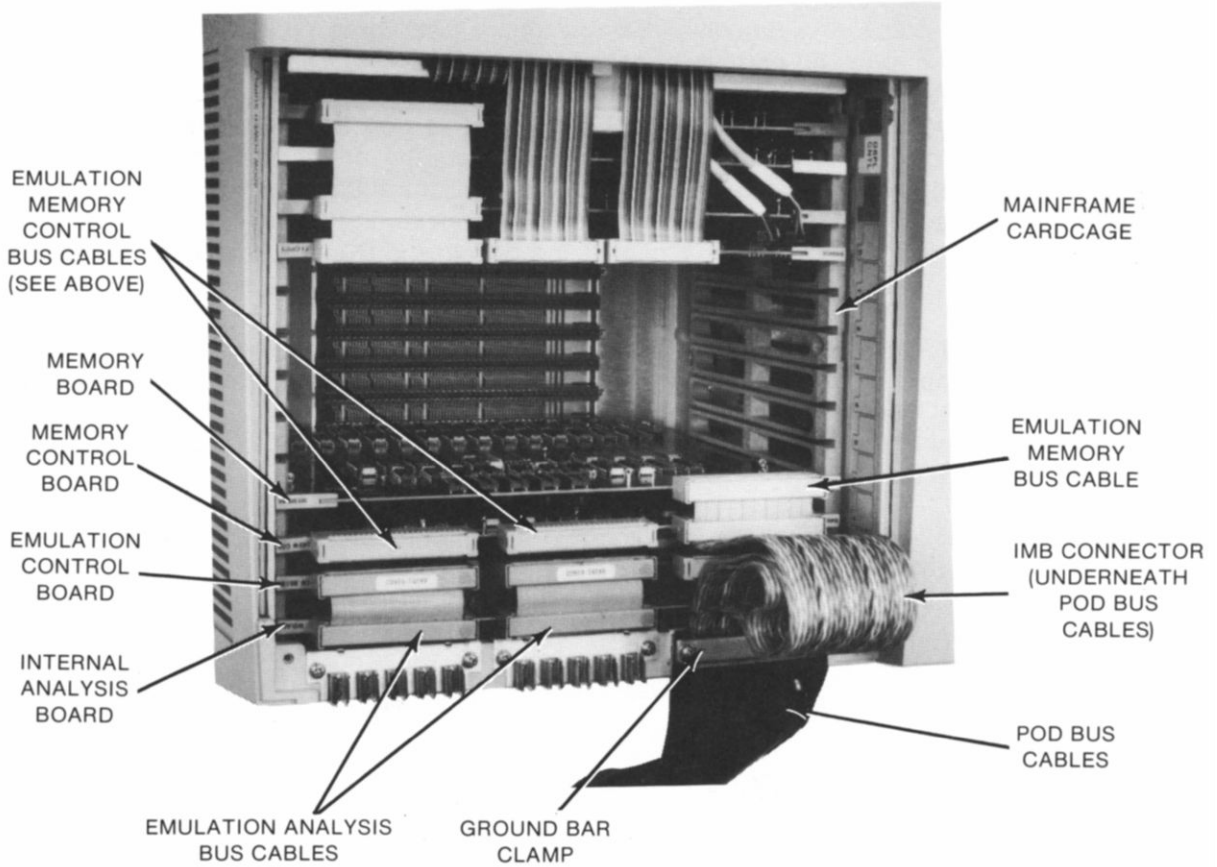
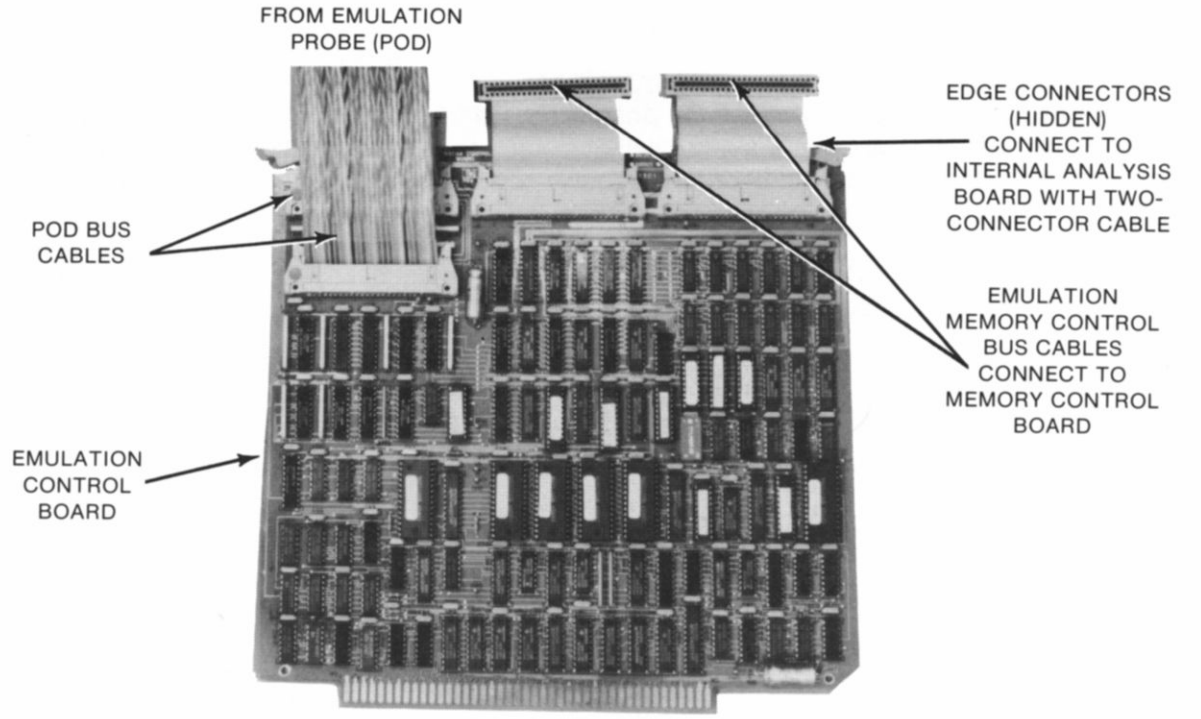


Figure G1-5. Emulation Control Board Connections

MULTIPLE MODULE SYSTEM--- Install the Emulation Control boards in slots 3 and 9 of the card cage so that problems connecting emulation and intermodule bus cables can be avoided. Before fully seating the Emulation Control Boards, connect the two Emulation Memory Control Split-bus cables from each Emulation Control Board to the Memory Control board located in the slot directly in front of each Emulation Control board.

- h. Install the ground bar clamp (see figure G1-6) across the emulator pod cable shielding to provide radio frequency interference (RFI) suppression, as follows:

Loosen both screws securing the ground bar clamp to the development station chassis. Swing the ground bar clamp back and insert the emulation pod cable between the two ground bar clamp screws. The shielded portion of the cable should be positioned between the two screws. There are notches in the cable to locate it in the correct position. You will need to fold the cable toward the front of the development station, then back to the rear, in order to position the cable properly.

Swing the ground bar clamp over the cable so that the slot in the ground bar clamp fits beneath the screw head. Tighten both screws securely, but take care not to damage the cable by over tightening the screws.

- i. Install the bus cables

After all the emulation, memory, and analysis circuit boards have been installed in the development station card cage, the emulation memory, emulation analysis bus cables, and the intermodule bus (IMB) cables must be installed across the top of the board set. See figure G1-7 for the cable configuration of a complete system.

The two cables (located at the top edge and on the right of the circuit board set as you face the front of the development station) connecting the Internal Analysis board and the Emulation Control board are the Emulation Analysis Bus cables. The connectors are keyed to facilitate correct installation, and each connector has a triangle indicator molded into the connector to indicate the location of the pin 1 side and end in the connector. When properly installed, the red stripe marker on the bus cable is on the left-hand side of the cable when viewed from the front and above the card cage.

The Emulation Memory Bus cable is on the left-hand side of the board set, as you face the front of the development station. The Memory Control board is joined to the Memory boards by this cable. The connectors are color coded and keyed to facilitate proper installation, with the color coding placed to the left end of the connector. Each connector has a triangle indicator molded into the connector to indicate the location of the pin 1 side and end in the connector. When properly installed, the red stripe marker will be on the left of the ribbon cable as you look down on the card cage from the front of the development station.

The intermodule bus (IMB) is not shown in figure G1-5. It is used when logic analyzers, other than the internal analyzer, are used with the emulation system. The IMB, when used with the emulation system, requires the internal analysis board to be installed in the development station to interface with other analysis boards. The analysis boards contain a 20-pin IMB connector on the top, left side of the board which is used to connect the boards for intermodule measurements. The intermodule bus between boards consists of a 20-conductor ribbon cable that is installed on the IMB connector of the appropriate board(s).

- j. Replace the card cage access cover and secure in place with the two attached hold-down screws.

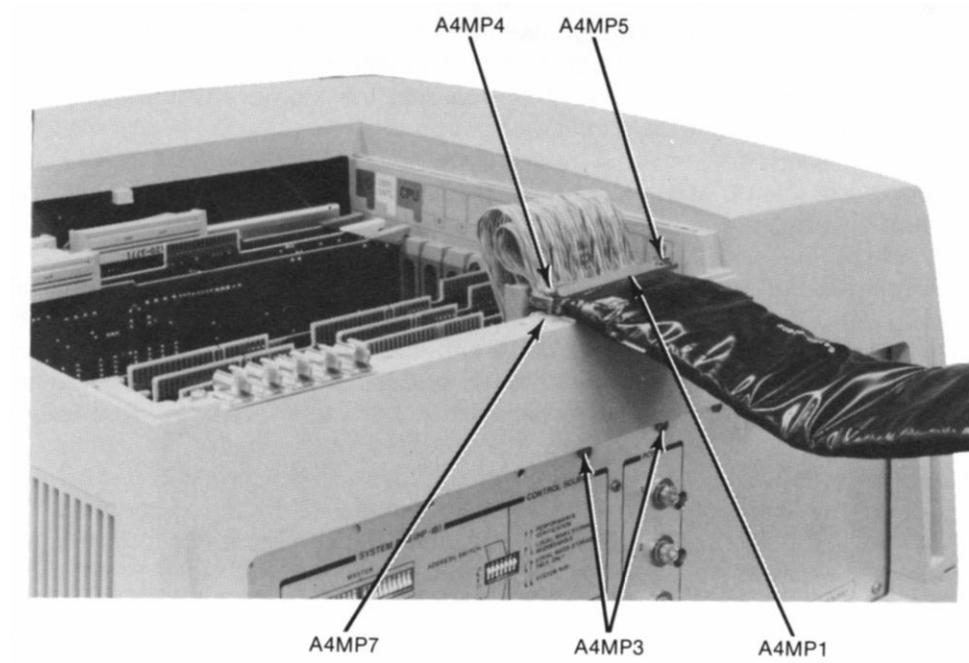


Figure G1-6. Ground Bar Clamp Installation (HP 64100A)

INSTALLING EMULATION SYSTEM HARDWARE INTO AN HP 64110 LOGIC DEVELOPMENT STATION

WARNING

Any installation, servicing, adjustment, maintenance, or repair of this product must be performed only by qualified personnel. Make sure power is off prior to performing any of the installation instructions given below.

NOTE

The following installation steps assume the installation of a complete system (maximum memory and internal analysis). Disregard procedure steps for equipment you do not require or have.

Configuration of the Boards in the Station

While the emulation and analysis circuit boards may be installed in any card slot in the station chassis, mechanical considerations (i.e.; cabling) make the following card groupings most convenient:

- slot 0 Internal Analysis board
- slot 1 Emulation Control board
- slot 2 Memory Control board
- slot 3 Memory board
- slot 4 Memory board (if used)

Configuration of the Memory Boards

Configuration of the memory boards is described previously in the installation instructions for the HP 64100 station.

Installation Instructions

GENERAL. Installation of the circuit boards is accomplished by sliding each circuit board into the circuit board guide slots, with the component side of the board facing the top of the chassis. Align the connector at the bottom of the board with the motherboard connector in the card cage, then apply an inward pressure until the board is seated in the motherboard connector. Be sure the ejector handles on the top of the boards are parallel to the top of the board when the board has reached its full inward travel.

To install the Emulation System and related equipment, proceed as follows:

WARNING

Read the safety summary at the front of this manual before installation or removal of the Emulation Subsystem.

- a. Turn OFF power to the HP 64110 Development Station. (See figure G1-2 for the location of the power switch.)

CAUTION

Power to the HP 64110 Development Station must be removed before installation or removal of option cards (emulation, etc.) to avoid damage to the option cards and the development station.

- b. Gain access to the card cage as follows:

- o Loosen the two hold-down screws (see figure G1-7) on the rear of the card cage access cover, and remove the cover.
- o Remove the two rear plate screws (figure G1-7), and remove the rear plate.

- c. Install the Internal Analysis board

Install the Analysis board in the next higher slot (toward the top of the HP 64110) above the slot where the Emulation Control board will be installed.

- d. Install the Memory Control board

Install the Memory Control board in the next vacant slot beneath the slot where the Emulation Control board will be located.

- e. Install the Memory board(s)

Install the Memory board(s) in the slot(s) beneath the associated Memory Control board.

Memory board configuration is described previously in the paragraph entitled "Configuration of the Memory Boards" for the HP 64100 Development Station installation.

- f. Install the Emulation Pod And Emulation Control board

Three multicolored ribbon cables are attached to the Emulation Probe (pod). These cables are used to connect the pod to the Emulation Control board (see figure G1-5). The cables have connectors which will only fit one way, described as follows. The connector on one of the cables connects to the surface-mounted connector located on the upper-left portion of the component side of the Emulation Control board. The connector on the other cable mates with the connector on the top edge of the Emulation Control board. Pin 1 on the cable connectors is indicated by a triangle molded into each connector. Proper connection is facilitated by the

color coding and keying of the connectors. Connect the pod to the control board by joining the connectors.

Install the Emulation Control board into the slot guides between the Internal Analysis board and the Memory Control board to maximize the free cable length outside the development station chassis. Before fully seating the Emulation Control Board, connect the two Emulation Memory Control Bus cables from the Emulation Control Board to the Memory Control board located in the slot directly below the Emulation Control Board.

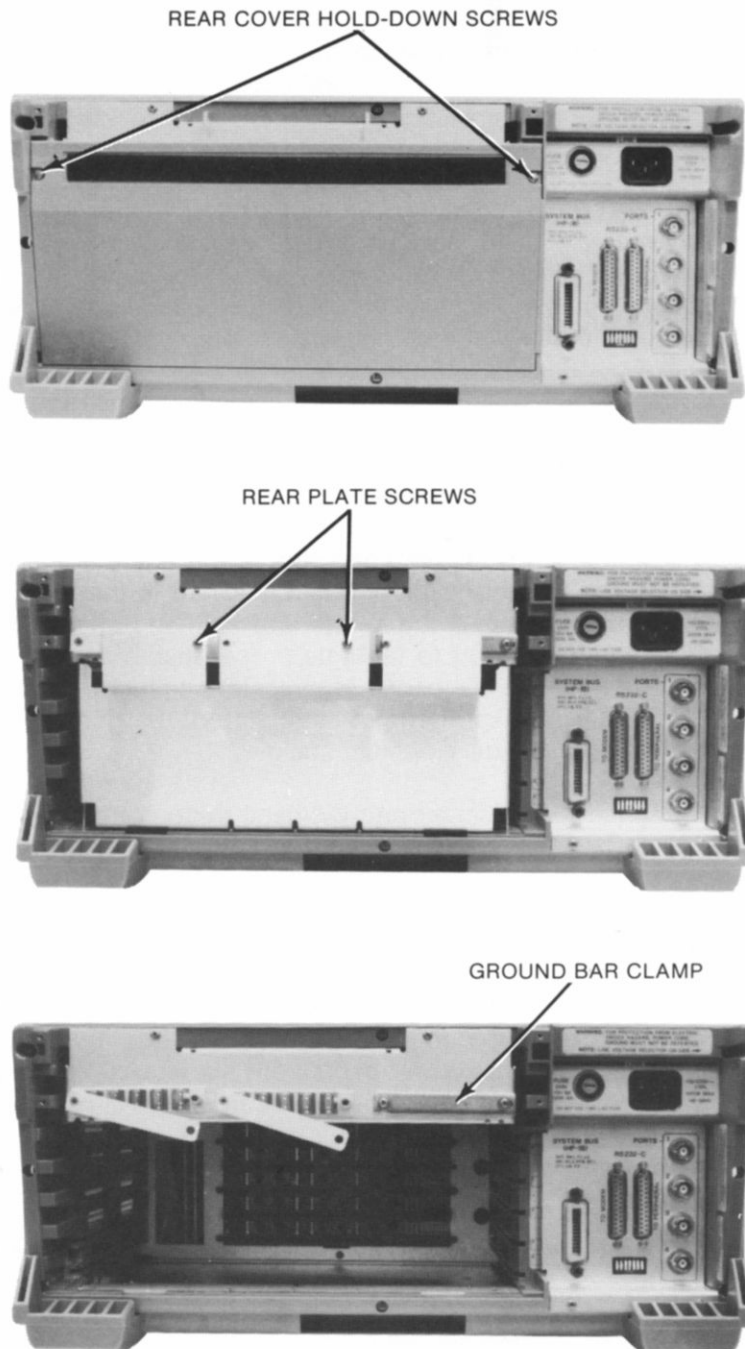


Figure G1-7. HP 64110 Installation Details

- g. Install the ground bar clamp (see figure G1-8) across the emulator pod cable shielding to provide radio frequency interference (RFI) suppression, as follows:

Loosen both screws securing the ground bar clamp to the development station chassis. Swing the ground bar clamp back and insert the emulation pod cable between the two ground bar clamp screws. The shielded portion of the cable should be positioned between the two screws. There are notches in the cable to locate it in the correct position. You will need to fold the cable toward the bottom of the development station, then back toward the top, in order to position the cable properly.

Swing the ground bar clamp over the cable so that the slot in the ground bar clamp fits beneath the screw head. Tighten both screws securely, but take care not to damage the cable by over tightening the screws.

- h. Install the bus cables

After all the emulation, memory, and analysis circuit boards have been installed in the development station card cage, the emulation memory, emulation analysis bus cables, and the intermodule bus (IMB) cables must be installed across the exposed edge of the board set. See figure G1-9 for the cable configuration for a complete system.

The three multicolored ribbon cables at the top right of the circuit board set (as viewed from the rear) that connect the Internal Analysis board and the Emulation Control Board are the Emulation Analysis Bus cables. These connectors are keyed and color coded. Each connector also has a triangle indicator molded into the connector to indicate the pin 1 location.

The Emulation Memory Split-bus is on the left-hand side of the board set as you face the rear of the development station. The Memory Control board is joined to the Memory boards by the two short Split-bus cables which attach to the center and left-hand edge connectors (as seen from the rear). The connectors are keyed to facilitate proper installation, and each connector has a triangle indicator molded into the connector to indicate the pin 1 location. When properly installed, the red stripe marker will be on the right-hand side of the ribbon cables as you look into the card cage from the rear of the development station.

The analysis system is attached to the center and left-hand edge connectors as seen from the rear) with the Emulation Bus Cable HP 64960A.

The intermodule bus (IMB) is not shown in figure G1-9. It is used when logic analyzers, other than the internal analyzer, are used with the emulation system. The IMB, when used with the emulation system, requires the internal analysis board to be installed in the development station to interface with other analysis boards. The analysis boards contain a 20-pin IMB connector on the top, left side of the board which is used to connect the boards for intermodule measurements. The intermodule bus between boards consists of a 20-conductor ribbon cable that is installed on the IMB connector of the appropriate board(s).

- i. Reinstall the rear plate and rear access cover by reversing the procedures given in step b, above.

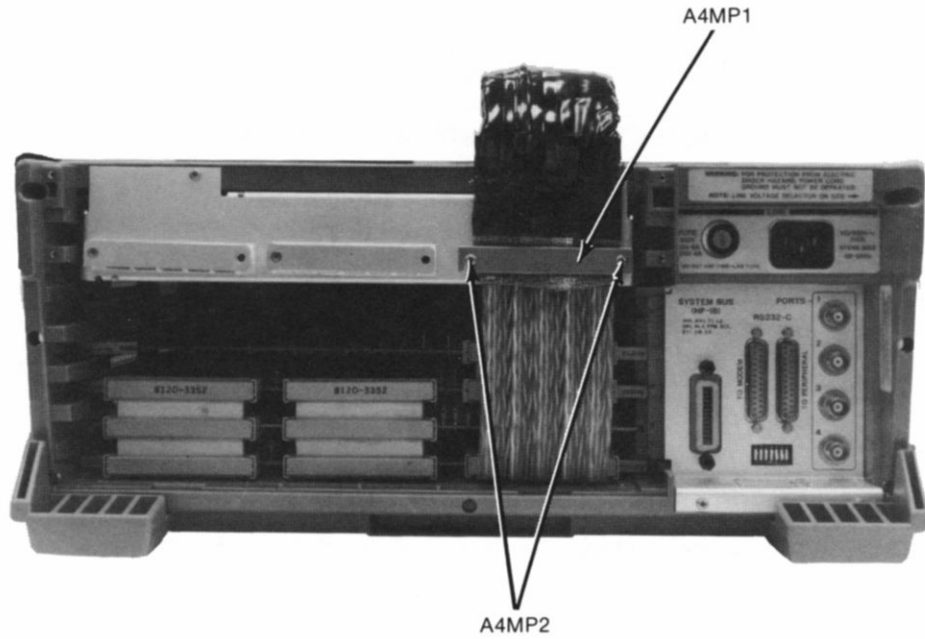


Figure G1-8. Ground Bar Clamp Installation (HP 64110A)

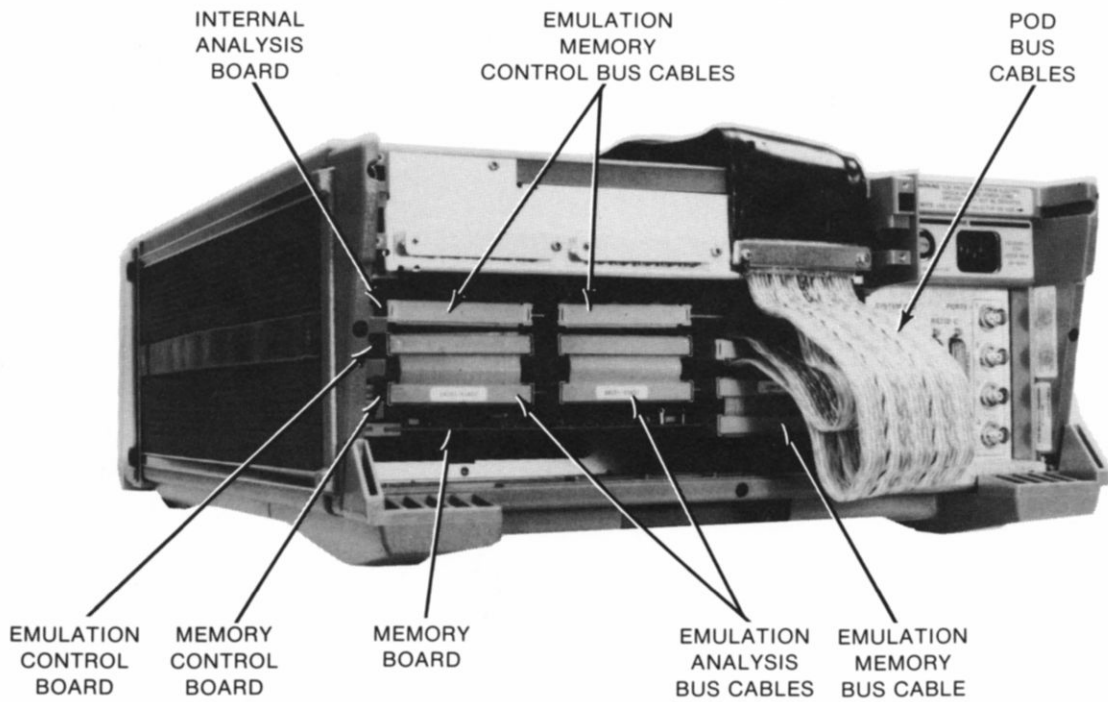


Figure G1-9. HP 64110 Memory, Emulation, and Intermodule Bus Cabling

INSTALLING THE EMULATION PROBE INTO THE TARGET SYSTEM



PROTECT AGAINST STATIC DISCHARGE

The emulation probe contains devices that are susceptible to damage by static discharge. Therefore, precautionary measures should be taken before handling the microprocessor connector attached to the end of the cable from the emulation probe to avoid damaging the internal components of the probe by static electricity.



Do not install the emulation probe into the processor socket with power applied to the target system. The pod may be damaged if power is not removed before installation.

The emulation probe is provided with a pin protector that prevents damage to the probe when connecting and removing the probe from the microprocessor socket. DO NOT use the probe without a pin protector installed. If the emulation probe is being installed on a densely loaded circuit board there may not be enough room to accommodate the plastic shoulders of the probe socket. If this occurs, another pin protector may be stacked onto the existing pin protector. The wire extending from the emulation pod may be connected to the target system signal ground.

Carefully remove the target processor from its socket, and place the processor into a protected area. Then install the emulation probe into the vacant socket.

Target System Microprocessor Connector Compatibility

To properly connect and operate the emulator, your target system microprocessor socket must be compatible with the microprocessor connector on the emulation system.

Installing Into a DIP Socket (See Figure G1-10)

To connect the microprocessor connector to a target system containing a dual in-line pin (DIP) socket for the host processor, proceed as follows: Remove the 68000 microprocessor from the target system microprocessor socket. Store the microprocessor in a protected environment. Note the location of pin 1 on both the microprocessor connector and the target system socket. Place the microprocessor connector attached to the end of the cable from the Emulation Probe into the target system microprocessor DIP socket.

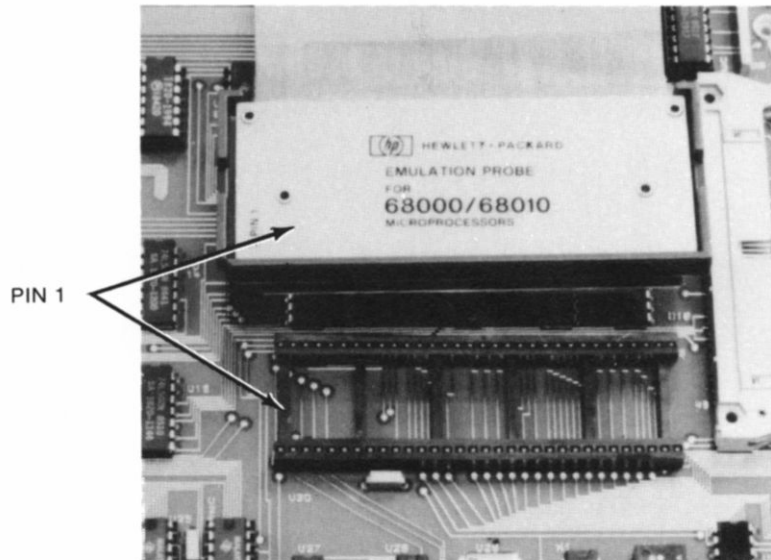


Figure G1-10. Installing the Emulation Probe into a DIP Socket

Installing into a PGA Socket (See Figure G1-11)

To connect the microprocessor connector to a target system containing a Pin Grid Array (PGA) socket, proceed as follows: Remove the 68000 microprocessor from the target system microprocessor PGA socket. Store the microprocessor in a protected environment. Note the location of pin 1 on both the microprocessor connector and the target system socket. Place the microprocessor connector attached to the end of the cable from the Emulation Probe into the target system microprocessor socket.

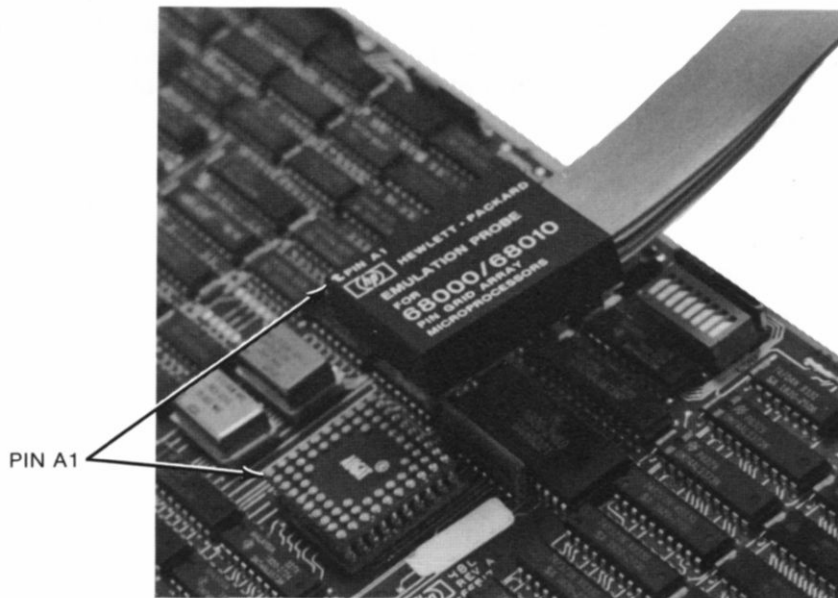


Figure G1-11. Installing the Emulation Probe into a PGA Socket



PROTECT PGA PINS FROM DAMAGE

To avoid damaging the PGA (Pin Grid Array) probe connector pins, an insertion/extraction tool (such as Augat P/N TX 8136-12) is recommended for removing the PGA probe connector.

TURNING ON THE DEVELOPMENT STATION

The power switches for the HP 64100 and HP 64110 development stations are identified in figure G1-1 and figure G1-2, respectively.

Turn the station power on.

You will hear several tones which indicate that power has been applied. Two additional tones indicate that a self-test has been performed. The initial display on the cathode ray tube (CRT) will be similar to the display in figure G1-12. The I/O bus configuration for your HP 64000 cluster system should list every station, hard disc and printer on the system bus.

```
I/O BUS CONFIGURATION

ADRS  DEVICE
  0  13037 DISC CONTROLLER
      UNIT  0  7906 DISC MEMORY  LU=0
  1  PRINTER (t2)
  3  64000
  4  THIS 64000
  5  64000

STATUS: Awaiting command      userid _____ 0:00

  edit  compile  assemble  link  emulate  _____  run  ---ETC---
```

Figure G1-12. I/O Bus Configuration Display

LOADING EMULATION SYSTEM SOFTWARE

Backing Up Your Software

As soon as you receive your emulation system software you should make a back-up copy for archival purposes. Keep the flexible discs which you received from Hewlett-Packard as an archive copy.

To duplicate your emulation system software:

1. PRESS *-BACKUP-- floppy utilities* **(RETURN)**.
2. Insert a new flexible disc into disc drive 1.
3. PRESS *format 1* **(RETURN)**.
4. When the disc format is complete insert Emulation System Software disc 1 into disc drive 0.
5. PRESS *duplicate 0* **(RETURN)**.
6. Repeat the above steps for Emulation System Software disc 2.
7. PRESS *end* **(RETURN)**.

Store the master copy of the Emulation System Software that you received from Hewlett-Packard in a safe place.

Loading Software in Clustered Stations Configuration

Your 68000 and/or 68008 Emulation System includes two flexible discs containing all of the emulation software. Follow the instructions given below to load this software onto a hard disc in a cluster configuration.

1. Insert disc 1 of the Emulation System Software into disc drive 0.
2. PRESS *-BACKUP-- floppy sys_gen* **(RETURN)**.
3. PRESS *copy all from local 0 to bus_disc 0* **(RETURN)**.

NOTE

On the left side of the display you see module name EMULATION_S68000. When the copy is complete, EMULATION_S68000 will be added to the list on the right side of the display. Press the **(NEXT PAGE)** key on the cursor control panel until you see EMULATION_S68000 on the "System modules on bus disc" list. For the 68008 processor, substitute "S68008" in place of "S68000" in all the software backup and loading procedures in this section.

4. Remove disc 1 of the Emulation System Software from disc drive 0.
5. Insert disc 2 of the Emulation System Software into disc drive 0.
6. Press **RETURN**.
7. PRESS *end* **RETURN** to exit from sys_gen.
8. Remove disc 2 of the Emulation System Software from disc drive 0 and store both discs for future use.

Configuring A Flexible Disc For Stand-Alone Operation

Follow these guidelines to configure flexible discs for stand-alone operation (operation from 5 1/4 inch flexible discs).

- o Format a new flexible disc.
- o Define the configurations of software you will need on each flexible disc.
- o Copy (use floppy sys_gen) the operating system modules and emulation modules you need onto the newly formatted disc(s).

FORMAT A NEW FLEXIBLE DISC. You will need to format three discs for the suggested configuration given below. To format a disc, proceed as follows:

1. Insert Operating System disc 2 into disc drive 0.
2. Insert a new disc into disc drive 1.
3. PRESS *floppy utilities* **RETURN**.
4. PRESS *format 1* **RETURN**.
5. When the formatting is complete the STATUS line will read:

Disc 1 formatting completed
6. To exit floppy utilities, press *end* **RETURN**.

DEFINE DISC SOFTWARE CONFIGURATIONS. The following configuration is a suggestion for configuring your discs to assemble, link, edit your files, and run emulation and analysis. Disc 1 is required in all situations. Install it in disc drive 0. Use disc 2 with disc 1 for assembling, linking, or editing. Use disc 3 with disc 1 for emulation and analysis.

Disc 1	Disc 2	Disc 3
FLOPPY_OP_SYS	ASSEMBLER	SYS_GEN
FLOPPY_UTILITIES	LINKER	MEAS_SYS
MONITOR_S68000	COPY	EMULATION_S68000
YOUR SOURCE FILE	DIRECTORY	ANLY_302_S68000
YOUR ABSOLUTE FILE	EDITOR	
	ASMB_LINK_68000	

COPY THE MODULES ONTO NEW DISCS. To copy the Operating System and Emulation Software modules onto the suggested new discs, proceed as follows:

NOTE

The following procedure is based on the 2.01 version of the Operating System Software. If you have any other version, the software modules referred to in the procedures may not be on the disc specified. To find out where each of the modules reside, perform steps 1 through 5. Note the modules on disc 2 for future reference during the procedure. Remove Operating System disc 2 from disc drive 0 and insert Operating System disc 1 in its place. Press **(RETURN)** and note the modules on this disc. Repeat the procedure for Operating System disc 3. Now that you know where all of the modules reside, substitute the disc number in the remaining portion of the procedure for the disc number that contains the module that you want to transfer.

1. Insert Operating System disc 2 into disc drive 0 to boot the system up. Label one of the new formatted discs DISC #1, and insert it into disc drive 1.
2. Remove Operating System disc 2 from disc drive 0 and insert Operating System disc 3 in its place. This disc contains the SYS_GEN software which allows you to show and copy modules.
3. PRESS *floppy sys_gen* **(RETURN)**.
4. Remove Operating System disc 3 from disc drive 0 and insert Operating System disc 2 in its place.
5. PRESS *show local 0* **(RETURN)**. The display will show the modules on Operating System disc 2. You will copy some of these to Disc #1 in disc drive 1.
6. PRESS *copy* (Number of FLOPPY_OP_SYS) *from local 0 to local 1* **(RETURN)**.

7. PRESS *copy* (Number of FLOPPY_UTILITIES) *from local 0 to local 1* **(RETURN)**.
8. Remove Operating System disc 2 from disc drive 0 and insert Emulation System Software disc 2.
9. PRESS *show local 0* **(RETURN)**.
10. PRESS *copy* (Number of MONITOR_S68000) *from local 0 to local 1* **(RETURN)**.
11. Remove DISC#1 from disc drive 1. Label one of the new formatted discs DISC#2, and insert it into disc drive 1.
12. Remove Operating System disc 2 from disc drive 0 and insert the 68000 Assembler/Linker disc in its place.
13. PRESS *copy all from local 0 to local 1* **(RETURN)**.
14. Remove 68000 Assembler/Linker disc from disc drive 0 and insert Operating System disc 2 in its place.
15. PRESS *show local 0* **(RETURN)**. The display will show the modules on Operating System disc 2.
16. PRESS *copy* (Number of ASSEMBLER) *from local 0 to local 1* **(RETURN)**.
17. PRESS *copy* (Number of LINKER) *from local 0 to local 1* **(RETURN)**.
18. PRESS *copy* (Number of COPY) *from local 0 to local 1* **(RETURN)**.
19. PRESS *copy* (Number of DIRECTORY) *from local 0 to local 1* **(RETURN)**.

20. Remove Operating System disc 2 from disc drive 0 and insert Operating System disc 1 in its place.
21. PRESS *show local 0* (RETURN). The display will show the modules on Operating System disc 1.
22. PRESS *copy* (Number of EDITOR) *from local 0 to local 1*.
23. Remove DISC#2 from disc drive 1. Label the last of the new formatted discs DISC#3, and insert it into disc drive 1.
24. Remove Operating System disc 1 from disc drive 0 and insert Operating System disc 3 in its place.
25. PRESS *show local 0* (RETURN). The display will show the modules on Operating System disc 3.
26. PRESS *copy* (Number of MEAS_SYS) *from local 0 to local 1* (RETURN).
27. PRESS *copy* (Number of SYS_GEN) *from local 0 to local 1* (RETURN).
28. Remove Operating System disc 3 from disc drive 0 and insert Emulation System Software disc 1 in its place.
29. PRESS *show local 0* (RETURN). The display will show the modules on Operating System disc 1.
30. PRESS *copy* (Number of EMULATION_S68000) *from local 0 to local 1* (RETURN).
31. Remove Emulation System Software disc 1 from disc drive 0 and insert Emulation System Software disc 2 in its place.
32. PRESS *show local 0* (RETURN). The display will show the modules on Emulation System Software disc 2.
33. PRESS *copy* (Number of ANLY_302_S68000) *from local 0 to local 1* (RETURN).
34. Remove DISC#3 from disc drive 1. Cover the notch on the side of discs #2 and #3 that you have just loaded to protect your new stand-alone configuration operating system discs. Leave disc #1 unprotected so that you can write your program to it.

REMOVING EMULATION SOFTWARE FROM THE OPERATING SYSTEM

System software, such as Emulation software, cannot be purged from the system and it cannot be removed file-by-file. System software must be "removed" from the Operating System via floppy `sys_gen`. The following instructions will illustrate this process:

1. PRESS *-BACKUP-- floppy sys_gen* **(RETURN)**.
2. PRESS *show bus_disc 0* **(RETURN)**.

NOTE

PRESS **(NEXT PAGE)** key on cursor control panel until you locate the module you want to remove. You can remove or copy modules by their list number or by the module name.

3. PRESS *remove <MODULE> or <NUMBER> from bus_disc <DISC #>* **(RETURN)**.
4. PRESS *end* **(RETURN)**.

PERFORMING OPERATION VERIFICATION

To perform operation verification:

1. Make sure the board configuration is correct.
2. Unplug the emulator from your target system.
3. Remove the conductive pin protector from the emulator plug-in.
4. PRESS *---ETC---* until `opt_test` appears as a softkey option.
5. PRESS *opt_test* **(RETURN)**.
6. PRESS *<SLOT #>* (of your emulator) **(RETURN)**. The display shown in figure G1-13 will appear on the screen.
7. PRESS *cycle*. The inverse video bar will cycle down through each of the lines denoting the different tests being performed.
8. If # Fail column indicates that any of the tests have failed, refer to Section G2 of this manual.
9. To exit the `opt_test` routine press *end*.
10. To return to the system monitor press *end*.

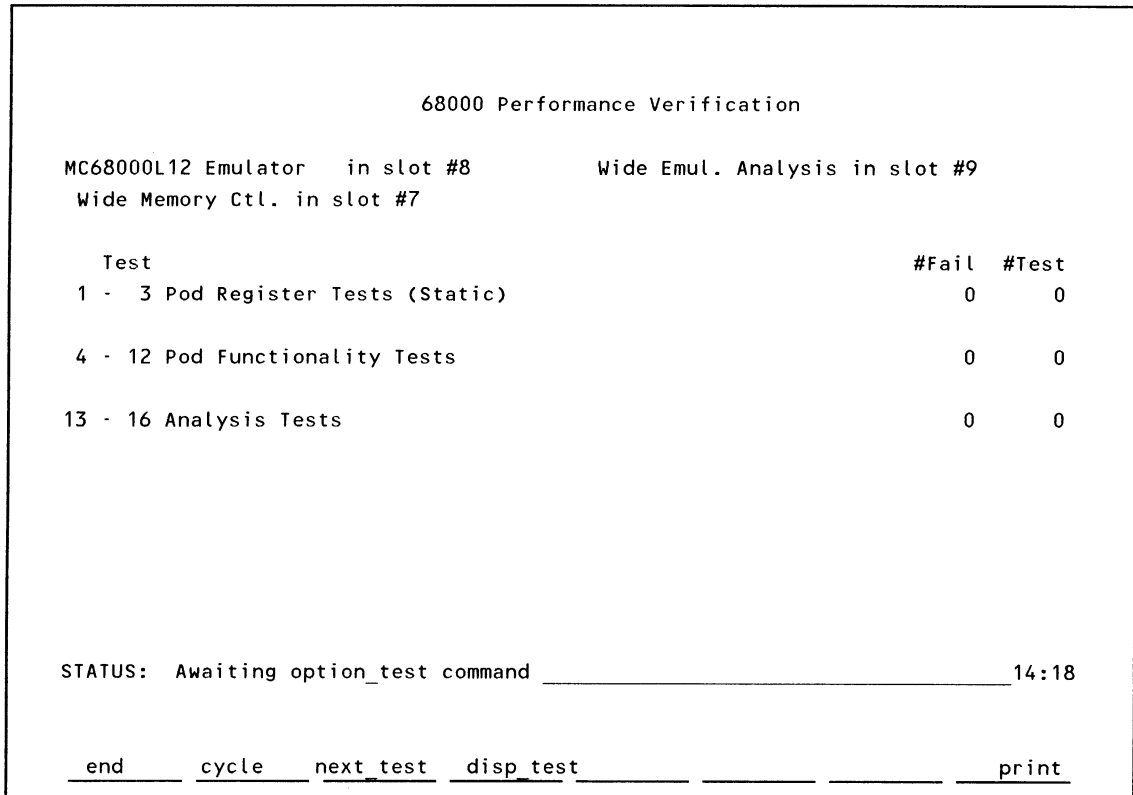


Figure G1-13. Option Test Display.

OPERATING THE EMULATION SYSTEM IN STAND-ALONE CONFIGURATION

Remember, when you are operating in stand-alone mode:

- a. If your operating system in disc drive 0 is write-protected, the files you create, edit, assemble, and emulate must be assigned to disc drive 1 by adding :1 to the file name--i.e., FILENAME:1.
- b. When you are writing to your files you will see an error message on the status line "Disc 0 write protected". You can ignore this message as long as you are correctly assigning your files.

REMOVING THE EMULATOR CONTROL BOARD

To remove the 68000 Emulator Control board, use the following procedures:

WARNING

Read the safety summary at the front or this manual before installation or removal of the Emulation Subsystem.

- a. Turn OFF power to the HP 64000 Development Station.

CAUTION

Power to the HP 64000 Development Station must be removed before installation or removal of option cards (emulation, etc.) to avoid damage to the option cards and the development station.

- b. Remove the card cage access cover. (Refer to installation procedures given earlier in this section for instructions to remove this access cover.)
- c. Remove the emulation bus cables.
- d. Remove the RFI ground bar clamp.
- e. Pull up on the board extractor levers and pull the card clear of the development station card cage.

REPACKAGING FOR SHIPMENT

Original Packaging Materials

Containers and packing materials identical to those used in factory packaging are available through Hewlett-Packard offices.

Other Packaging Materials

The following general instructions should be used for re-packing with commercially available materials:

- a. Wrap the emulator component in heavy paper or plastic.
- b. Use a strong shipping container. A double-wall carton made of 350-pound test material is adequate.
- c. Use a layer of shock-absorbing material 70 to 100 mm (3 to 4 inches) thick around all sides of the components to provide firm cushioning and prevent movement inside the container.
- d. Seal shipping container securely.
- e. Mark shipping container FRAGILE to ensure careful handling.
- f. In any correspondence, refer to instrument by model number and full serial number.

Appendix G

INSTALLATION AND SERVICE INFORMATION

Section G2

PERFORMANCE VERIFICATION AND TROUBLESHOOTING

INTRODUCTION

This section is arranged in two parts, Performance Verification and Troubleshooting. The scope of the Performance Verification (PV) is to detect problems at the board level only. Part II is restricted to a System and Subsystem Overview, and a Troubleshooting Flow Chart. Board level troubleshooting is in support of Hewlett-Packard's Bluestripe Exchange Program. For convenience, the PV screen displays that are generated by the Performance Verification are grouped together in Figures G2-2 through G2-7.

PERFORMANCE VERIFICATION

In a networked HP 64000 system (hard disk based), and in a stand-alone HP 64000 configuration (disk based), the Performance Verification software for the Emulator Control Card and the Emulator Pod is a subset of the option_test PV. The option_test PV tests all possible option modules that can be configured within the expansion slots of the HP 64000 development station.

The following procedures explain how to run Performance Verification on the Emulation Subsystem for both a networked and a stand-alone HP 64000 system.

Other option cards connected to the emulator, such as memory control and analysis, have their own option_test software which tests the major functions of those cards but not interaction with other modules. To run Performance Verification on those option cards, refer to the service manual for the model number in question.

To test the 68000 emulator in a networked HP 64000 system, or in a stand-alone HP 64000 system, proceed as follows:

NOTES

Both the HALT and the RESET switches must be set to the UNBUFFERED positions.

If the HP 64000 system is a stand-alone configuration (disk based), then the following software modules must be contained on the current local disk system:

FLOPPY_OP_SYS
OPTION_TEST
PV_EMUL_S68000 or PV_EMUL_S68008

Other software modules may be necessary to perform other system functions.

- a. Verify that the emulator is correctly installed in the development station and that the user probe is disconnected from the target system. (Refer to Section G1 for installation instructions.)
- b. With the operating system initialized and awaiting a command, use the softkey or manually type the lower case command:

[option_test] **RETURN**

NOTES

Figures G2-2 through G2-7 are Performance Verification displays for the 68000 microprocessor. For the 68008 microprocessor, the PV displays will indicate 68008 Performance Verification at the top of the screen. Also, on the Pod Functionality Test (Figure 2-5), the Address field will only be 20 bits wide (00000H) rather than 24 bits wide (000000H) as shown for the 68000.

In Figure G2-6 (Analysis Stimulus Tests) the results in parentheses are the cumulative results for all the analysis test. The results that are not in parentheses are for the last analysis test run.

- c. The PV will now display a directory of the installed option 68000 boards and their card slot number (See figure G2-2). Locate the Emulator and enter the card slot number. For example, in Figure G2-2 the 68000 Emulator is in card slot 8. Therefore, enter:

8 **RETURN**

- d. A menu will now be displayed listing the four major tests available to exercise the 68000 Emulator (Figure G2-3). Press the *[cycle]* softkey. This will cause the performance verification software to cycle through the emulator PV tests. If no failures are observed, the 68000 Emulator operates correctly, and the testing may be terminated as described in the next paragraph. If a failure is observed, refer to the following sections for information on test operation and troubleshooting procedures.
- e. To terminate execution of the Performance Verification, press the *[end]* softkey. This returns the display to the option_test Performance Verification card cage listing. If *[end]* is pressed at this level, the option_test Performance Verification is completely exited and the system is returned to the awaiting command status.
- f. If further information is needed on the error results of each test, or you wish to execute only an individual test repeatedly, then the softkeys must be used to bring up the appropriate test display and begin test execution. The softkeys have the following actions:

[cycle]

Causes the performance verification software to execute each test on the display in turn; the inverse video bar will highlight each test name as the test is executed.

- [disp_test]*** When this softkey is pressed, the subtest display for the test name highlighted by the inverse video bar is put on screen. This subtest display shows all of the subtests and error status information for the particular test name.
- [end]*** This softkey, when pressed at the performance verification overview level (six major tests displayed), causes the testing to terminate at the end of the current cycle and returns the display to the option_test card slot listing. When ***[end]*** is pressed at this level, the performance verification software is exited entirely and the system is returned to the awaiting command status.
- [exit_test]*** This softkey, when pressed in the subtest display level, ends test execution at the end of the current cycle and returns the performance verification software to the next higher test display level.
- [next_test]*** Causes the inverse video bar to move to and highlight the next test name on the test display.
- [option_test]*** Causes the operating system to load the performance verification software and begin execution; the option_test card slot listing is displayed.
- [print]*** Causes all the information above the status line on the current test display to be copied to the system printer, if one is connected. The copy will not be done until the end of the current test cycle, if testing is in progress.
- [<SLOT #>]*** Pressing this softkey while in the option_test card slot listing display causes the system to prompt you for the slot number of the option module you wish to test.
- [start]*** Begins repeated execution of the test highlighted by the inverse video bar.

```
I/O BUS CONFIGURATION

ADRS  DEVICE
  0    7910 DISC MEMORY          LU=0
  1    2631 PRINTER
  5    64100
  6    THIS 64100

STATUS: Awaiting command          userid _____ 14:18

option_test _____
```

Figure G2-1. Awaiting Command Status

```
HP 64000 Option Performance Verification

Slot #  ID #  Module
-----
  6    01F8H  128 Kbyte Emul Memory
  7    0201H  Wide Memory Ctl.
  8    0035H  MC68000L12 Emulator
  9    0102H  Wide Emul. Analysis
 10    0402H  32K Memory Expander

STATUS: Awaiting option_test command _____ 14:18

end <SLOT #> _____ print _____
```

Figure G2-2. Option_Test Card Slot Listing


```
68000 Performance Verification
```

```
MC68000L12 Emulator      in slot# 8      Emulation Analysis. in slot# 9
Wide Memory Ctl.  in slot# 7
```

Test	# Fail	# Test
1 - 3 Pod Register Tests (Static)	0	0
4 - 12 Pod Functionality Tests	0	0
13 - 16 Analysis Tests	0	0

```
STATUS: Awaiting option_test command _____ 14:18
```

```
_____ end _____ cycle _____ next_test disp_test _____ _____ print
```

Figure G2-3. Performance Verification

```
68000 Performance Verification
Pod Register Tests (Static)
```

Test	Results (Cumulative)		# Fail	# Test
1 - Pod Index Register	00H	00H	0	0
2 - Pod Control Register	0000H	0000H	0	0
3 - Break Vector Register	00000000H	00000000H	0	0

```
STATUS: Awaiting option_test command _____ 14:18
```

```
_____ _____ _____ start _____ exit_test _____ _____ print
```

Figure G2-4. Pod Register Tests (Static)

68000 Performance Verification Pod Functionality Test					
Test	Address	Status	Data	#Fail	#Test
4 - Reset Address	000000H	00H		0	0
5 - Run Status/Hold Proc.		00H/00H		0	0
6 - Address	000000H	00H		0	0
7 - Byte Operations		00H	0000H	0	0
8 - Data Bits		00H	0000H	0	0
9 - Break Vector Register	000000H	00H		0	0
10 - User Probe	000000H	00H		0	0
11 - Address counter	000000H	00H		0	0
12 - Pod user mapper		00H		0	0

STATUS: Awaiting option_test command _____ 14:18

_____ cycle next_test start exit_test _____ print

Figure G2-5. Pod Functionality Test

68000 Performance Verification Analysis Tests			
Analysis Stimulus Test	Results (Cumulative)	# Fail	# Test
Analysis Trace: Complete		0	0
Address	Address = 000000(000000)	0	0
Data	Data = 0000(0000)	0	0
Status	Status = 00000000(00000000)	0	0
Analysis Break		0	0

STATUS: Awaiting option_test command _____ 14:18

_____ start exit_test _____ print

Figure G2-6. Analysis Stimulus Test

68000 Performance Verification Analysis Tests		
Test	#Fail	#Test
13 - Bus cycle mode test	0	0
14 - Executed mode test #1	0	0
15 - Executed mode test #2	0	0
16 - Executed mode test #3	0	0

STATUS: Awaiting option_test command _____ 14:18

_____ start _____ exit_test _____ print _____

Figure G2-7. Analysis Tests

TROUBLESHOOTING

WHAT IS AN EMULATION SYSTEM?

Scope

Theory of operation for the 68000 emulator subsystem is restricted to an overview of emulation; a description of the emulation subsystem; and a description of the functional blocks of the emulator circuitry. A detailed (component-level) theory of operation for the emulator pod processor board A1 and timing board A2 are provided with the pod board schematics in Section G8.

Emulation Overview

"Emulate" means to equal. In the HP 64000 Logic Development system, the emulation implementation allows the user to replace the microprocessor in the target system breadboard with a plug from the emulator pod. The plug appears to act as much like the emulated microprocessor as is possible within design constraints.

The user may develop programs using the assembler and compiler utilities of the HP 64000 system, store these on disk for later use, edit them as desired, and link them together while assigning absolute addresses (using the relocatable linker utility of the system). These programs may then be loaded into memory provided by the target system or by optional high-speed emulation memory boards resident within the development station.

Once the programs are loaded into memory, the user may then release the emulator to run them so that an evaluation of the software or firmware can be made. When problems are found, the causes may be diagnosed by requesting the emulator to display its register, display a user or emulation memory location or locations, or perform other non-real time analysis functions. With the optional analysis module, you can request the emulator to step through the program, and you can accomplish real-time tracing of program state flow, thus providing powerful tools for program analysis and debugging.

The emulation processor actually accomplishes the non-real time functions listed above. This is done by linking an emulation monitor program into the user's code loaded into memory. The emulation processor may alternately execute from the actual user's program or from the emulation monitor routines as necessary to perform the emulation functions.

For example, if the user wishes to display the contents of the processor's registers, the development station CPU asserts a break request to the emulator. The emulator control card will then respond by jamming a series of instructions onto the processor's bus which cause the processor to jump into monitor code execution. After dumping register contents into emulation memory, the processor is then returned to the location in the user's program immediately following the location it was executing when the break request occurred. The development station CPU can now access the contents of emulation memory to read the register contents and display them on screen.

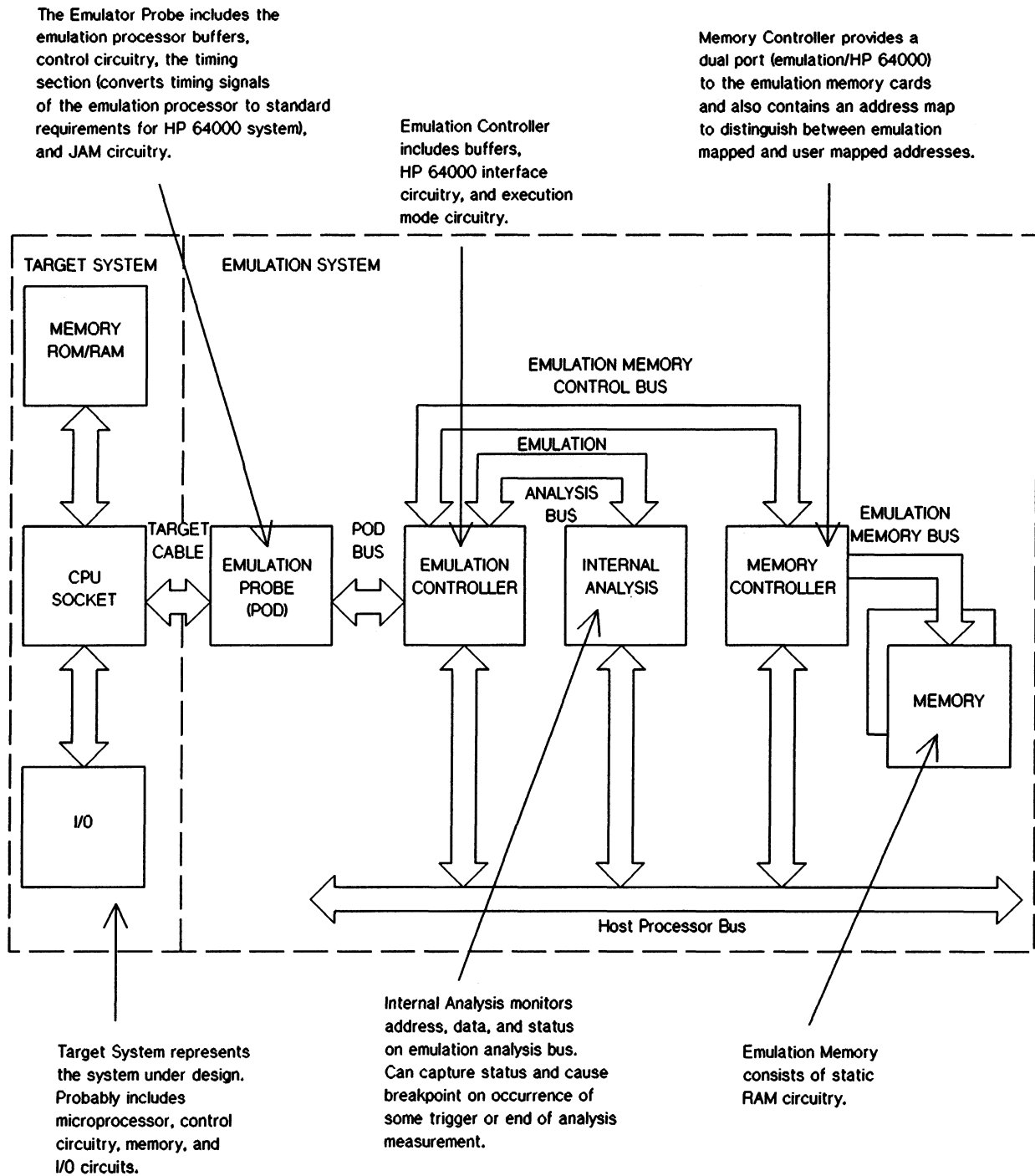


Figure G2-8. Emulation System Block Diagram

Emulation Subsystem Overview

GENERAL. Figure G2-8 shows a block diagram of the emulation subsystem. The subsystem provides emulation of a selected microprocessor in the user's target system. Capabilities include software development and debugging, hardware simulation, and real-time program execution.

EMULATOR POD. The Emulator Pod contains the specific microprocessor to be emulated. It may be used without a target system for software development; or its user plug may replace the microprocessor in the target system, allowing execution of software developed on the HP 64000 system. The pod communicates with the rest of the emulation subsystem via the pod bus.

EMULATION CONTROL CARD. The Emulation Control Card performs the interface functions between the emulator pod, the emulation analysis bus, the emulation memory control bus, and the development station bus. It buffers and controls data transactions between these three buses. This card also contains most of the execution mode circuitry and records various pieces of status information.

INTERNAL ANALYSIS. The Internal Analysis module continuously stores data present on the emulation analysis bus. It may be requested to store only certain sequences or types of data, and start or stop a store based upon the type of information present on the bus. The trace storage buffer is able to contain 256 different program states. A software disassembly function is also provided which converts binary information stored on the analysis board to instruction mnemonics, then displays them on the HP 64000 CRT at the user's request.

EMULATION MEMORY CONTROLLER. The Emulation Memory Controller is used to control access to emulation memory and perform a mapping function, which allows various blocks of memory to appear to reside at user selected address ranges and respond as different types of memory (ROM, RAM, undefined).

EMULATION MEMORY. The Emulation Memory board(s) consist of banks of RAM in multiples of 32K. Access to this memory is controlled by the emulation memory controller.

Emulator Block Diagrams

EMULATOR CONTROL CARD. Figure G2-9 is a block diagram of the emulator control card.

The emulator control card performs two functions. First, it is essentially an interface mechanism. It controls communication between the development station CPU (host processor), the emulator pod, the memory control bus, and the analysis bus. Second, it contains the bulk of the execution mode circuitry (the rest is contained on the emulation pod circuit boards).

In bus cycle mode, all address, data, and status information is buffered and is transmitted directly to the emulation memory controller bus for use in addressing emulation memory locations after translation by the memory controller mapper RAM. In addition, this information may be transmitted directly to the analysis address bus through the Analysis Address Buffers if the user has selected analysis of processor operation by bus cycle rather than executed cycles. The Last Address Latch may capture selected addresses if necessary for further interpretation by the user. This is done on receipt of a break request so that the cause of the break can be determined.

Data communications between the pod and the emulation interfaces is carried out through the Memory Bus Data Buffers. These buffers are controlled by signals from the emulator pod and the emulator control board. Typically, the emulation processor determines the orientation of these buffers as it carries out read and write transactions with emulation memory. However, in some instances the development station CPU must write data to the emulator pod registers to set up default execution conditions or read data from the pod registers during performance verification.

To accomplish communications with the pod registers, two additional sets of buffers come into play. These are the Development Station Data Transceivers and the Development Station to Memory Bus Data Transceivers. The Development Station Data Transceivers are controlled by logic driven by the development station CPU and buffer all data transactions between emulator circuitry and the development station CPU. The Development Station to Memory Bus Data Transceivers are controlled jointly by the emulator control card and the emulator pod; they buffer the reads and writes of the emulator pod registers initiated by the development station.

In addition to communications with the emulator pod data bus, the Development Station Data Transceivers buffer writes and reads of other emulator control card circuits. The ID Buffer provides an identification of the emulator to the development station upon request. The Emulation Status Buffer drives miscellaneous status information from the emulator pod and the control board to the development station CPU. This information allows the emulation software to determine whether or not the processor is executing bus cycles; whether it is in a bus grant, wait, or halted state, and whether or not the emulator pod is inserted into a powered up target system. The Emulator Control Register allows programming of various emulator pod options by the development station CPU. These options allow the user to reset the processor or request a hardware break and determine the mode of emulation analysis traces (all bus cycles or executed opcodes only). The Development Station Control Logic orchestrates all these operations by providing the proper function selection and timing signals to complete the transaction.

Address, data, and status from the emulator pod is also provided to the inputs of the analysis data buffers. The buffers drive information from the pod data bus to the analysis data bus continuously if the bus cycle mode of analysis tracing has been selected; otherwise, the buffers are tri-stated, and the analysis bus is driven by the execution mode RAMs.

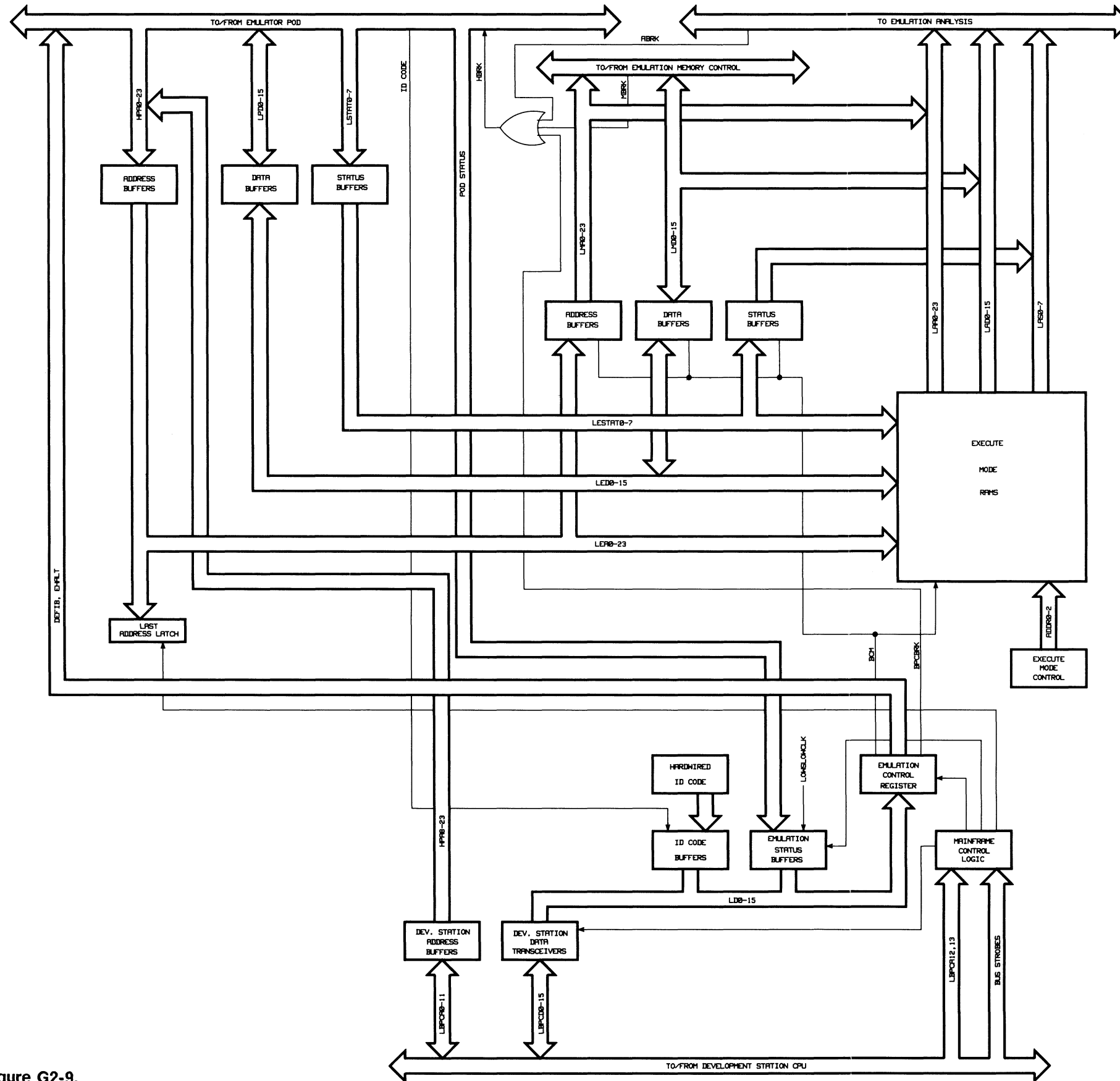


Figure G2-9.
 Emulator Control Card Block Diagram
 G2-12

EMULATOR POD. Figure G2-10 is a block diagram of the emulator pod. The emulator pod contains the 68000 microprocessor and circuits necessary to control the operation of that processor.

The interface between the 68000 microprocessor and the user target system is fully buffered to reduce the possibility of loading the user's system. Buffers drive address, data and status information to and from the user system. Note that the upper address and status information can be driven both to and from the user system. This allows DMA operations to emulation memory contained inside the development station from the user system.

A1 Processor Board. The Processor Board contains a 68000 microprocessor, buffers to interface address, data, and control signals to the user target system, logic gating for control signals, and buffers to interface address and memory control signals to the emulation system. Also on the processor board are latches for performance verification and a clock generator and selector.

The 68000 microprocessor is the heart of the emulator pod. Other circuitry in the emulator pod is used to buffer, monitor, and control the activity of the microprocessor. The processor may execute instructions stored in emulation or user memory, make I/O transactions with the target system, or release the bus to allow target system access to emulation memory (DMA). It is also allowed to respond to control lines from the user which may affect its instructions execution, such as interrupt or bus error signals.

The address transceivers buffer the emulation processor's address bus to the target system during normal operation (processor has bus control). When the processor has released the bus to a requesting device for a DMA transaction, the direction of the address buffers are reversed so that the target system may drive the address lines to the emulation system.

The PV latches are used to capture addresses present on the user side of the address transceivers during performance verification. These latches are then read onto the data bus through the data input buffers. The data read back is then compared with the addresses that were placed on the bus. This provides verification that the address transceivers are working properly.

The data input tri-state buffers transfer data onto the emulation processor's data bus when PDBDIR is high; PDBDIR is normally high during a read from user memory.

The data output buffers are tri-state drivers which allow data to pass from the emulation processor's data bus to the user data bus when PDBDIR from the emulator pod timing board is low. This will typically occur during a write to user memory or a read from emulation memory during DMA.

The status transceivers buffer status lines FC0-FC2 to and from the target system. They will be oriented to transmit these function codes to the target system from the emulation processor if the processor has not relinquished the system bus to a requesting device. If a DMA is granted to a target system device, the transceiver direction is changed so that the target system device may drive these lines into the emulator.

The memory control transceiver performs the same function as the status transceiver and responds in the same way to DMA requests. However, the signals buffered are LVMA, LLDS, LUDS, LAS, and R/LW, which are used to control data transactions with memory.

The interrupt enable/disable circuitry determines whether or not interrupts from the user system will be allowed to reach the emulation processor. Also input to this circuitry is the LINT7 signal from the emulator pod timing board, which is a level 7 (highest priority) interrupt. This particular

interrupt signal is used to transfer execution of the emulation processor into the emulation monitor program.

The VPA/BERR enable-disable is used to control whether or not VPA and BERR will be seen by the emulation processor, depending on system conditions at the moment those signals are asserted. Generally, VPA and BERR are only recognized during accesses to addresses mapped as residing in target system address space or during an acknowledge cycle for a user generated interrupt.

RESET/HALT switching allows the RESET and HALT lines to be either bi-directional between the user system and the processor; or they may only be an input line to the processor from the target system depending on the settings of the Reset and Halt switches on top of the emulator pod.

The clock select circuitry selects either the output of the 10 MHz crystal or the UCLK from the target system to drive the emulation processor's clock input, depending on the state of HICLK from the pod control register on the timing board.

The memory control logic drives memory control signals from the emulation processor to the emulation bus, which in turn drives emulation memory. Also performed in this block are gating functions producing signals which indicate when a bus cycle is in process and indicate valid data for an emulation write cycle.

The emulation control buffers are used to drive several control signals produced by the control board into the emulator pod. These signals are used by the emulation system to control various emulator functions, such as loading the pod control register, interrupting the emulation processor, orienting buffers and latches for data transactions, requesting the system bus, etc.

The memory mapper RAMs determine the direction of the user and emulation data buffers depending upon the configured memory map.

The execution mode address counters and comparators inform the execution mode circuitry on the emulation control board when a non-sequential address fetch is performed by the processor.

The Break Jam Registers are loaded by the development station CPU with an interrupt address which is forced onto the processor data bus when the processor performs the break. These registers are enabled onto the emulation processor's data bus by the break circuitry.

A2 Timing Board. The Timing Board contains circuits which convert user signals and emulation signals into properly timed signals for the emulation processor, such as interrupt, bus request, and data transfer acknowledge signals. Also included are data transceivers and logic for status information sent to the emulation system from the emulator pod.

The data transceivers and latches buffer data between the emulation processor data bus and the pod data bus (which is sent to the emulation data bus via the Control Board). The direction of these transceivers is controlled by signals from the emulation system and memory control signals output by the emulation processor.

The pod control register is a 16-bit latch which is used to select different features of the emulator, such as internal/external clock, DMA tracing modes, wait timing, interrupt enables, and other functions. The loading of this register is controlled by signals from the emulation system.

The status buffers are used to drive status information to the Control Board and the emulation system. Information is given about the emulation processor and the DMA arbiter circuitry. Status is used by the analysis module to report what type of bus cycle is currently in process.

The user data receive logic takes inputs from the emulation processor and the emulation system then determines whether or not to enable the outputs of the user data input buffers for a data receive operation by the emulation processor.

The DMA arbiter is used to arbitrate between bus requests by the emulation system and the user system. When the arbiter receives a request, it asserts the bus request line to the emulation processor.

The DTACK circuit applies a properly timed DTACK to the emulation processor. DTACK indicates that the data transfer for the memory cycle is complete.

The break circuit is used to send a level 7 (highest priority) interrupt to the processor and control the break vector register. A "break" is used to direct emulation processor instruction execution to a portion of the program known as the emulation monitor.

A break may be caused by the development station CPU, the analysis module, or the memory control board (in case of an illegal memory access).

DMA tracing options are provided so that the user can selectively observe DMA activity with the analysis module. Several bits from the pod control register along with the user bus arbitration control lines are used to modify memory control signals so that various modes of operation can be achieved. These include analysis of all DMA bus cycles with access to emulation memory allowed.

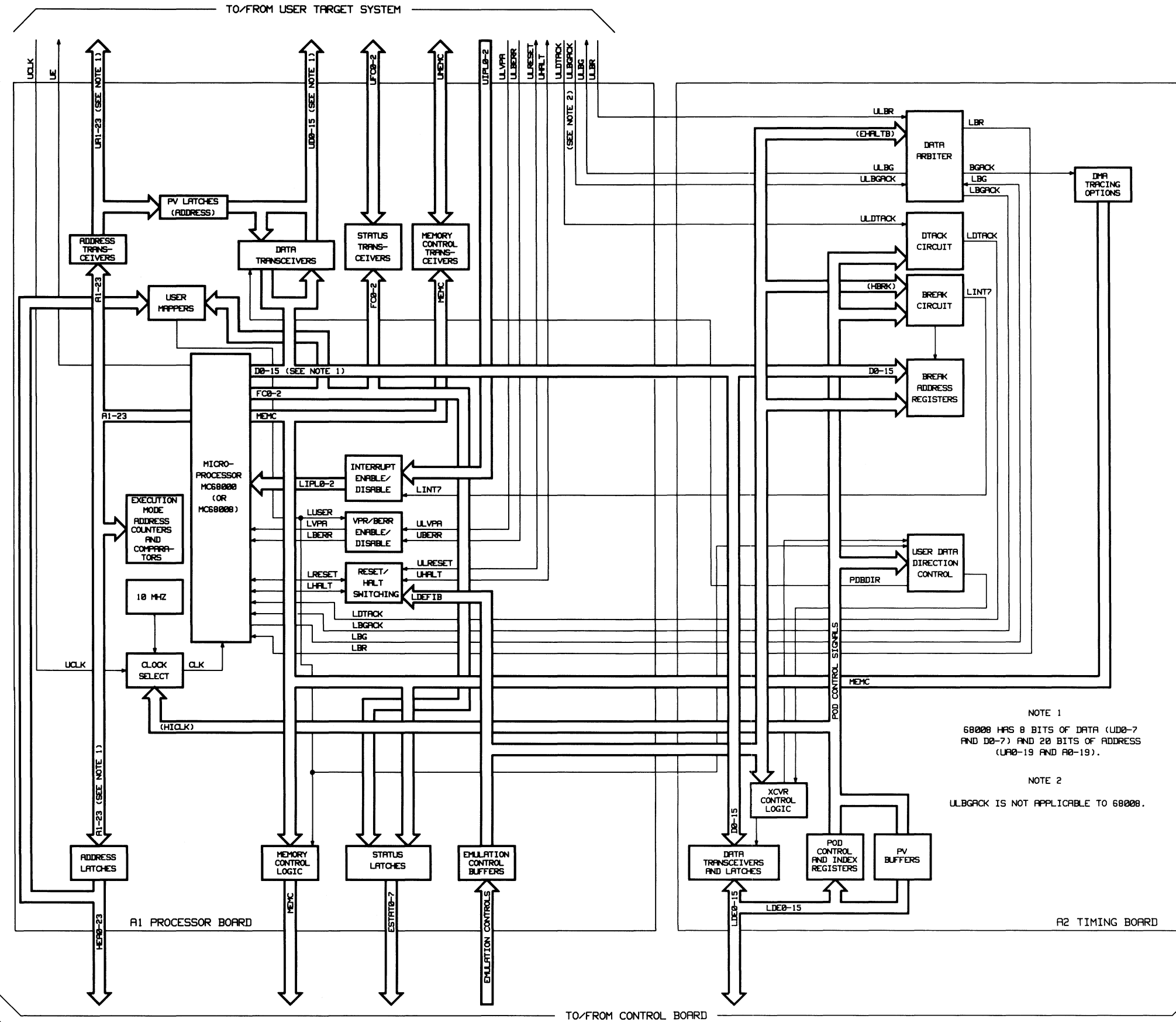
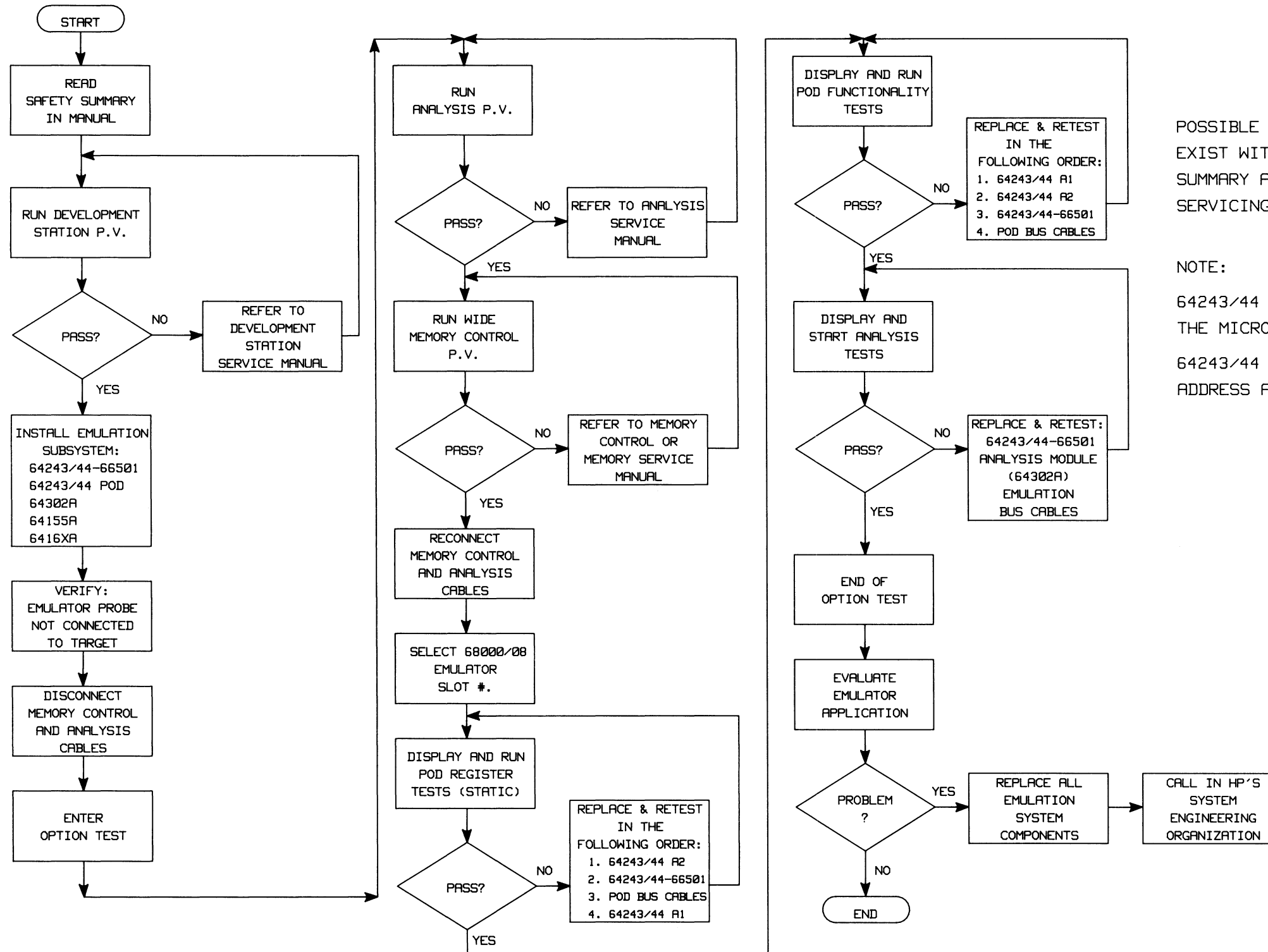


Figure G2-10.
 Emulator Pod Block Diagram
 G2-16

TROUBLESHOOTING FLOW CHART

Figure G2-11 is a troubleshooting flowchart for the 68000 emulator subsystem. When using the flowchart:

- a. If you need more specific instructions on running performance verification or interpreting the test results, refer to part one in this section.
- b. A block marked "Replace & Retest" on the flowchart, followed by a list of assemblies, indicates that each assembly should be replaced one at a time in that order and the emulation subsystem should be retested after each replacement.
- c. Servicing of the emulator circuitry should only be performed by qualified personnel aware of the hazards involved. Read the safety summary at the front of this appendix for further details on safety precautions to be taken while servicing this instrument.



WARNING

POSSIBLE ELECTRICAL SHOCK AND FIRE HAZARDS EXIST WITHIN THE MAINFRAME; READ THE SAFETY SUMMARY AT THE FRONT OF THE MANUAL BEFORE SERVICING.

NOTE:
 64243/44 A1 IS THE TOP POD BOARD CONTAINING THE MICROPROCESSOR.
 64243/44 A2 IS THE BOTTOM POD BOARD WITH THE ADDRESS AND DATA BUS LATCHES, ETC.

Figure G2-11.
 Emulator Troubleshooting Flowchart
 G2-18

Appendix G

INSTALLATION AND SERVICE INFORMATION

Section G3

ADJUSTMENTS

The Model HP 64243 68000 and the Model HP 64244 68008 Emulators have no adjustments.

NOTES

Appendix G

INSTALLATION AND SERVICE INFORMATION

Section G4

REPLACEABLE PARTS

INTRODUCTION

This section contains information for ordering parts. Table G4-1 lists abbreviations used in the parts list and throughout the manual. Table G4-2 lists all replaceable parts in reference designator order.

EXCHANGE ASSEMBLIES

The components of the Model HP 64243 are supported through the Hewlett-Packard Corporate Parts Center on the Bluestripe Exchange Program. Exchange, factory repaired and tested assemblies are available only on a trade-in basis; therefore, assemblies required for spare parts stock must be ordered using the new assembly part number. Part numbers for both new and exchange assemblies may be found in Tables G4-2 and G4-3.

ABBREVIATIONS

Table G4-1 lists abbreviations used in the parts list, schematics, and throughout the manual. In some cases, two forms of the abbreviation are used: one all in capital letters, and one partial or no capitals. This occurs because the abbreviations in the parts list are always capitals. However, in the schematics and other parts of the manual, other abbreviation forms are used with both lowercase and uppercase letters.

REPLACEABLE PARTS LIST

Tables G4-2 and G4-3 are lists of replaceable parts and are organized as follows:

- a. Chassis-mounted parts in alphanumerical order by reference designation.
- b. Electrical assemblies and their components in alphanumerical order by reference designation.
- c. Miscellaneous.

The information given for each part consists of the following:

- a. The Hewlett-Packard part number and the check digit.
- b. The total quantity (Qty) in the instrument.
- c. The description of the part.

The total quantity for each part is given only once - at the first appearance of the part number in the list.

ORDERING INFORMATION

To order a part listed in the replaceable parts table, quote the Hewlett-Packard part number and check digit, indicate the quantity required, and address the order to the nearest Hewlett-Packard office.

To order a part that is not listed in the replaceable parts table, include the instrument model number, instrument repair number, the description and function of the part, and the number of parts required. Address the order to the nearest Hewlett-Packard office.

DIRECT MAIL ORDER SYSTEM

Within the USA, Hewlett-Packard can supply parts through a direct mail order system. Advantages of using the system are as follows:

- a. Direct ordering and shipment from the HP Parts Center in Mountain View, California.
- b. No maximum or minimum on any mail order (there is a minimum order amount for parts ordered through a local HP office when the orders require billing and invoicing).
- c. Prepaid transportation (there is a small handling charge for each order).
- d. No invoices - to provide these advantages, a check or money order must accompany each order.

Mail-order forms and specific ordering information are available through your local HP office. Addresses and phone numbers are located at the back of this manual.

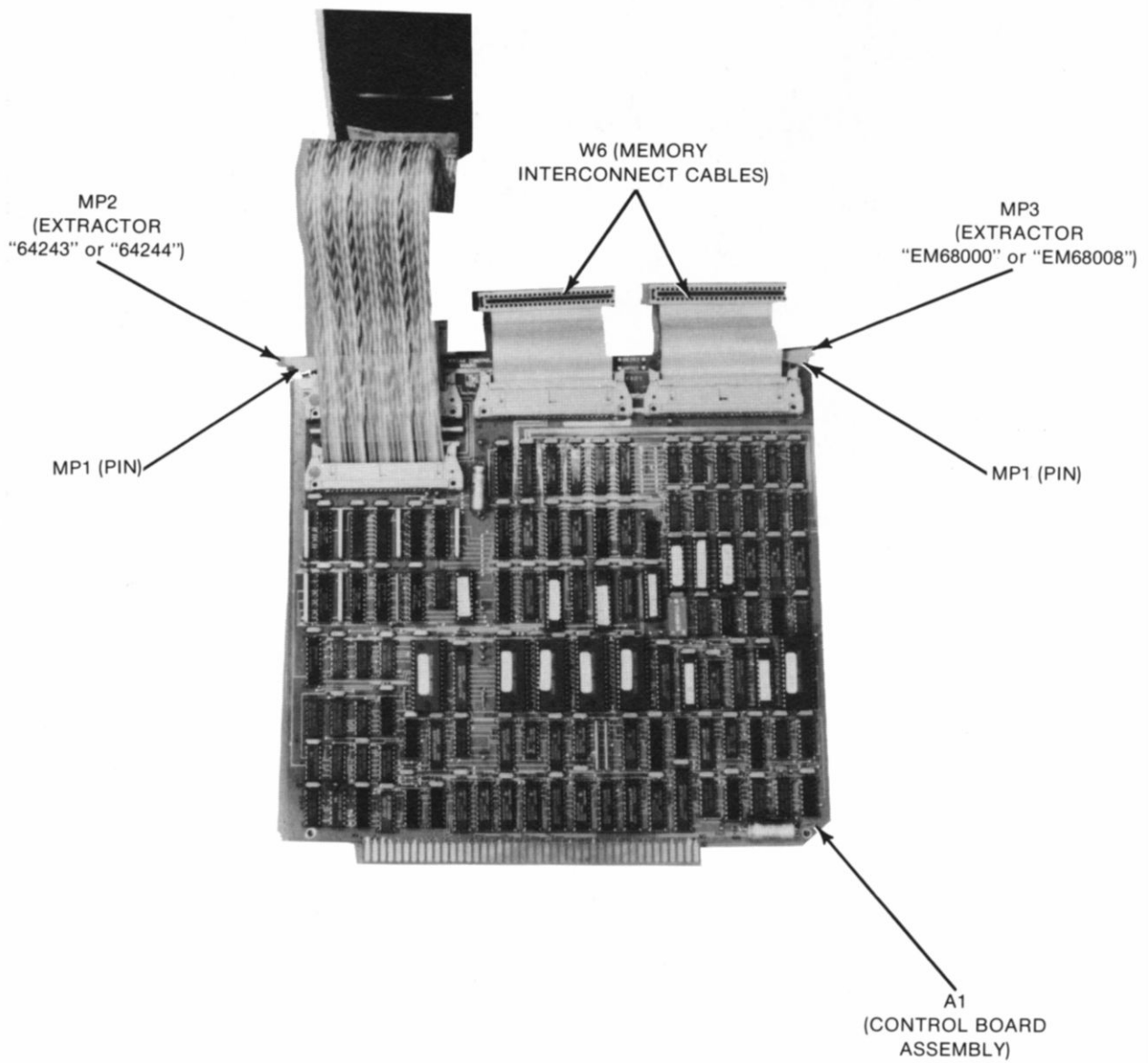


Figure G4-1. Emulator Control Card Parts Locator

Emulator/Analyzer 68000/68008
 Installation and Service Information - Replaceable Parts

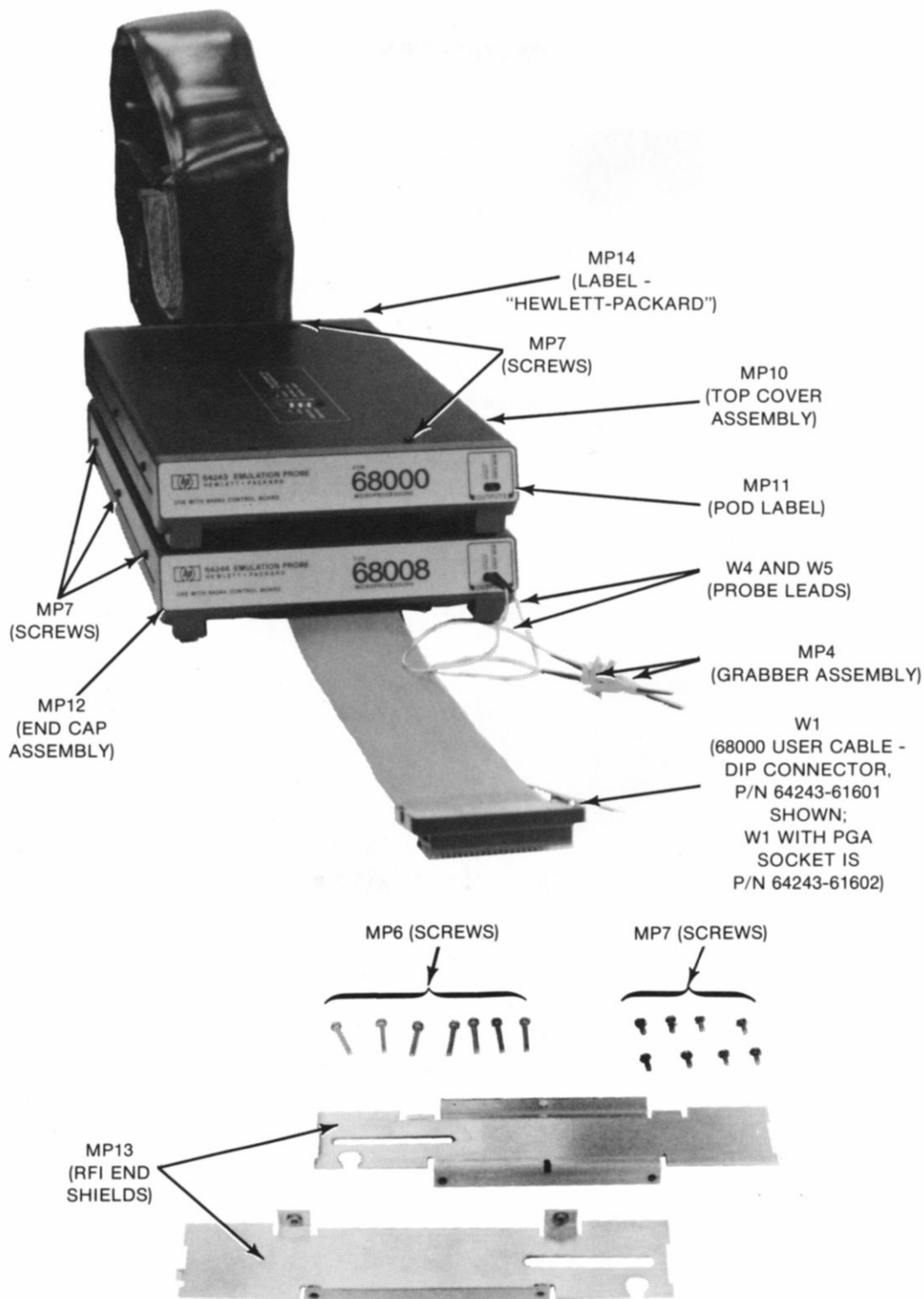


Figure G4-2. Emulator Pod Illustrated Parts Breakdown (Sheet 1 of 3)

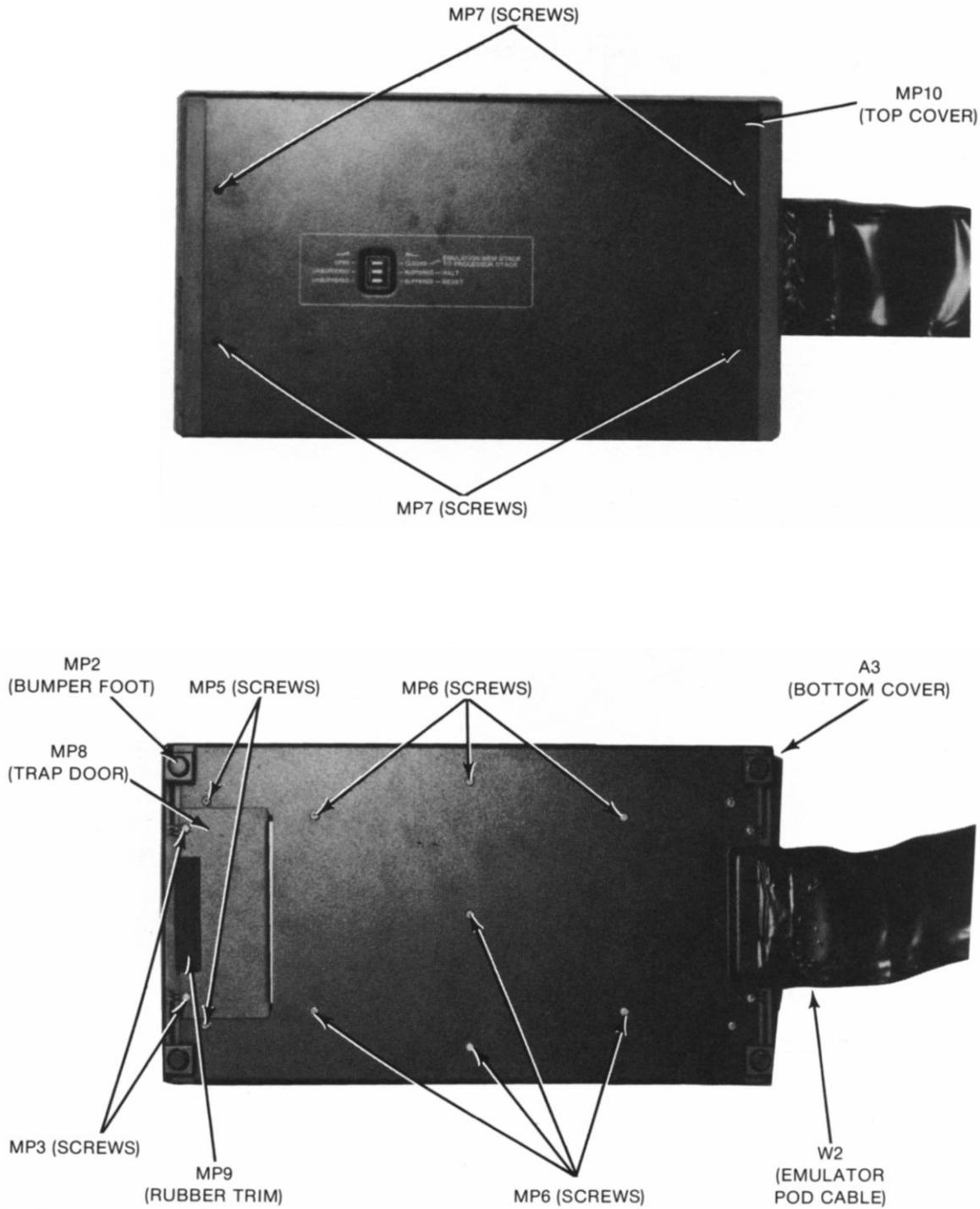


Figure G4-2. Emulator Pod Illustrated Parts Breakdown (Sheet 2 of 3)

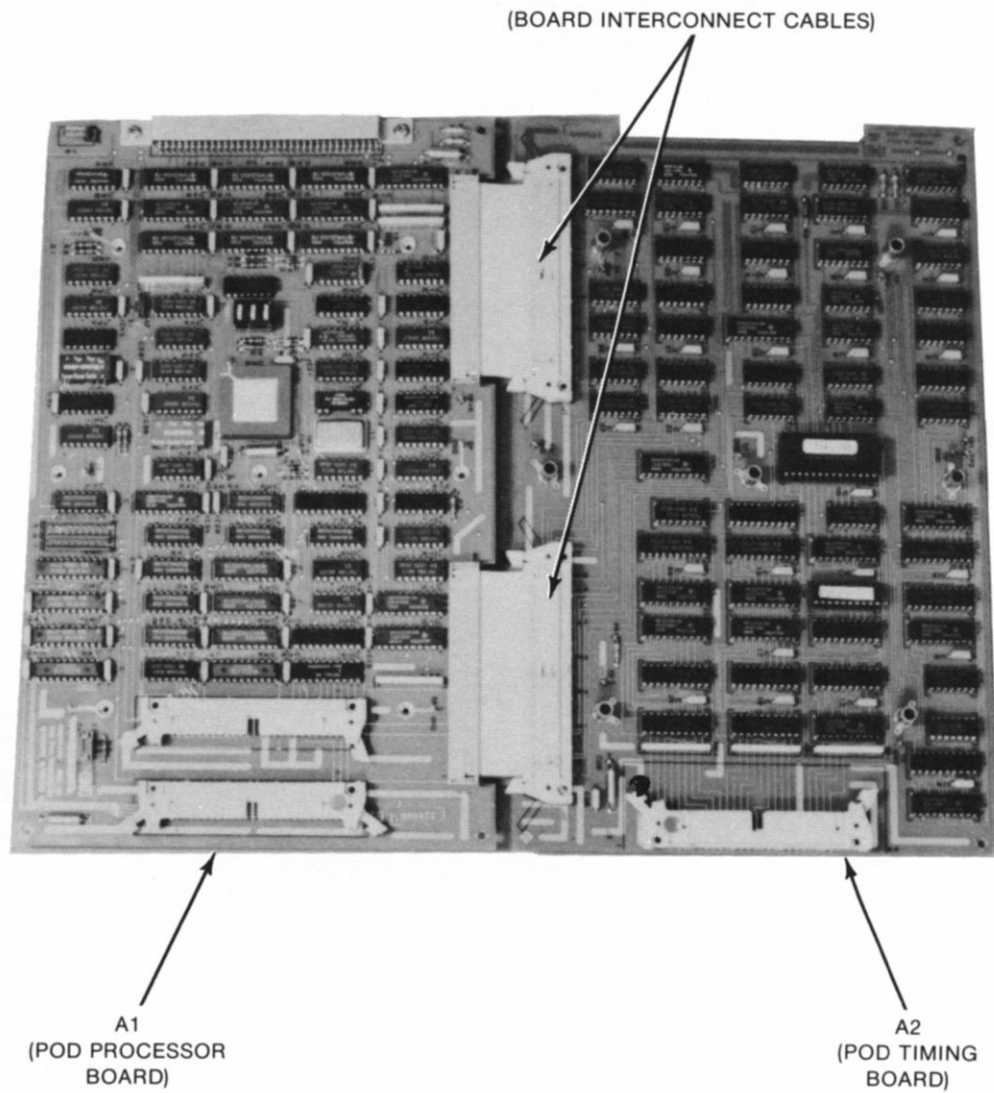


Figure G4-2. Emulator Pod Illustrated Parts Breakdown (Sheet 3 of 3)

Table G4-1. Reference Designators and Abbreviations

REFERENCE DESIGNATORS					
A	= assembly	F	= fuse	MP	= mechanical part
B	= motor	FL	= filter	P	= plug
BT	= battery	IC	= integrated circuit	Q	= transistor
C	= capacitor	J	= jack	R	= resistor
CP	= coupler	K	= relay	RT	= thermistor
CR	= diode	L	= inductor	S	= switch
DL	= delay line	LS	= loud speaker	T	= transformer
DS	= device signaling (lamp)	M	= meter	TB	= terminal board
E	= misc electronic part	MK	= microphone	TP	= test point
				U	= integrated circuit
				V	= vacuum, tube, neon bulb, photocell, etc
				VR	= voltage regulator
				W	= cable
				X	= socket
				Y	= crystal
				Z	= tuned cavity network
ABBREVIATIONS					
A	= amperes	H	= henries	N/O	= normally open
AFC	= automatic frequency control	HDW	= hardware	NOM	= nominal
AMPL	= amplifier	HEX	= hexagonal	NPO	= negative positive zero (zero temperature coefficient)
BFO	= beat frequency oscillator	HG	= mercury	NPN	= negative-positive-negative
BE CU	= beryllium copper	HR	= hour(s)	NRFR	= not recommended for field replacement
BH	= binder head	HZ	= hertz	NSR	= not separately replaceable
BP	= bandpass			OBD	= order by description
BRS	= brass	IF	= intermediate freq	OH	= oval head
BWO	= backward wave oscillator	IMPG	= impregnated	OX	= oxide
		INCD	= incandescent		
CCW	= counter-clockwise	INCL	= include(s)	P	= peak
CER	= ceramic	INS	= insulation(ed)	PC	= printed circuit
CMO	= cabinet mount only	INT	= internal	PF	= picofarads= 10 ⁻¹² farads
COEF	= coefficient			PH BRZ	= phosphor bronze
COM	= common	K	= kilo=1000	PHL	= phillips
COMP	= composition			PIV	= peak inverse voltage
COMPL	= complete	LH	= left hand	PNP	= positive-negative-positive
CONN	= connector	LIN	= linear taper	P/O	= part of
CP	= cadmium plate	LK WASH	= lock washer	POLY	= polystyrene
CRT	= cathode-ray tube	LOG	= logarithmic taper	PORC	= porcelain
CW	= clockwise	LPF	= low pass filter	POS	= position(s)
				POT	= potentiometer
DEPC	= deposited carbon	M	= milli=10 ⁻³	PP	= peak-to-peak
DR	= drive	MEG	= meg=10 ⁶	PT	= point
		MET FLM	= metal film	PWV	= peak working voltage
ELECT	= electrolytic	MET OX	= metallic oxide	RECT	= rectifier
ENCAP	= encapsulated	MFR	= manufacturer	RF	= radio frequency
EXT	= external	MHZ	= mega hertz	RH	= round head or right hand
		MINAT	= miniature		
F	= farads	MOM	= momentary		
FH	= flat head	MOS	= metal oxide substrate		
FIL H	= fillister head	MTG	= mounting		
FXD	= fixed	MY	= "mylar"		
G	= giga (10 ⁹)	N	= nano (10 ⁻⁹)		
GE	= germanium	N/C	= normally closed		
GL	= glass	NE	= neon		
GRD	= grounded	NI PL	= nickel plate		
				TA	= tantalum
				TD	= time delay
				TGL	= toggle
				THD	= thread
				TI	= titanium
				TOL	= tolerance
				TRIM	= trimmer
				TWT	= traveling wave tube
				U	= micro=10 ⁻⁶
				VAR	= variable
				VDCW	= dc working volts
				W/	= with
				W	= watts
				WIV	= working inverse voltage
				WW	= wirewound
				W/O	= without

Emulator/Analyzer 68000/68008
 Installation and Service Information - Replaceable Parts

Table G4-2. Control Card Replaceable Parts

REFERENCE DESIGNATOR	HP PART NUMBER	CD	QTY	DESCRIPTION
A1	64243-66501	9	1	PC BOARD - 680XX EMUL CNTL - NEW
A1	64244-66501	9	1	PC BOARD - 680XX EMUL CNTL - NEW
A1	64243-69501	5	1	PC BOARD - 680XX EMUL CNTL - EXCHANGE
A1	64244-69501	5	1	PC BOARD - 680XX EMUL CNTL - EXCHANGE
MP1	1480-0116	8	2	PIN-GRV .062-IN-DIA .25-IN-LG STL
MP2	64243-85002	5	1	EXTRACTOR - EM68000
MP2	64244-85002	5	1	EXTRACTOR - EM68008
MP3	64243-85004	7	1	EXTRACTOR - PC 64243
MP3	64244-85004	7	1	EXTRACTOR - PC 64244

Emulator/Analyzer 68000/68008
Installation and Service Information - Replaceable Parts

Table G4-3. 68000 Emulator Pod Replaceable Parts

REFERENCE DESIGNATOR	HP PART NUMBER	CD	QTY	DESCRIPTION
A1	64243-66502	0	1	68000 POD PCR BRD - NEW
A1	64243-69502	6	1	68000 POD PCR BRD - EXCHANGE
A2	64243-66505	0	1	680XX POD TMG BRD - NEW
A2	64243-69505	4	1	680XX POD TMG BRD - EXCHANGE
A3	64243-04102	4	1	COVER - BOTTOM
A4	64100-62102	4	1	ASSEMBLY - BRACKET GROUND
MP1	0400-0010	1	1	GROMMET-RND .25-IN-ID .375-IN-GRV-OD
MP2	0403-0179	0	4	BUMPER FOOT-ADH MTG
MP3	2200-0105	4	2	SCREW-MACH 4-40 .312-IN-LG PAN-HD-POZI
MP4	10230-62101	7	2	GRABBER ASSEMBLY
MP5	2200-0103	2	10	SCREW-MACH 4-40 .25-IN-LG PAN-HD-POZI
MP6	2200-0151	0	7	SCREW 4-40 0.750" LG
MP7	2200-0221	5	10	SCREW-MACH 4-40 .25-IN-LG TR-HD-POZI
MP8	64243-00301	7	1	TRAP DOOR - BOTTOM
MP9	0400-0251	3	1	RUBBER TRIM
MP10	64243-04101	3	1	COVER ASSEMBLY - TOP
MP11	64243-94303	8	1	POD LABEL - 68000
MP12	64232-60201	9	2	ENDCAP ASSEMBLY
MP13	64243-00602	8	2	POD END SHIELD - RFI
MP14	7121-1780	0	1	LABEL - GEN 16 BIT
MP15	7121-3182	0	1	LABEL - CABLE CAUTION
W1	64243-61601	8	1	CABLE - USER 68000 - DIP SKT
W1	64243-61602	9	1	CABLE - USER 68000 - PGA SKT
W2	64243-61603	0	1	CABLE - POD BUS
W3	64242-61602	8	1	CABLE - JMPR - 2CDTR
W4	5061-1217	8	1	PROBE LEAD ASSEMBLY (WHITE/BLACK)
W5	5061-1218	9	1	PROBE LEAD ASSEMBLY (WHITE/BROWN)
W6	64263-61602	3	2	CABLE - MEMORY INTERCONNECT
A4	64100-62102	4	1	ASSEMBLY - BRACKET GROUND
A4MP1	1531-0273	7	1	MACHINED PART-SST BAR-CLAMP
A4MP2	2200-0151	0	2	SCREW-MACH 4-40 .75-IN-LG PAN-HD-POZI
A4MP3	2360-0117	6	2	SCREW-MACH 6-32 .375-IN-LG PAN-HD-POZI
A4MP4	2360-0129	0	1	SCREW-MACH 6-32 1-IN-LG PAN-HD-POZI
A4MP5	2420-0001	5	1	NUT-HEX-W/LKWR 6-32-THD .109-IN-THK
A4MP6	3050-0235	3	2	WASHER-FL MTLC NO. 4 .117-IN-ID
A4MP7	64100-01207	2	1	BRKT-GROUND

Emulator/Analyzer 68000/68008
 Installation and Service Information - Replaceable Parts

Table G4-4. 68008 Emulator Pod Replaceable Parts

REFERENCE DESIGNATOR	HP PART NUMBER	CD	QTY	DESCRIPTION
A1	64244-66502	0	1	68008 POD PCR BRD - NEW
A1	64244-69502	6	1	68008 POD PCR BRD - EXCHANGE
A2	64243-66505	0	1	680XX POD TMG BRD - NEW
A2	64243-69505	4	1	680XX POD TMG BRD - EXCHANGE
A3	64243-04102	4	1	COVER - BOTTOM
A4	64100-62102	4	1	ASSEMBLY - BRACKET GROUND
MP1	0400-0010	1	1	GROMMET-RND .25-IN-ID .375-IN-GRV-OD
MP2	0403-0179	0	4	BUMPER FOOT-ADH MTG
MP3	2200-0105	4	2	SCREW-MACH 4-40 .312-IN-LG PAN-HD-POZI
MP4	10230-62101	7	2	GRABBER ASSEMBLY
MP5	2200-0103	2	10	SCREW-MACH 4-40 .25-IN-LG PAN-HD-POZI
MP6	2200-0151	0	7	SCREW 4-40 0.750" LG
MP7	2200-0221	5	10	SCREW-MACH 4-40 .25-IN-LG TR-HD-POZI
MP8	64243-00301	7	1	TRAP DOOR - BOTTOM
MP9	0400-0251	3	1	RUBBER TRIM
MP10	64243-04101	3	1	COVER ASSEMBLY - TOP
MP11	64244-94303	8	1	POD LABEL - 68008
MP12	64232-60201	9	2	ENDCAP ASSEMBLY
MP13	64244-00602	8	2	POD END SHIELD - RFI
MP14	7121-1780	0	1	LABEL - GEN 16 BIT
MP15	7121-3182	0	1	LABEL - CABLE CAUTION
W1	64244-61601	8	1	CABLE - USER 68008 - DIP SKT
W2	64243-61603	0	1	CABLE - POD BUS
W3	64242-61602	8	1	CABLE - JMPR - 2CDTR
W4	5061-1217	8	1	PROBE LEAD ASSEMBLY (WHITE/BLACK)
W5	5061-1218	9	1	PROBE LEAD ASSEMBLY (WHITE/BROWN)
W6	64263-61602	3	2	CABLE - MEMORY INTERCONNECT
A4	64100-62102	4	1	ASSEMBLY - BRACKET GROUND
A4MP1	1531-0273	7	1	MACHINED PART-SST BAR-CLAMP
A4MP2	2200-0151	0	2	SCREW-MACH 4-40 .75-IN-LG PAN-HD-POZI
A4MP3	2360-0117	6	2	SCREW-MACH 6-32 .375-IN-LG PAN-HD-POZI
A4MP4	2360-0129	0	1	SCREW-MACH 6-32 1-IN-LG PAN-HD-POZI
A4MP5	2420-0001	5	1	NUT-HEX-W/LKWR 6-32-THD .109-IN-THK
A4MP6	3050-0235	3	2	WASHER-FL MTLC NO. 4 .117-IN-ID
A4MP7	64100-01207	2	1	BRKT-GROUND

Appendix G

INSTALLATION AND SERVICE INFORMATION

Section G5

REMOVAL AND REPLACEMENT PROCEDURES

This section provides emulator pod disassembly and disassembly procedures.

EMULATOR POD DISASSEMBLY

To disassemble the Emulator Pod for troubleshooting and repair, use the following procedure:

- a. Remove the emulator from the development station, using the procedure described in Section G2.
- b. Remove three screws (MP7) on each side of the emulator pod See Figure G5-1.
- c. Remove four screws (MP7) from the top cover of the emulator pod See Figure G5-2.
- d. Turn over the emulator pod and remove two screws (MP5) located toward the back end of the pod on each side of the pod bus cable (W2). See Figure G5-3.
- e. Remove two screws (MP3) and the trap door (MP8). See Figure G5-3.
- f. To remove user cable (W1) from emulator pod board A1, carefully pull the female block connector of the user cable away from the matching male block connector located beneath the trap door opening.
- g. Now turn the emulator pod over and carefully pull the top cover away from the pod's bottom cover, making sure to disconnect the two-conductor jumper cable (W3) from the board nearest the top cover as you do so. See Figure G5-4.
- h. Further disassembly of the pod top cover may be accomplished by removing two screws (MP5) that hold each endcap to the top cover. See Figure G5-3.
- i. The metal RFI shields are removed from the pod bottom cover by removing two screws (MP5) that secure each shield to the cover. See Figure G5-3.
- j. Remove the emulator PC boards (A1 & A2) by removing seven screws (MP6) that secure the board spacers to the pod bottom cover. See Figure G5-3.

k. To disconnect the pod bus cable (W2, Figure G5-5) pull outward on the latching tips of the three connectors which hold the multi-colored ribbon cables to the pod boards. The female block connectors terminating the ends of the ribbon cables should snap free.

l. To disconnect emulator pod boards (A1 and A2) from each other, pull outward on the latching tips of the connectors on the A2 board which hold the ribbon cables from the A1 board. See Figure G5-5.

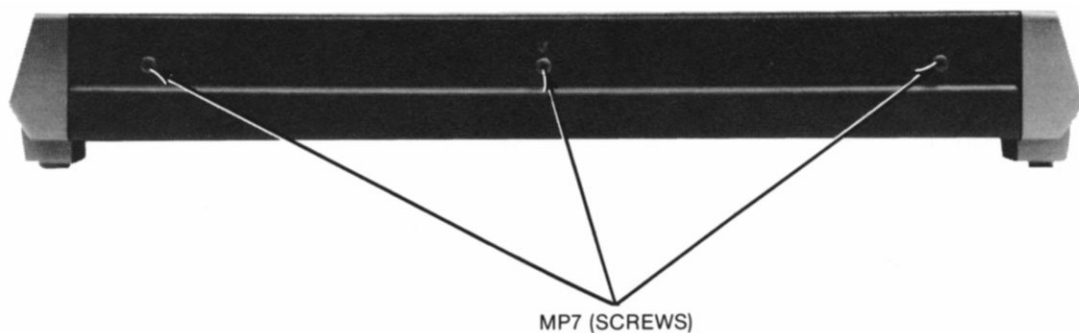


Figure G5-1. Emulator Pod Side View

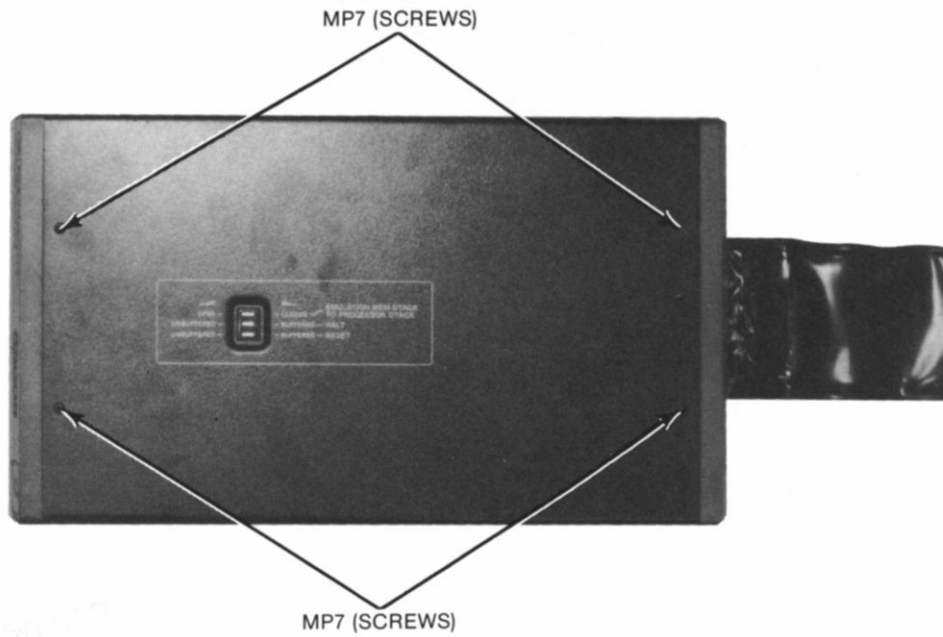


Figure G5-2. Emulator Pod Top View

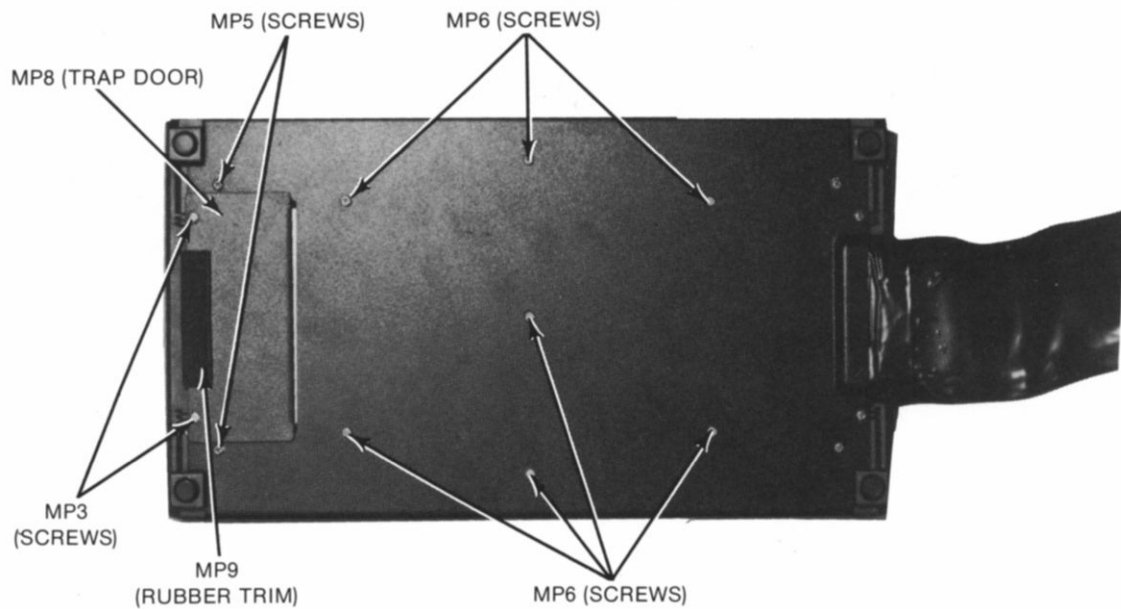


Figure G5-3. Emulator Pod Bottom View

Emulator/Analyzer 68000/68008
Installation and Service Information - Removal and Replacement Procedures

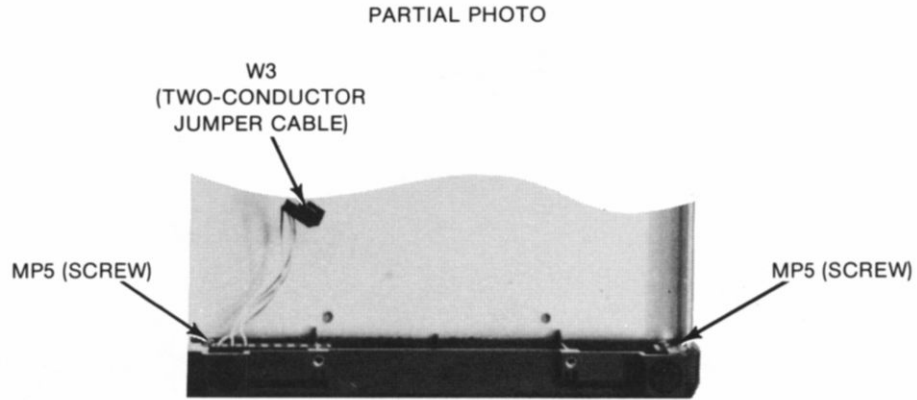


Figure G5-4. Emulator Pod - Inside Top Cover

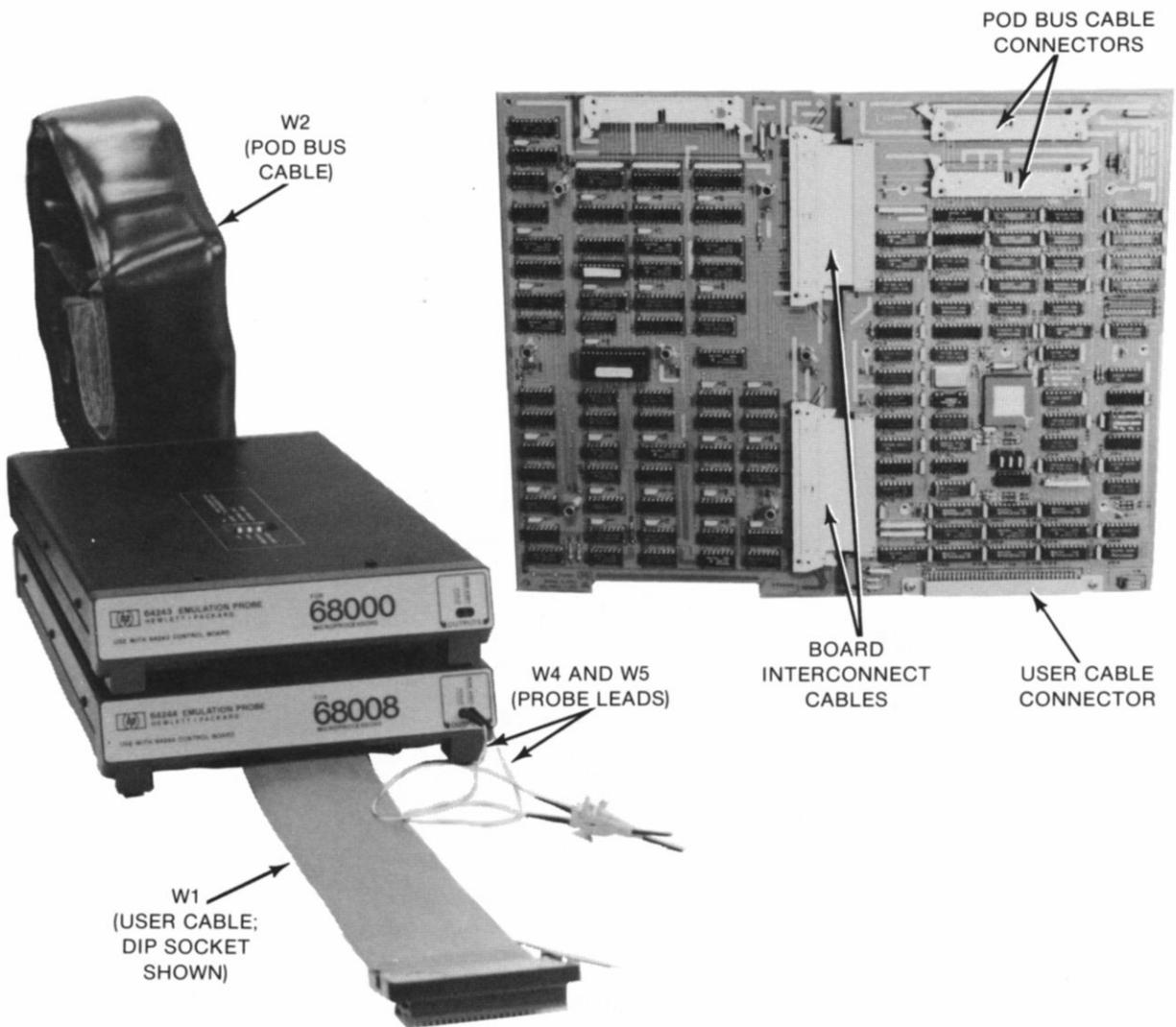


Figure G5-5. Pod Bus and Board Interconnect Cables

EMULATOR POD REASSEMBLY

To reassemble the emulator pod, proceed as follows:

a. Connect user cable (W1) to the emulator pod A1 microprocessor board as follows:

a. Reconnect the A1 board and A2 board by joining the female block connectors from the A1 board interconnect cables to the male pin connectors on the A2 board. See Figure G5-5. Make sure that the latching tips of the connectors are open before joining the connectors, then press the two connector sets together until the latching tips snap over the top of the cable connectors.

Fold the cables over so that the component side of one board is facing in the opposite direction of the component side of the other board.

b. Connect the three connectors on the pod bus cable (W2, Figure G5-5) to the emulator pod circuit boards. Each multi-colored ribbon cable can only connect to one of the matching connectors at the end of each pod board.

c. Reattach the pod boards to the bottom cover by placing the folded board set inside the bottom cover and inserting seven screws (MP6, Figure G5-3) from the outside of the cover. The A2 board should be closest to the bottom cover.

d. Replace the RFI end shields. One of these attaches to each end of the pod bottom cover with the slotted end pointing away from the bottom cover and the standoff end towards the bottom cover (the standoffs provide room for the user cable and pod bus cable to be sandwiched in between the shield and the bottom cover).

e. If the emulator pod endcaps were removed from the top cover, reattach them using four screws (MP5, Figure G5-4).

f. Reattach the pod top cover to the bottom cover as follows:

1. Reconnect the two-conductor cable (W3) from inside the top cover to the two-pin connector on pod board assembly A1.

2. Seat the grommet (MP1, Figure G5-3) from the cable into the RFI end shield notch near one end of the shield.

3. Place the pod top cover over the pod bottom cover; insert and tighten the four screws (MP5, Figure G5-3) that are nearest the pod bumper feet.

4. Insert and tighten the four screws (MP5, Figure G5-2) which attach the RFI end shields to the pod top cover.

5. Insert and tighten the four screws (MP7, Figure G5-2) that secure the top cover to the A1 board assembly.

6. Insert and tighten the six screws (MP7, Figure G5-1) which hold each side of the pod top cover to the pod bottom cover.

NOTES

Appendix G

INSTALLATION AND SERVICE INFORMATION

Section G6

ELECTRICAL AND ENVIRONMENTAL CHARACTERISTICS

This section provides electrical, environmental, and performance characteristics for the 68000/68008 Emulator.

USER INTERFACE

The following lists the electrical and mechanical characteristics of the 68000 Emulator when installed in a user's target system:

Processor compatibility: compatible with Motorola 68000/68008 microprocessors and any other microprocessors that comply with the specifications of the Motorola 68000/68008.

Maximum clock speed - Model HP 64243: 12.5 MHz

Maximum clock speed - Model HP 64244: 10.0 MHz

Inputs/Outputs: one ALS load plus approximately 40 pF of shunt capacitance.

Target system power: approximately 15 mA of current is drawn from the target system's +5 V supply for in-circuit sensing; all other emulator power requirements are provided for by the development station.

POWER SUPPLY REQUIREMENTS

The Emulator Control Card combined with the HP 64243/64244 Emulator Pod places the following demands on the 64000 development station power supply:

+5V -- 6.5 amperes typical (Control Card and Emulator Pod combined).



The following precautions should be taken while using Hewlett-Packard Emulator Pods. Damage to the emulator circuitry may result if these precautions are not observed.

POWER DOWN TARGET SYSTEM.

Turn off power to the user target system and the emulation development station before inserting the user plug to avoid circuit damage resulting from voltage transients or mis-insertion of the user plug.

VERIFY USER PLUG ORIENTATION.

Make certain that Pin 1 of the target system microprocessor socket and Pin 1 of the user plug are properly aligned before inserting the user plug in the socket. Failure to do so may result in damage to the emulator circuitry.

PROTECT AGAINST STATIC DISCHARGE.

The emulator pod contains devices which are susceptible to damage by static discharge. Therefore, operators should take precautionary measures before handling the user plug to avoid emulator damage.



OPERATING ENVIRONMENT

The HP 64243/64244 may be operated in environments within the following limits:

Temperature.....	0 ^o to +40 ^o C
Humidity.....	5% to 80% relative humidity
Altitude.....	4 600 m (15 000 ft)

It should be protected from temperature extremes which could cause condensation within the instrument.

STORAGE AND SHIPMENT

Environment

The HP 64243/64244 may be stored or shipped in environments within the following limits:

Temperature.....	-40 ^o to +75 ^o C
Humidity.....	5% to 80% relative humidity
Altitude.....	15 000 m (50 000 ft)

PERFORMANCE CHARACTERISTICS

The tables and timing diagrams on the following pages provide specifications for the Motorola 68000 and 68008 Microprocessors as well Performance characteristics for the respective Hewlett-Packard Emulators.

Timing diagrams (figures G6-1 through G6-12) and tables G6-3 and G6-4 are reproduced by permission © 1985 Motorola Inc. This material shall not be reproduced without the written consent of Motorola Inc.

Table G6-1. Timing Comparison - MC68000R12 vs 12.5 Mhz Emulator

		68000R12		12.5Mhz Pod		
		min	max	min	max	units
1	Clock Period	80	250	80	250	ns
2	Clock Width Low	35	125	35	125	ns
3	Clock Width High	35	125	35	125	ns
4	Clock Fall Time	-	5	-	5	ns
5	Clock Rise Time	-	5	-	5	ns
6	Clock Low to Address Valid	-	55	-	65	ns
6A	Clock High to FC Valid	-	55	-	66	ns
7	Clock High to Address/Data High Impedance (Maximum)	-	60	-	***	ns
8	Clock High to Address/FC Invalid (Minimum)	0	-	3	-	ns
9	Clock High to /AS,/DS Low Address Valid to /AS,/DS Low (Read)/ /AS Low (Write)	0	55	3	65	ns
11A	FC Valid to /AS,/DS Low (Read)/ /AS Low (Write)	0	-	-7	-	ns
12	Clock Low to /AS,/DS High	40	-	32	-	ns
13	/AS,/DS High to Address/FC Invalid	-	50	-	60	ns
14	/AS,/DS Width Low (Read)/ /AS Low (Write)	10	-	3	-	ns
14A	/DS Width Low (Write)	160	-	160	-	ns
15	/AS,/DS Width High	80	-	80	-	ns
16	Clock High to Control Bus High Impedance	65	-	65	-	ns
17	/AS,/DS High to R/W High (Read)	-	60	-	***	ns
18	Clock High to R/W High	10	-	10	-	ns
20	Clock High to R/W Low (Write)	0	60	3	70	ns
20A	/AS Low to R/W Valid (Write)	-	60	-	70	ns
21	Address Valid To R/W Low (Write)	-	20	-	20	ns
21A	FC Valid to R/W Low (Write)	0	-	-7	-	ns
22	R/W Low to /DS Low (Write)	30	-	22	-	ns
23	Clock Low To Data Out Valid (Write)	30	-	30	-	ns
25	/AS,/DS High to Data Out Invalid (Write)	-	55	-	65	ns
26	Data Out Valid to /DS Low (Write)	15	-	8	-	ns
27	Data In to Clock Low (Setup Time on Read)	15	-	8	-	ns
28	/AS,/DS High to /DTACK High	10	-	20	-	ns
29	/AS,/DS High to Data In Invalid (Hold Time on Read)	0	150	0	137	ns
30	/AS,/DS High to /BERR High	0	-	22	-	ns
31	/DTACK Low to Data In (Setup Time)	0	-	0	-	ns
32	/HALT and /RESET Input Transition Time	-	50	-	62	ns
		0	200	0	200	ns

Table G6-1. Timing Comparison - MC68000R12 vs 12.5 Mhz Emulator (Cont'd)

		68000R12		12.5Mhz Pod		
		min	max	min	max	units
33	Clock High to /BG Low	-	50	-	88	ns
34	Clock High to /BG High	-	50	-	54	ns
35	/BR Low to /BG Low		70ns		128ns	
		1.5	+3.5	1.5	+3.5	cp
36	/BR High to /BG High		70ns		128ns	
		1.5	+3.5	1.5	+3.5	cp
37	/BGACK Low to /BG High		70ns		121ns	
		1.5	+3.5	1.5	+3.5	cp
37A	/BGACK Low to /BR High		1.5		1.5	
		20	clks	27	clks	ns
38	/BG Low to Control, Address, Data Bus High Impedance (/AS High)	-	60	-	60	ns
39	/BG Width High	1.5	-	1.5	-	cp
40	Clock Low to /VMA Low	-	70	-	80	ns
41	Clock Low to E Transition	-	45	-	52	ns
42	E Output Rise and Fall Time	-	25	-	3	ns
43	/VMA Low to E High	90	-	83	-	ns
44	/AS,/DS High to /VPA High	0	70	0	59	ns
45	E Low to Control, Address Bus Invalid (Address Hold Time)	10	-	3	-	ns
46	/BGACK Width Low	-	1.5	-	1.5	
47	Asynchronous Input Setup Time	20	-	20	-	ns
48	/BERR Low to /DTACK Low	20	-	30	-	ns
49	/AS,/DS High to E Low	-45	45	-53	48	ns
50	E Width High	280	-	280	-	ns
51	E Width Low	440	-	440	-	ns
53	Clock High to Data Out Invalid	0	-	3	-	ns
54	E Low to Data Out Invalid	15	-	12	-	ns
55	R/W to Data Bus Driven	10	-	3	-	ns
56	/HALT,/RESET Pulse Width	10	-	10	-	cp
57	/BGACK High to Control Bus Driven	1.5	-	1.5	-	cp
58	/BG High to Control Bus Driven	1.5	-	1.5	-	cp

**** indicates timing spec not applicable to emulator. Note that timing modification due to emulator is worst case and is typically very close to processor timing.

cp = clock period

Table G6-2. Timing Comparison - MC68008L10 vs 10.0 Mhz Emulator

		68000L10		10.5Mhz Pod		
		min	max	min	max	units
1	Clock Period	100	500	100	500	ns
2	Clock Width Low	45	250	45	250	ns
3	Clock Width High	45	250	45	250	ns
4	Clock Fall Time	-	10	-	10	ns
5	Clock Rise Time	-	10	-	10	ns
6	Clock Low to Address Valid	-	60	-	70	ns
6A	Clock High to FC Valid	-	60	-	71	ns
7	Clock High to Address/Data High Impedance (Maximum)	-	70	-	***	ns
8	Clock High to Address/FC Invalid (Minimum)	0	-	3	-	ns
9	Clock High to /AS,/DS Low	0	55	3	65	ns
11	Address Valid to /AS,/DS Low (Read)/ /AS Low (Write)	20	-	13	-	ns
11A	FC Valid to /AS,/DS Low (Read)/ /AS Low (Write)	50	-	42	-	ns
12	Clock Low to /AS,/DS High	-	35	-	45	ns
13	/AS,/DS High to Address/FC Invalid	20	-	13	-	ns
14	/AS,/DS Width Low (Read)/ /AS Low (Write)	195	-	195	-	ns
14A	/DS Width Low (Write)	95	-	95	-	ns
15	/AS,/DS Width High	105	-	105	-	ns
16	Clock High to Control Bus High Impedance	-	70	-	***	ns
17	/AS,/DS High to R/W High (Read)	20	-	20	-	ns
18	Clock High to R/W High	0	40	3	50	ns
20	Clock High to R/W Low (Write)	-	40	-	50	ns
20A	/AS Low to R/W Valid (Write)	-	20	-	20	ns
21	Address Valid To R/W Low (Write)	0	-	-7	-	ns
21A	FC Valid to R/W Low (Write)	50	-	42	-	ns
22	R/W Low to /DS Low (Write)	50	-	50	-	ns
23	Clock Low To Data Out Valid (Write)	-	55	-	65	ns
25	/AS,/DS High to Data Out Invalid (Write)	20	-	13	-	ns
26	Data Out Valid to /DS Low (Write)	20	-	13	-	ns
27	Data In to Clock Low (Setup Time on Read)	10	-	20	-	ns
28	/AS,/DS High to /DTACK High	0	190	0	177	ns
29	/AS,/DS High to Data In Invalid (Hold Time on Read)	0	-	17	-	ns
30	/AS,/DS High to /BERR High	0	-	0	-	ns
31	/DTACK Low to Data In (Setup Time)	-	65	-	77	ns
32	/HALT and /RESET Input Transition Time	0	200	0	200	ns

Table G6-2. Timing Comparison - MC68008L10 vs 10.0 Mhz Emulator (Cont'd)

		68000L10		10.5Mhz Pod		
		min	max	min	max	units
33	Clock High to /BG Low	-	40	-	78	ns
34	Clock High to /BG High	-	40	-	54	ns
35	/R Low to /BG Low		80ns		129ns	
36	/BR High to /BG High	1.5	+3.5	1.5	+3.5	cp
			80ns		129ns	
		1.5	+3.5	1.5	+3.5	cp
38	/BG Low to Control, Address, Data Bus High Impedance (/AS High)	-	70	-	60	ns
39	/BG Width High	1.5	-	1.5	-	cp
41	Clock Low to E Transition	-	50	-	57	ns
42	E Output Rise and Fall Time	-	15	-	3	ns
44	/AS,/DS High to /VPA High	0	90		79	ns
45	E Low to Control, Address Bus Invalid (Address Hold Time)	10	-	3	-	ns
47	Asynchronous Input Setup Time	10	-	10	-	ns
48	/BERR Low to /DTACK Low	20	-	30	-	ns
49	/AS,/DS High to E Low	-80	80	-88	83	ns
50	E Width High	350	-	350	-	ns
51	E Width Low	550	-	550	-	ns
53	Clock High to Data Out Invalid	0	-	3	-	ns
54	E Low to Data Out Invalid	20	-	17	-	ns
55	R/W to Data Bus Driven	20	-	13	-	ns
56	/HALT,/RESET Pulse Width	10	-	10	-	cp
58	/BG High to Control Bus Driven	1.5	-	1.5	-	cp

**** indicates timing spec not applicable to emulator. Note that timing modification due to emulator is worst case and is typically very close to processor timing.

cp = clock period

Table G6-3. Electrical Specifications - Read and Write Cycles (68000)(Vcc=5.0 Vdc +/- 5%; GND=0 Vdc; $T_A = T_L$ to T_H ; see Figures G6-1 and G6-2)

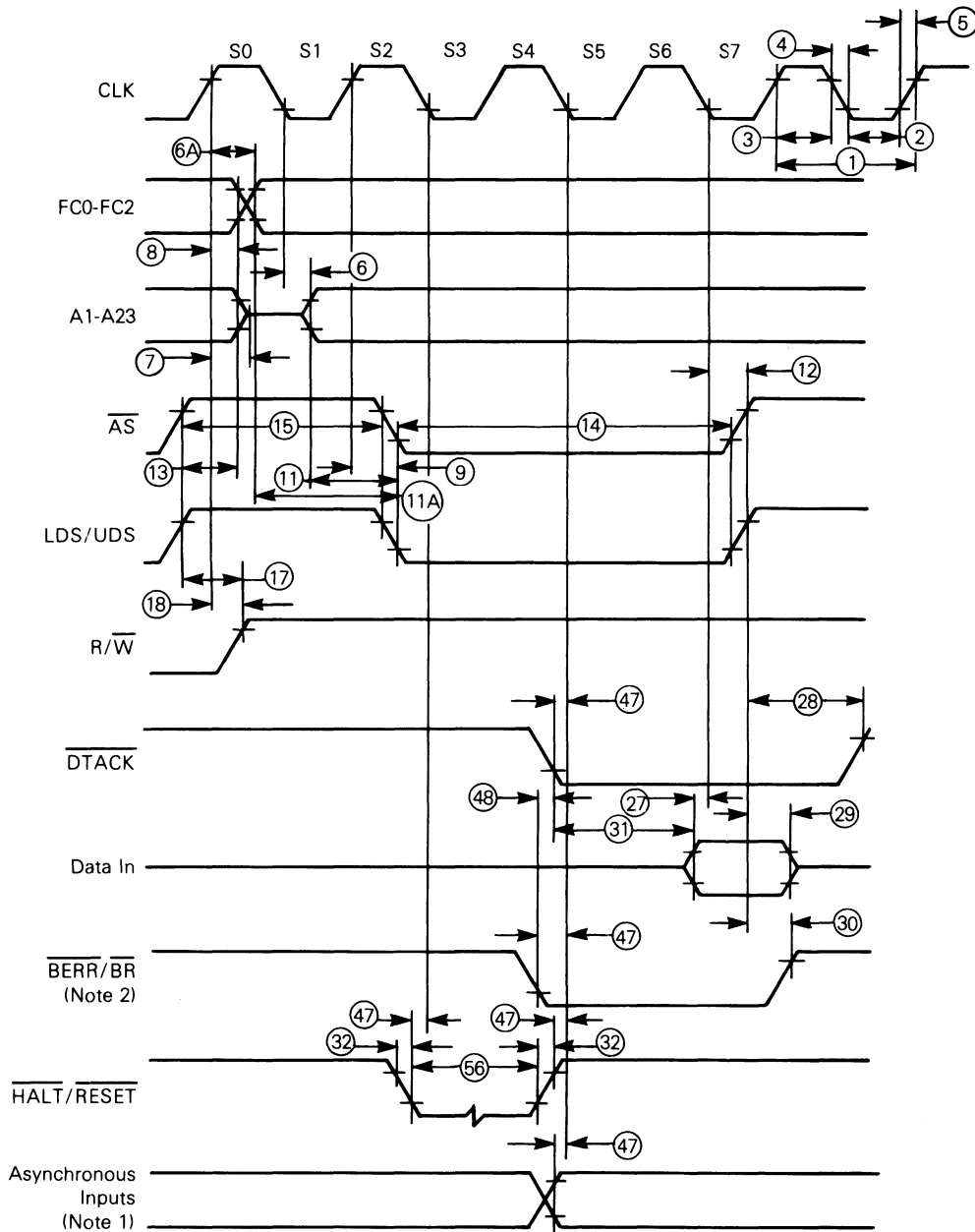
Num.	Characteristic	Symbol	8 MHz		10 MHz		12.5 MHz		Unit
			Min	Max	Min	Max	Min	Max	
1	Clock Period	t_{cyc}	125	250	100	250	80	250	ns
2	Clock Width Low	t_{CL}	55	125	45	125	35	125	ns
3	Clock Width High	t_{CH}	55	125	45	125	35	125	ns
4	Clock Fall Time	t_{Cf}	–	10	–	10	–	5	ns
5	Clock Rise Time	t_{Cr}	–	10	–	10	–	5	ns
6	Clock Low to Address Valid	t_{CLAV}	–	70	–	60	–	55	ns
6A	Clock High to FC Valid	t_{CHFCV}	–	70	–	60	–	55	ns
7	Clock High to Address, Data Bus High Impedance (Maximum)	t_{CHADZ}	–	80	–	70	–	60	ns
8	Clock High to Address, FC Invalid (Minimum)	t_{CHAFI}	0	–	0	–	0	–	ns
9 ¹	Clock High to \overline{AS} , \overline{DS} Low	t_{CHSL}	0	60	0	55	0	55	ns
11 ²	Address Valid to \overline{AS} , \overline{DS} Low (Read)/ AS Low (Write)	t_{AVSL}	30	–	20	–	0	–	ns
11A ^{2,7}	FC Valid to \overline{AS} , \overline{DS} Low (Read)/ AS Low (Write)	t_{FCVSL}	60	–	50	–	40	–	ns
12 ¹	Clock Low to \overline{AS} , \overline{DS} High	t_{CLSH}	–	70	–	55	–	50	ns
13 ²	\overline{AS} , \overline{DS} High to Address/FC Invalid	t_{SHAFI}	30	–	20	–	10	–	ns
14 ^{2,5}	\overline{AS} , \overline{DS} Width Low (Read)/ \overline{AS} Low (Write)	t_{SL}	240	–	195	–	160	–	ns
14A ²	\overline{DS} Width Low (Write)	t_{DSL}	115	–	95	–	80	–	ns
15 ²	\overline{AS} , \overline{DS} Width High	t_{SH}	150	–	105	–	65	–	ns
16	Clock High to Control Bus High Impedance	t_{CHCZ}	–	80	–	70	–	60	ns
17 ²	\overline{AS} , \overline{DS} High to R/ \overline{W} High (Read)	t_{SHRH}	40	–	20	–	10	–	ns
18 ¹	Clock High to R/ \overline{W} High	t_{CHRH}	0	70	0	60	0	60	ns
20 ¹	Clock High to R/ \overline{W} Low (Write)	t_{CHRL}	–	70	–	60	–	60	ns
20A ⁸	\overline{AS} Low to R/ \overline{W} Valid (Write)	t_{ASRV}	–	20	–	20	–	20	ns
21 ²	Address Valid to R/ \overline{W} Low (Write)	t_{AVRL}	20	–	0	–	0	–	ns
21A ^{2,7}	FC Valid to R/ \overline{W} Low (Write)	t_{FCVRL}	60	–	50	–	30	–	ns
22 ²	R/ \overline{W} Low to \overline{DS} Low (Write)	t_{RLSL}	80	–	50	–	30	–	ns
23	Clock Low to Data Out Valid (Write)	t_{CLDO}	–	70	–	55	–	55	ns
25 ²	\overline{AS} , \overline{DS} High to Data Out Invalid (Write)	t_{SHDOI}	30	–	20	–	15	–	ns
26 ²	Data Out Valid to \overline{DS} Low (Write)	t_{DOSL}	30	–	20	–	15	–	ns
27 ⁶	Data In to Clock Low (Setup Time on Read)	t_{DICL}	15	–	10	–	10	–	ns
28 ^{2,5}	\overline{AS} , \overline{DS} High to \overline{DTACK} High	t_{SHDAH}	0	245	0	190	0	150	ns
29	\overline{AS} , \overline{DS} High to Data In Invalid (Hold Time on Read)	t_{SHDII}	0	–	0	–	0	–	ns
30	\overline{AS} , \overline{DS} High to \overline{BERR} High	t_{SHBEH}	0	–	0	–	0	–	ns
31 ^{2,6}	\overline{DTACK} Low to Data In (Setup Time)	t_{DALDI}	–	90	–	65	–	50	ns
32	\overline{HALT} and \overline{RESET} Input Transition Time	$t_{RHr,f}$	0	200	0	200	0	200	ns
33	Clock High to \overline{BG} Low	t_{CHGL}	–	70	–	60	–	50	ns
34	Clock High to \overline{BG} High	t_{CHGH}	–	70	–	60	–	50	ns
35	\overline{BR} Low to \overline{BG} Low	t_{BRLGL}	1.5	90 ns + 3.5	1.5	80 ns + 3.5	1.5	70 ns + 3.5	Clk.Per.

Table G6-3. Electrical Specifications - Read and Write Cycles (68000) (Cont'd)

Num.	Characteristic	Symbol	8 MHz		10 MHz		12.5 MHz		Unit
			Min	Max	Min	Max	Min	Max	
36 ⁹	\overline{BR} High to \overline{BG} High	tBRHGH	1.5	90 ns + 3.5	1.5	80 ns + 3.5	1.5	70 ns + 3.5	Clk. Per.
37	\overline{BGACK} Low to \overline{BG} High	tGALGH	1.5	90 ns + 3.5	1.5	80 ns + 3.5	1.5	70 ns + 3.5	Clk. Per.
37A ¹⁰	\overline{BGACK} Low to \overline{BR} High	tGALBRH	20	1.5 Clocks	20	1.5 Clocks	20	1.5 Clocks	ns
38	\overline{BG} Low to Control, Address, Data Bus High Impedance (\overline{AS} High)	tGLZ	—	80	—	70	—	60	ns
39	\overline{BG} Width High	tGH	1.5	—	1.5	—	1.5	—	Clk. Per.
40	Clock Low to \overline{VMA} Low	tCLVML	—	70	—	70	—	70	ns
41	Clock Low to E Transition	tCLET	—	70	—	55	—	45	ns
42	E Output Rise and Fall Time	t _{Er,f}	—	25	—	25	—	25	ns
43	\overline{VMA} Low to E High	tVMLEH	200	—	150	—	90	—	ns
44	\overline{AS} , \overline{DS} High to \overline{VPA} High	tSHVPH	0	120	0	90	0	70	ns
45	E Low to Control, Address Bus Invalid (Address Hold Time)	tELCAI	30	—	10	—	10	—	ns
46	\overline{BGACK} Width Low	tGAL	1.5	—	1.5	—	1.5	—	Clk. Per.
47 ⁶	Asynchronous Input Setup Time	tASI	20	—	20	—	20	—	ns
48 ³	\overline{BERR} Low to \overline{DTACK} Low	tBELDAL	20	—	20	—	20	—	ns
49 ¹¹	\overline{AS} , \overline{DS} High to E Low	tSHEL	— 70	70	— 55	55	— 45	45	ns
50	E Width High	tEH	450	—	350	—	280	—	ns
51	E Width Low	tEL	700	—	550	—	440	—	ns
53	Clock High to Data Out Invalid	tCHDOI	0	—	0	—	0	—	ns
54	E Low to Data Out Invalid	tELDOI	30	—	20	—	15	—	ns
55	R/ \overline{W} to Data Bus Driven	tRLDBD	30	—	20	—	10	—	ns
56 ⁴	$\overline{HALT/RESET}$ Pulse Width	tHRPW	10	—	10	—	10	—	Clk. Per.
57	\overline{BGACK} High to Control Bus Driven	tGABD	1.5	—	1.5	—	1.5	—	Clk. Per.
58 ⁹	\overline{BG} High to Control Bus Driven	tGHBD	1.5	—	1.5	—	1.5	—	Clk. Per.

NOTES:

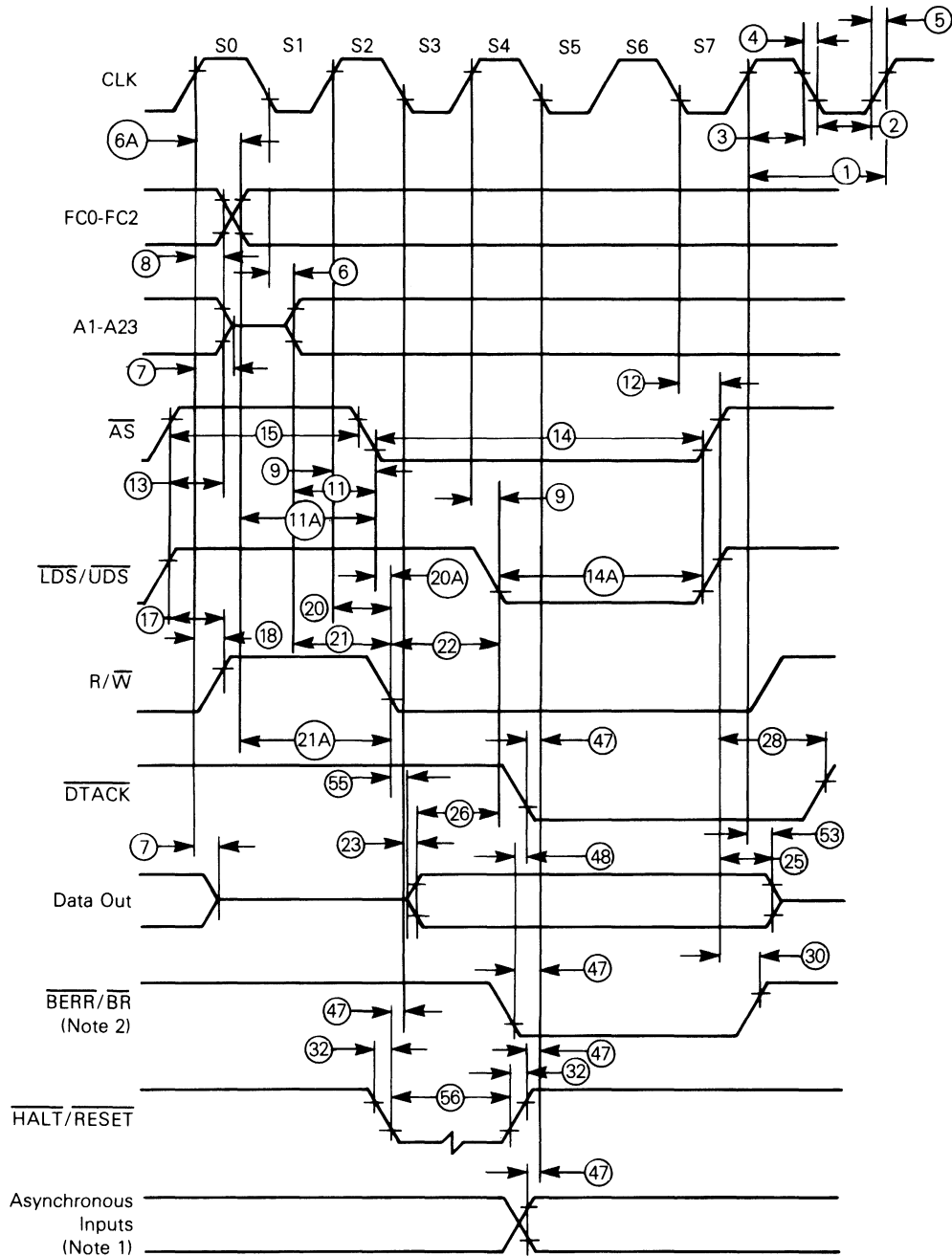
1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the value given in the maximum columns.
2. Actual value depends on clock period.
3. If #47 is satisfied for both \overline{DTACK} and \overline{BERR} , #48 may be 0 nanoseconds.
4. For power up, the MPU must be held in RESET state for 100 ms to allow stabilization of on-chip circuitry. After the system is powered up, #56 refers to the minimum pulse width required to reset the system.
5. #14, #14A, and #28 are one clock period less than the given number for T6E, BF4, and R9M mask sets.
6. If the asynchronous setup time (#47) requirements are satisfied, the \overline{DTACK} low-to-data setup time (#31) requirement can be ignored. The data must only satisfy the data-in clock-low setup time (#27) for the following cycle.
7. For T6E, BF4, and R9M mask set #11A timing equals #11, and #21A equals #21. #20A may be 0 for T6E, BF4, and R9M mask sets.
8. When \overline{AS} and R/ \overline{W} are equally loaded ($\pm 20\%$), subtract 10 nanoseconds from the values given in these columns.
9. The processor will negate \overline{BG} and begin driving the bus again if external arbitration logic negates \overline{BR} before asserting \overline{BGACK} .
10. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, \overline{BG} may be reasserted.
11. The falling edge of S6 triggers both the negation of the strobes (\overline{AS} and \overline{DS}) and the falling edge of E. Either of these events can occur first, depending upon the loading on each signal. Specification #49 indicates the absolute maximum skew that will occur between the rising edge of the strobes and the falling edge of the E clock.



NOTES:

1. Setup time for the synchronous inputs \overline{BGACK} , $\overline{IPLO-2}$, and $\overline{VP\bar{A}}$ guarantees their recognition at the next falling edge of the clock.
2. \overline{BR} need fall at this time only in order to insure being recognized at the end of this bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage 2.0 volts, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 volt and 2.0 volts.

Figure G6-1. Read Cycle Timing Diagram (68000)



NOTES:

1. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 volt and 2.0 volts.
2. Because of loading variations, R/\overline{W} may be valid after \overline{AS} even though both are initiated by the rising edge of S2 (Specification 20A).

Figure G6-2. Write Cycle Timing Diagram (68000)

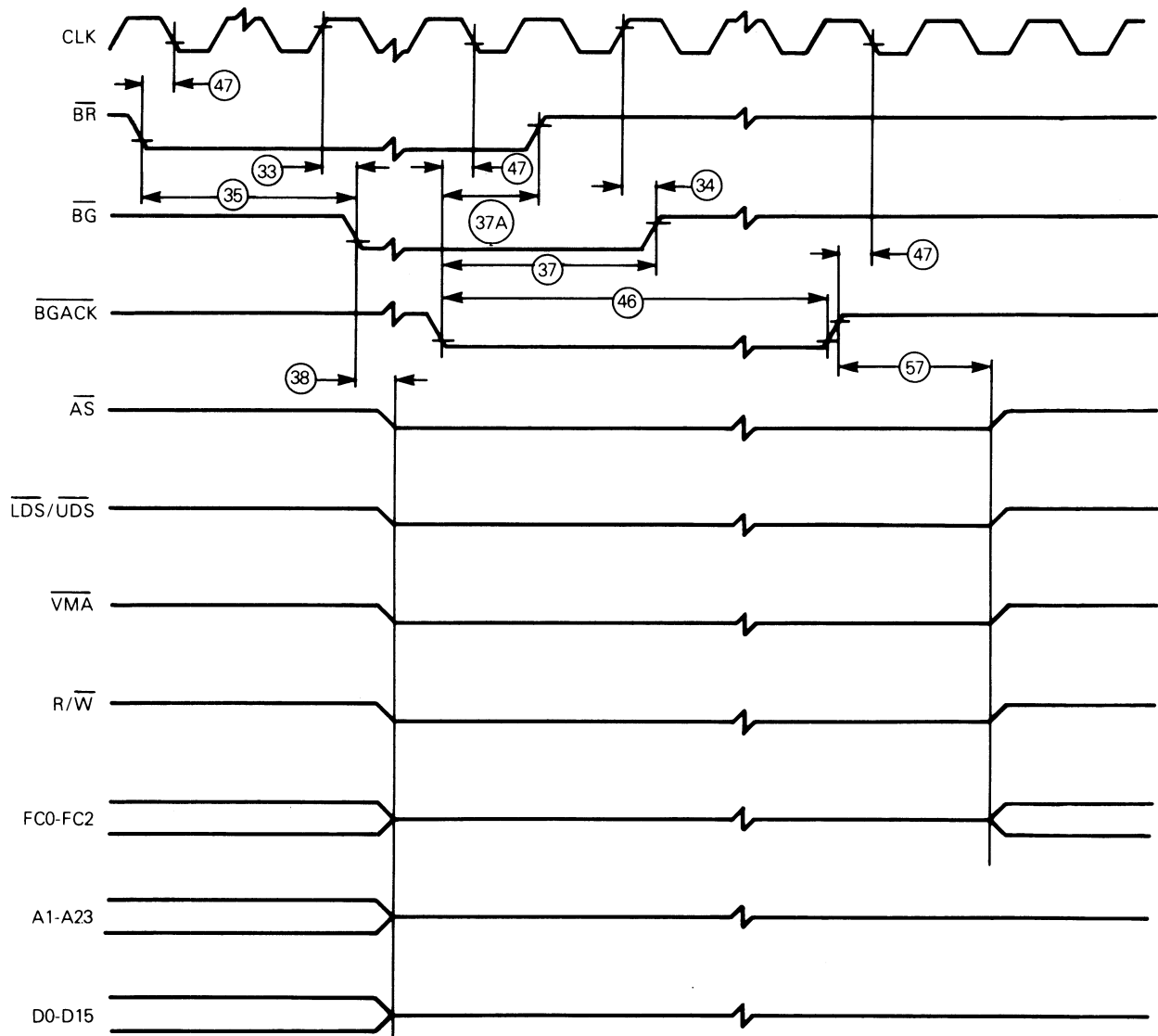


Figure G6-3. Bus Arbitration Timing Diagram - Idle Bus Case (68000)

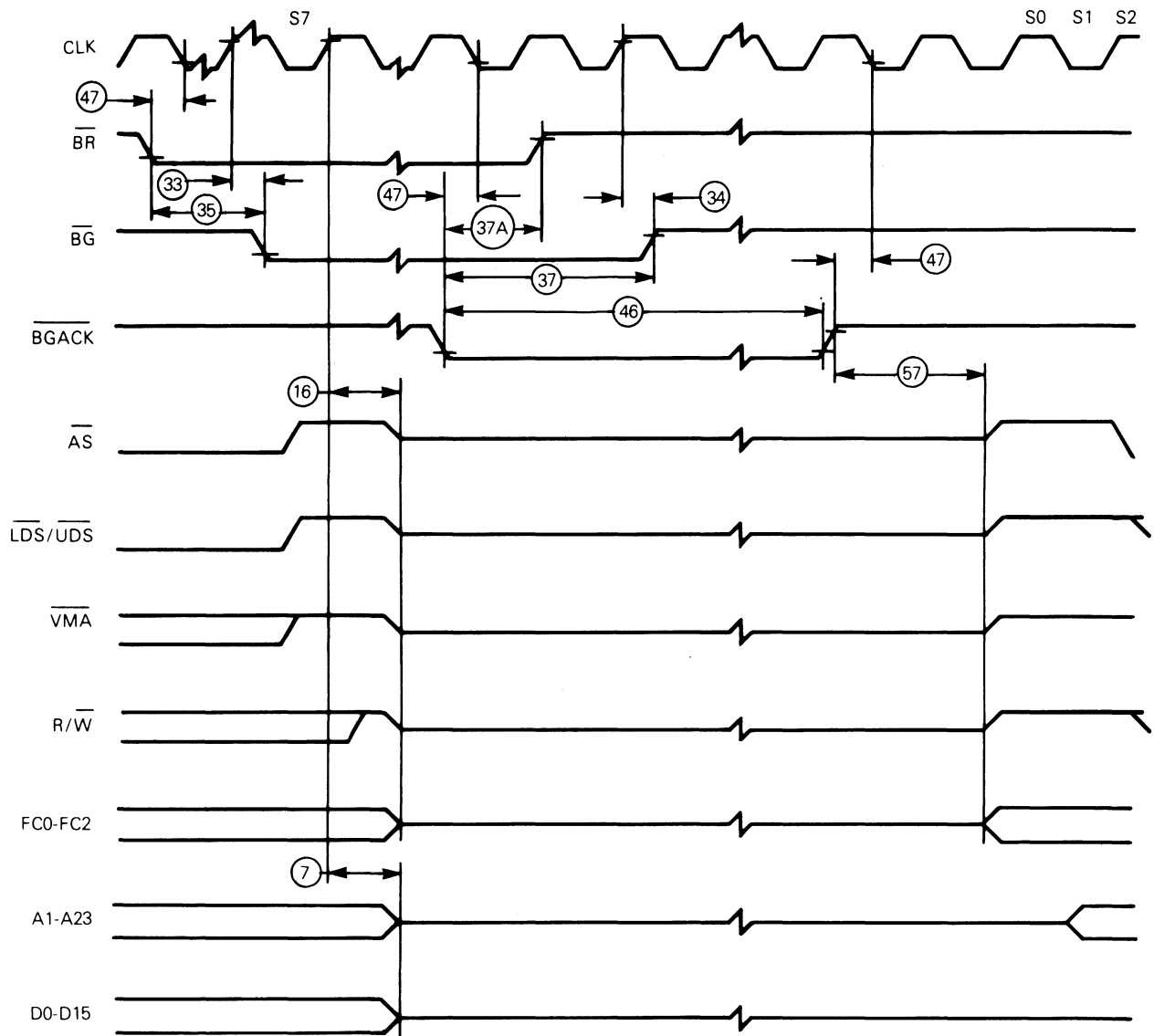


Figure G6-4. Bus Arbitration Timing Diagram - Active Bus Case (68000)

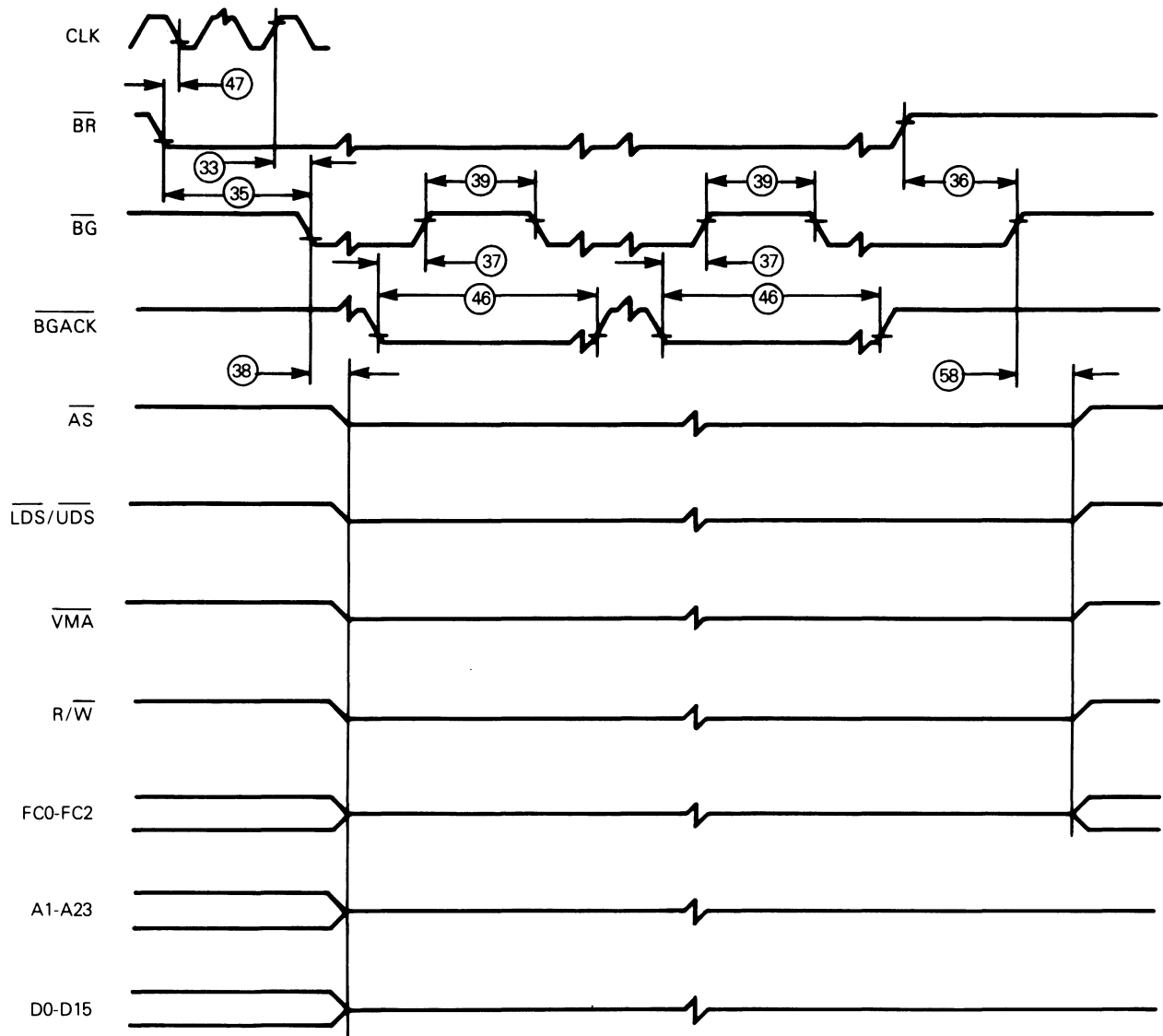


Figure G6-5. Bus Arbitration Timing Diagram - Multiple Bus Requests (68000)

Table G6-4. Electrical Specifications - Read and Write Cycles (68008)

(Vcc=5.0 Vdc +/- 5%; GND=0 Vdc; T_A=T_L to T_H; see Figures G6-6 and G6-7)

Num.	Characteristic	Symbol	8 MHz		10 MHz		Unit
			Min	Max	Min	Max	
1	Clock Period	t _{cyc}	125	500	100	500	ns
2	Clock Width Low	t _{CL}	55	250	45	250	ns
3	Clock Width High	t _{CH}	55	250	45	250	ns
4	Clock Fall Time	t _{Cf}	–	10	–	10	ns
5	Clock Rise Time	t _{Cr}	–	10	–	10	ns
6	Clock Low to Address Valid	t _{CLAV}	–	70	–	60	ns
6A	Clock High to FC Valid	t _{CHFCV}	–	70	–	60	ns
7	Clock High to Address, Data Bus High Impedance (Maximum)	t _{CHADZ}	–	80	–	70	ns
8	Clock High to Address, FC Invalid (Minimum)	t _{CHAFI}	0	–	0	–	ns
9 ¹	Clock High to \overline{AS} , \overline{DS} Low	t _{CHSL}	0	60	0	55	ns
11 ²	Address Valid to \overline{AS} , \overline{DS} Low (Read)/ \overline{AS} Low (Write)	t _{AVSL}	30	–	20	–	ns
11A ^{2,7}	FC Valid to \overline{AS} , \overline{DS} Low (Read)/ \overline{AS} Low (Write)	t _{FCVSL}	60	–	50	–	ns
12 ¹	Clock Low to \overline{AS} , \overline{DS} High	t _{CLSH}	–	35	–	35	ns
13 ²	\overline{AS} , \overline{DS} High to Address/FC Invalid	t _{SHARI}	30	–	20	–	ns
14 ^{2,5}	\overline{AS} , \overline{DS} Width Low (Read)/ \overline{AS} Low (Write)	t _{SL}	270	–	195	–	ns
14A ²	\overline{DS} Width Low (Write)	t _{DSL}	140	–	95	–	ns
15 ²	\overline{AS} , \overline{DS} Width High	t _{SH}	150	–	105	–	ns
16	Clock High to Control Bus High Impedance	t _{CHCZ}	–	80	–	70	ns
17 ²	\overline{AS} , \overline{DS} High to R/ \overline{W} High (Read)	t _{SHRH}	40	–	20	–	ns
18 ¹	Clock High to R/ \overline{W} High	t _{CHRH}	0	40	0	40	ns
20 ¹	Clock High to R/ \overline{W} Low	t _{CHRL}	–	40	–	40	ns
20A ⁶	\overline{AS} Low to R/ \overline{W} Valid (Write)	t _{ASRV}	–	20	–	20	ns
21 ²	Address Valid to R/ \overline{W} Low (Write)	t _{AVRL}	20	–	0	–	ns
21A ^{2,7}	FC Valid to R/ \overline{W} Low (Write)	t _{FCVRL}	60	–	50	–	ns
22 ²	R/ \overline{W} Low to \overline{DS} Low (Write)	t _{RLSL}	80	–	50	–	ns
23	Clock Low to Data Out Valid (Write)	t _{CLDO}	–	70	–	55	ns
25 ²	\overline{AS} , \overline{DS} High to Data Out Invalid (Write)	t _{SHDOI}	50	–	20	–	ns
26 ²	Data Out Valid to \overline{DS} Low (Write)	t _{DOSL}	35	–	20	–	ns
27 ⁵	Data In to Clock Low (Setup Time on Read)	t _{DICL}	15	–	10	–	ns
28 ^{2,5}	\overline{AS} , \overline{DS} High to \overline{DTACK} High	t _{SHDAH}	0	245	0	190	ns
29	\overline{AS} , \overline{DS} High to Data In Invalid (Hold Time on Read)	t _{SHDII}	0	–	0	–	ns
30	\overline{AS} , \overline{DS} High to \overline{BERR} High	t _{SHBEH}	0	–	0	–	ns
31 ^{2,5}	\overline{DTACK} Low to Data Valid (Asynchronous Setup Time on Read)	t _{DALDI}	–	90	–	65	ns
32	HALT and \overline{RESET} Input Transition Time	t _{RHr, f}	0	200	0	200	ns
33	Clock High to \overline{BG} Low	t _{CHGL}	–	40	–	40	ns
34	Clock High to \overline{BG} High	t _{CHGH}	–	40	–	40	ns
35	\overline{BR} Low to \overline{BG} Low	t _{BRLGL}	1.5	90 ns + 3.5	1.5	80 ns + 3.5	Clk.Per.
36 ⁸	\overline{BR} High to \overline{BG} High	t _{BRHGH}	1.5	90 ns + 3.5	1.5	80 ns + 3.5	Clk.Per.
37	\overline{BGACK} Low to \overline{BG} High (52-Pin Version Only)	t _{GALGH}	1.5	90 ns + 3.5	1.5	80 ns + 3.5	Clk.Per.
37A ⁹	\overline{BGACK} Low to \overline{BR} High (52-Pin Version Only)	t _{GALBRH}	20	1.5 Clocks	20	1.5 Clocks	ns
38	\overline{BG} Low to Control, Address, Data Bus High Impedance (\overline{AS} High)	t _{GLZ}	–	80	–	70	ns
39	\overline{BG} Width High	t _{GH}	1.5	–	1.5	–	Clk.Per.
41	Clock Low to E Transition	t _{CLET}	–	50	–	50	ns
42	E Output Rise and Fall Time	t _{Er, f}	–	15	–	15	ns
44	\overline{AS} , \overline{DS} High to \overline{VPA} High	t _{SHVPH}	0	120	0	90	ns

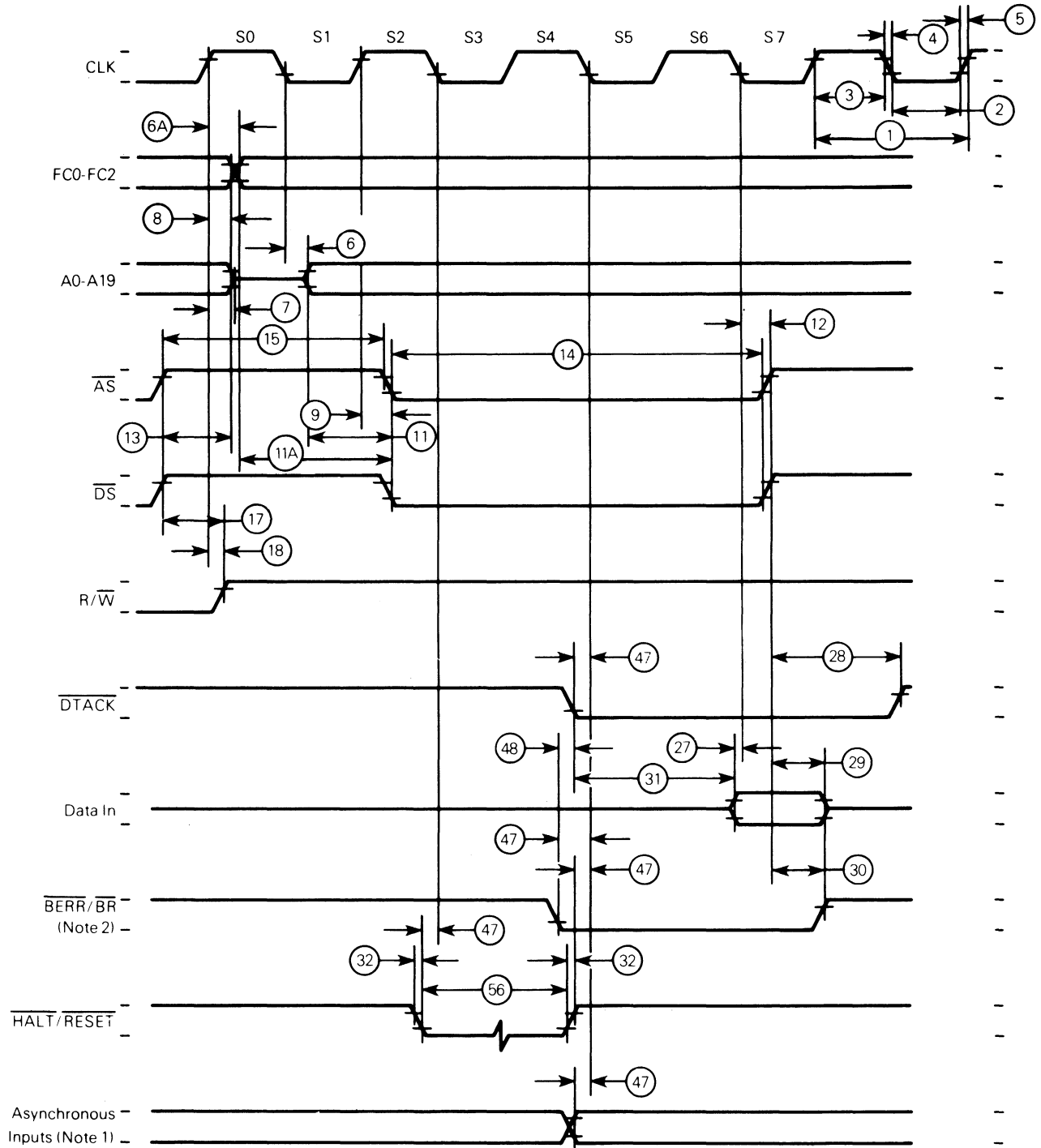
– Continued

Table G6-4. Electrical Specifications - Read and Write Cycles (68008) (Cont'd)

Num.	Characteristic	Symbol	8 MHz		10 MHz		Unit
			Min	Max	Min	Max	
45	E Low to Control, Address Bus Invalid (Address Hold Time)	t _{ELCAI}	30	—	10	—	ns
46	$\overline{\text{BGACK}}$ Width Low (52-Pin Version Only)	t _{GAL}	1.5	—	1.5	—	Clk.Per.
47 ⁵	Asynchronous Input Setup Time	t _{ASI}	10	—	10	—	ns
48 ³	$\overline{\text{BERR}}$ Low to $\overline{\text{DTACK}}$ Low	t _{BELDAL}	20	—	20	—	ns
49 ¹⁰	$\overline{\text{AS}}$, $\overline{\text{DS}}$ High to E Low	t _{SHEL}	—80	80	—80	80	ns
50	E Width High	t _{EH}	450	—	350	—	ns
51	E Width Low	t _{EL}	700	—	550	—	ns
53	Clock High to Data Out Invalid	t _{CHDOI}	0	—	0	—	ns
54	E Low to Data Out Invalid	t _{ELDOI}	30	—	20	—	ns
55	R/ $\overline{\text{W}}$ to Data Bus Impedance Driven	t _{RLDBD}	30	—	20	—	ns
56 ⁴	$\overline{\text{HALT}}$ / $\overline{\text{RESET}}$ Pulse Width	t _{HRPW}	10	—	10	—	Clk.Per.
57	$\overline{\text{BGACK}}$ High to Control Bus Driven (52-Pin Version Only)	t _{GABD}	1.5	—	1.5	—	Clk.Per.
58 ⁸	$\overline{\text{BG}}$ High to Control Bus Driven	t _{GHBD}	1.5	—	1.5	—	Clk.Per.

NOTES:

1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the values given in these columns.
2. Actual value depends on clock period.
3. If #47 is satisfied for both $\overline{\text{DTACK}}$ and $\overline{\text{BERR}}$, #48 may be 0 nanoseconds.
4. For power up the MPU must be held in $\overline{\text{RESET}}$ state for 100 milliseconds to allow stabilization of on-chip circuitry. After the system is powered up, #56 refers to the minimum pulse width required to reset the system.
5. If the asynchronous setup time (#47) requirements are satisfied, the $\overline{\text{DTACK}}$ low-to-data setup time (#31) requirement can be ignored. The data must only satisfy the data-in to clock-low setup time (#27) for the following cycle.
6. When $\overline{\text{AS}}$ and R/ $\overline{\text{W}}$ are equally loaded ($\pm 20\%$), subtract 10 nanoseconds from the values in these columns.
7. Setup time to guarantee recognition on next falling edge of clock.
8. The processor will negate $\overline{\text{BG}}$ and begin driving the bus again if external arbitration logic negates $\overline{\text{BR}}$ before asserting $\overline{\text{BGACK}}$.
9. The minimum value must be met to guarantee proper operation. If the maximum value is exceeded, $\overline{\text{BG}}$ may be reasserted.
10. The falling edge of S6 triggers both the negation of the strobes ($\overline{\text{AS}}$ and $\overline{\text{DS}}$) and the falling edge of E. Either of these events can occur first, depending upon the loading on each signal. Specification #49 indicates the absolute maximum skew that will occur between the rising edge of the strobes and the falling edge of the E clock.

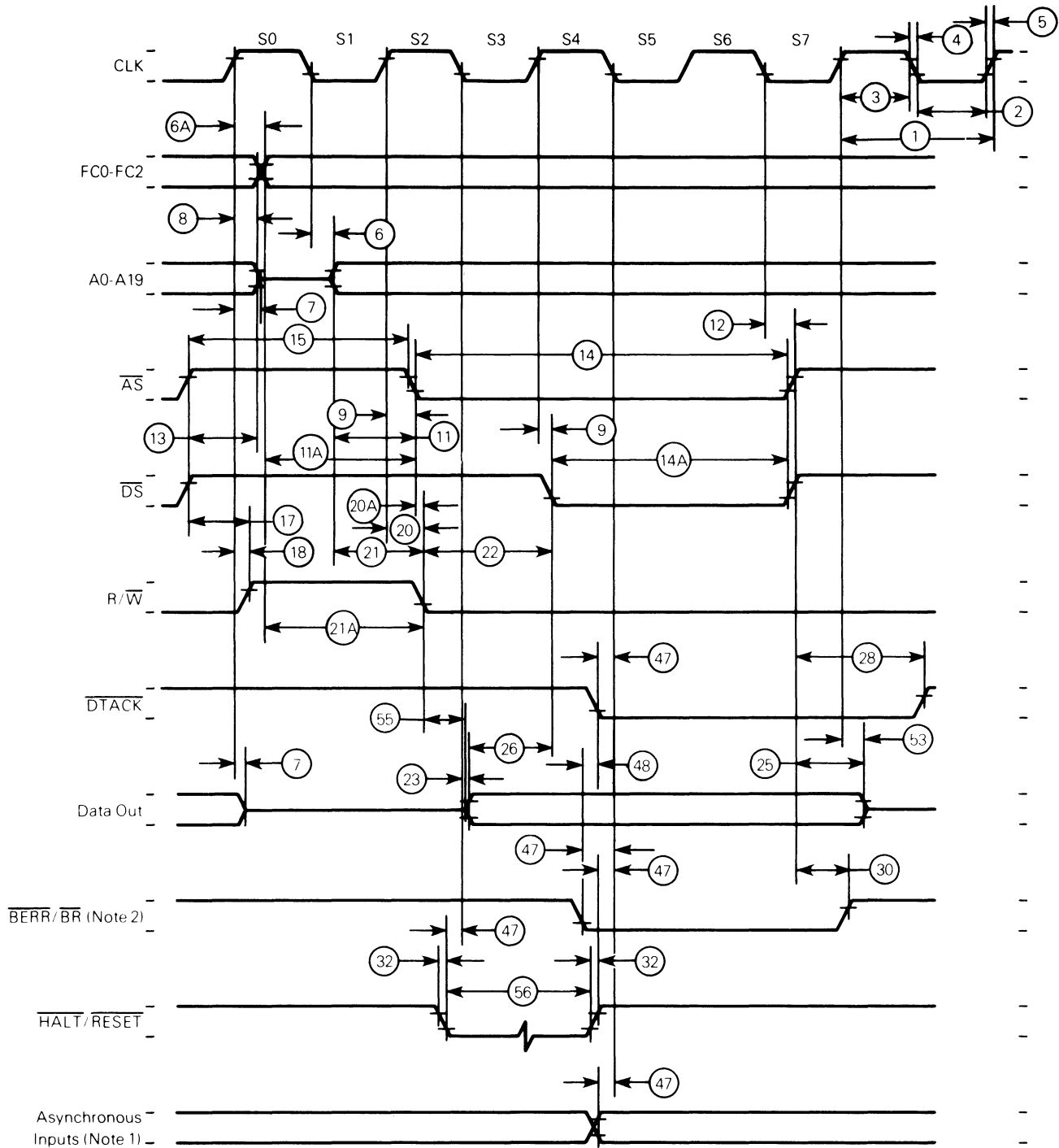


NOTES:

1. Setup time for the asynchronous inputs $\overline{IPLO/2}$, $\overline{IPL1}$, and \overline{VPA} guarantees their recognition at the next falling edge of the clock.
2. \overline{BR} need fall at this time only in order to insure being recognized at the end of this bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.

1-280

Figure G6-6. Read Cycle Timing Diagram (68008)

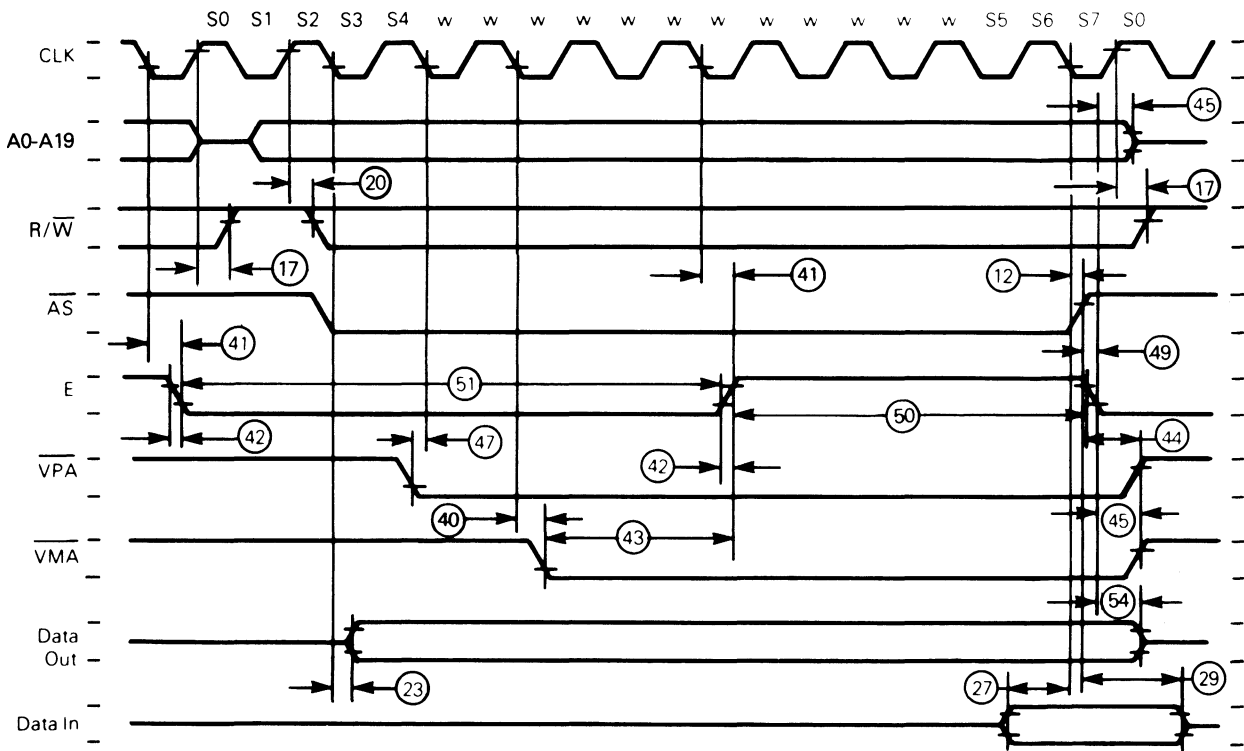


NOTES:

1. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.
2. Because of loading variations, R/W may be valid after AS even though both are initiated by the rising edge of S2 (Specification 20A)

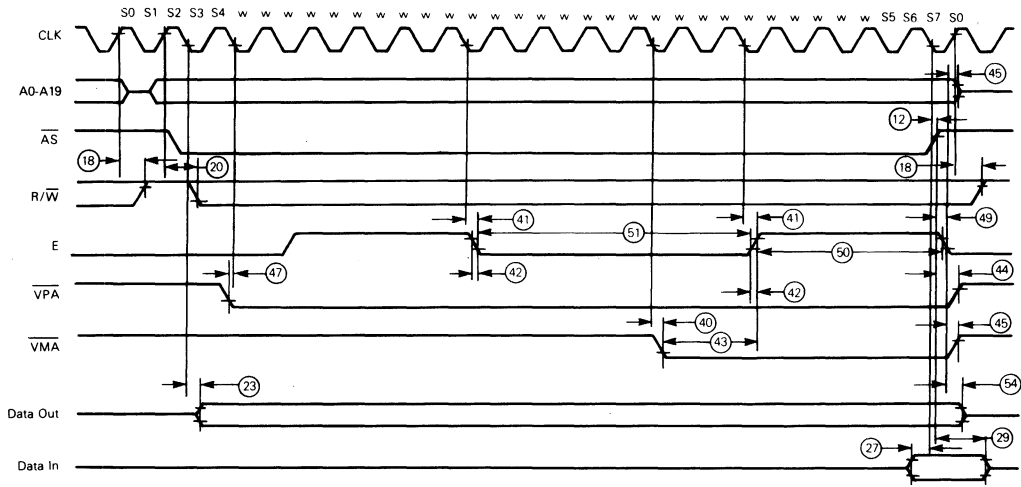
1-281

Figure G6-7. Write Cycle Timing Diagram (68008)



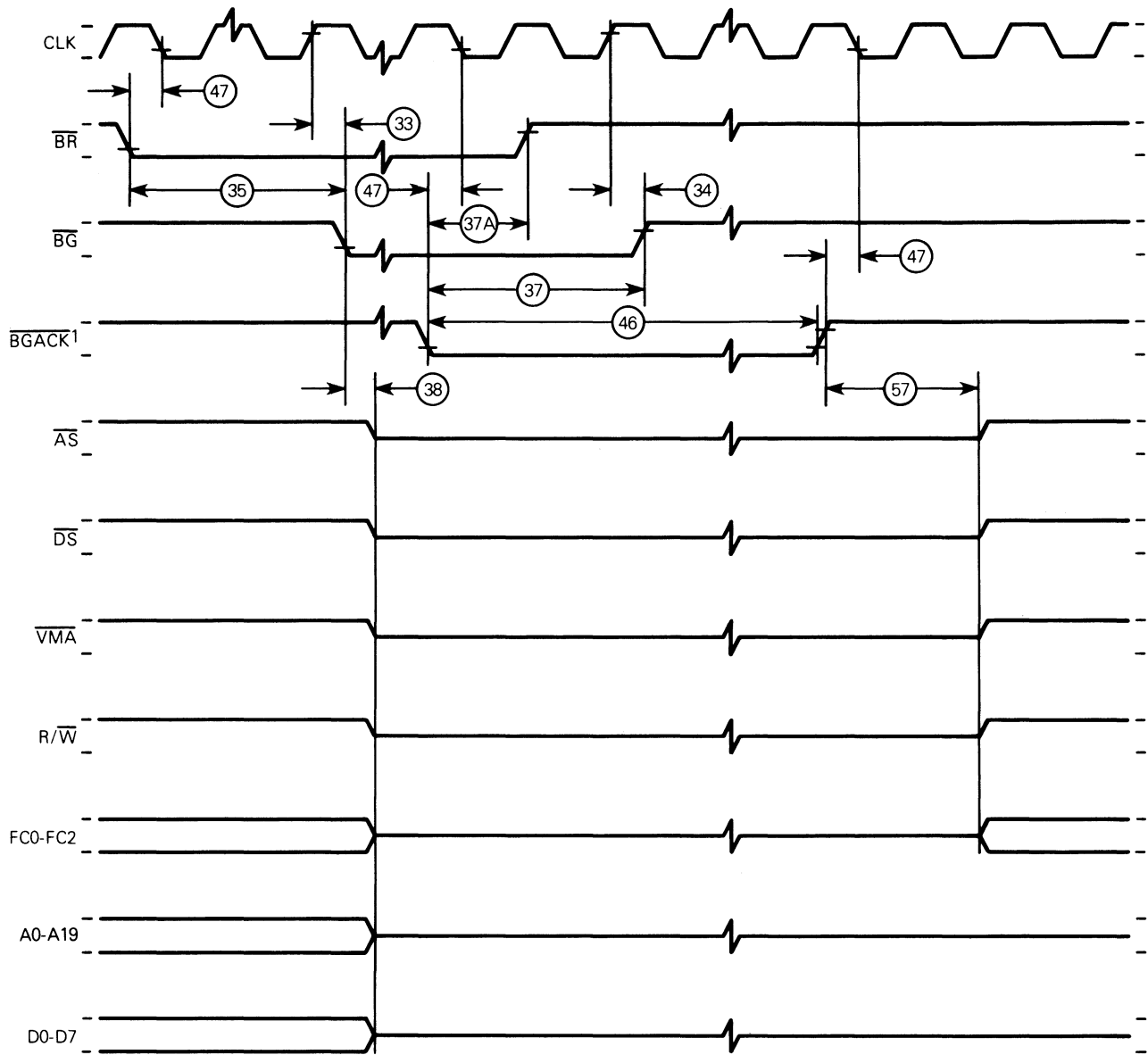
NOTE: This timing diagram is included for those who wish to design their own circuit to generate VMA. It shows the best case possibly attainable.

Figure G6-8. MC68008 to M6800 Peripheral Timing Diagram - Best Case (68008)



NOTE: This timing diagram is included for those who wish to design their own circuit to generate VMA. It shows the worst case possibly attainable.

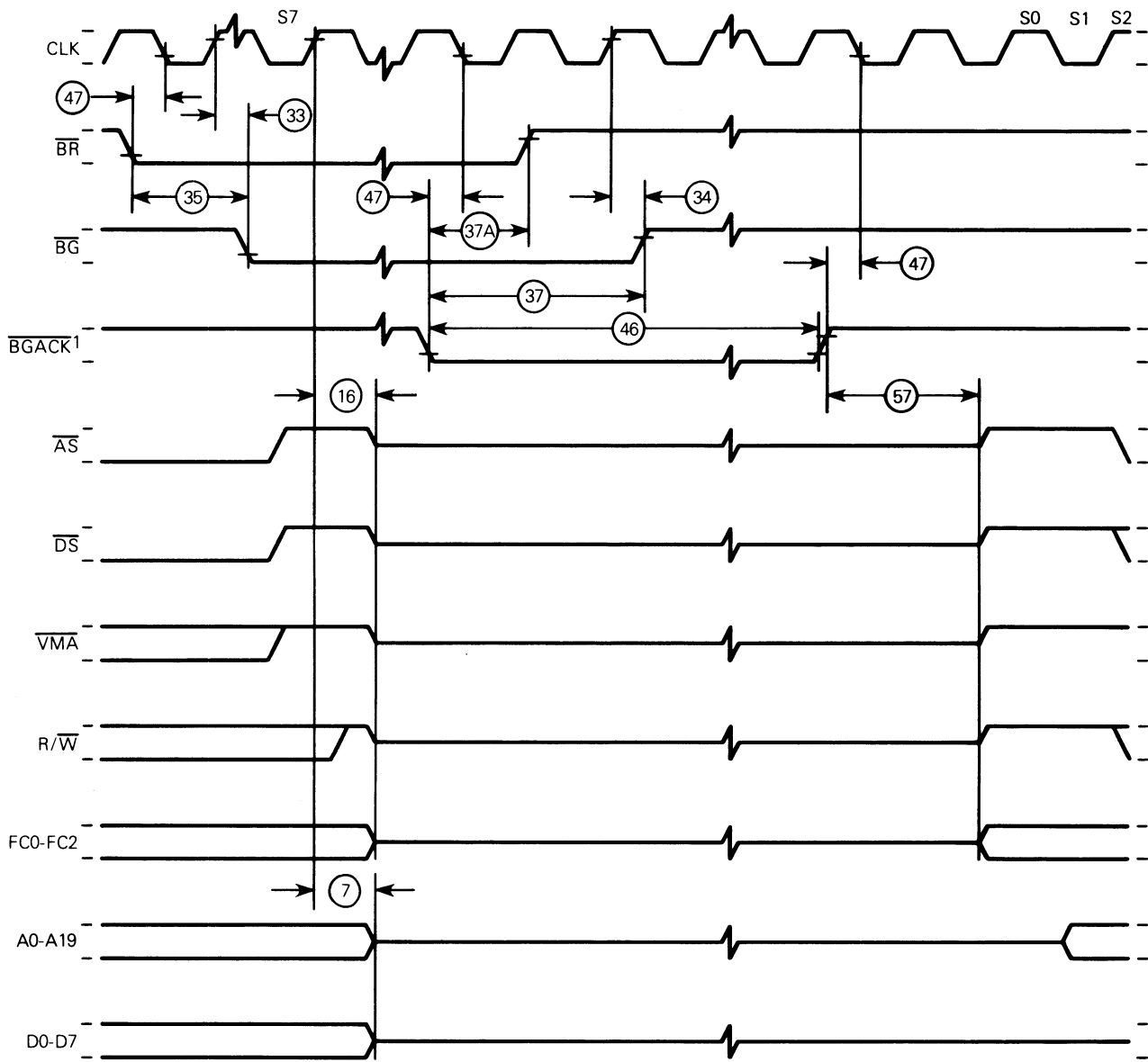
Figure G6-9. MC68008 to M6800 Peripheral Timing Diagram - Worst Case (68008)



NOTE:
 1. 52-Pin Version of MC68008 Only.

1-282

Figure G6-10. Bus Arbitration Timing Diagram - Idle Bus Case (68008)

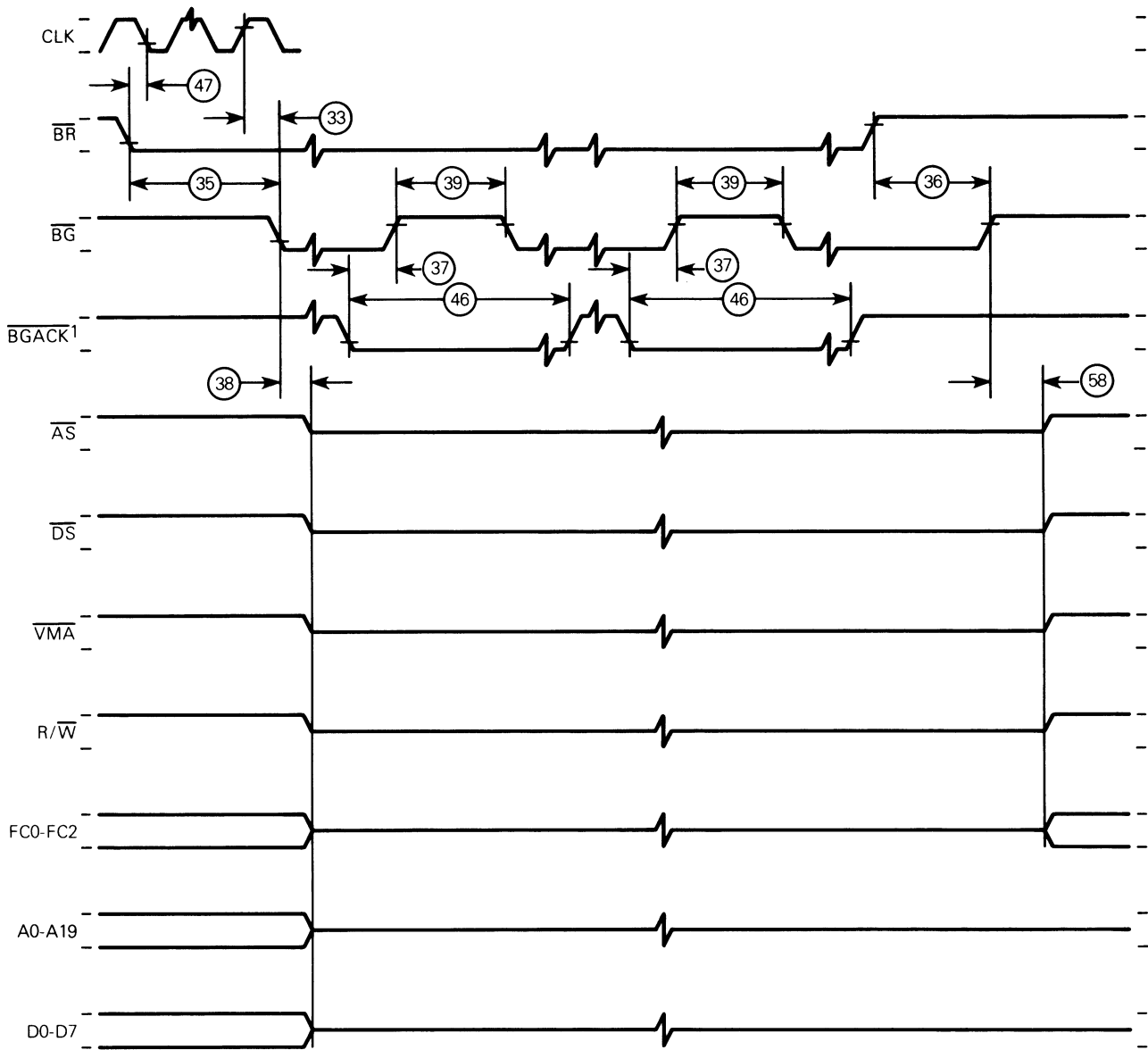


NOTE:

1. 52-Pin Version of MC68008 Only.

1-283

Figure G6-11. Bus Arbitration Timing Diagram - Active Bus Case (68008)



NOTE:
 1. 52-Pin Version of MC68008 Only.

1-284

Figure G6-12. Bus Arbitration Timing Diagram - Multiple Bus Requests (68008; 52-Pin Version Only)

NOTES

Appendix G

SERVICE INFORMATION

Section G7

ELECTROMAGNETIC COMPATIBILITY

With an emulation system installed in the Model 64000, several methods of operation (physical setup) may result in an increased electromagnetic (EM) emissions. To reduce EM emissions, any of the following techniques may be used:

- a. When the emulator is used infrequently, disconnect the emulator pod and cables from both the host system and target system.
- b. For systems that use the emulator intermittently, select "external clock" and disconnect the pod cable from the target system when not in use.
- c. Consistent with design needs, minimize the time that the emulator is used without being connected to a target system.
- d. All 64000 system covers should be in place and properly attached to the Development Station (all housing screws tight).
- e. Emulator performance verification is a service tool. Minimize its usage consistent with performance assurance.
- f. To further reduce the possibility of EMI (electromagnetic interference), ensure that your target system is electromagnetically compatible with other equipment in your area whenever it is connected to the emulator.

NOTES

Appendix G

INSTALLATION AND SERVICE INFORMATION

Section G8

SCHEMATIC DIAGRAMS

INTRODUCTION

This section contains schematics and detailed theory of operation for Models HP 64243 (68000) and HP 64242 (68008) Emulator Pod Circuit Boards.

THEORY OF OPERATION

The following paragraphs provide detailed theory of operation for both 6G8000 and 6G800G8 Emulator Pod Processor Boards and for the common Timing Board. Schematics are shown in figures G8-1, G8-2 and G8-3. Mnemonics (signal names) are described in Table G8-1.

NOTE

Whenever a signal name is preceded by an L in the following discussion, it indicates that the signal is asserted in the low state.

68000 PROCESSOR BOARD A1

U3G is an MC68000R12, 12.5 MHz microprocessor. It forms the core around which the emulator is built.

U5M, U5N, and U5O latch the 24-bit address bus to the emulation control board. The control board contains termination resistors to suppress transmission line problems.

U2A, U3A, and U4A are the user address trapeivers. These tristate buffers drive the address bus from the emulation processor onto the target system address bus. This will occur when LTSEN and HDMA (from the pod DMA circuitry) are both low. The target system will be allowed to drive the address bus into the emulator when HDMA is high, which will occur during a user bus request/grant/acknowledge sequence. These buffers will go to the high impedance state in both directions if LTSEN is false (high).

U2C and U3C are the user data trapeivers. These tristate buffers drive the data bus from the emulation processor onto the target system data bus. This will occur when LPDBEN and PDBDIR

(from the timing board) are both low, typically during an emulation processor write cycle or DMA read cycle. The target system will be allowed to drive the data bus into the emulator when PDBDIR is high, which will occur during an emulation processor read cycle. These buffers will go to the high impedance state in both directions if LPDEN is false (high).

U5A is a transceiver used to buffer the function codes (FC0-FC2) to the target system. These lines carry status information from the emulation processor. During a DMA, HDMA reverses the direction of these transceivers so that the target system may provide the status information for possible capture by the analysis unit.

U4C is the memory control transceiver used to drive the memory control signals needed for processor read and write cycles. The signals are LAS, LUDS, LLDS, R/LW, and LVMA. The signals are driven to the target system during normal operation (HDMA low). The target system will be allowed to drive the control signals into the emulator when HDMA is high. These buffers will go to the high impedance state in both directions if LTSEN is high. The buffers will also go into the high impedance state during an interrupt acknowledge cycle, which is caused by an emulation system break, if the user answers "yes" to the "Enable Interrupt level 7 sharing?" configuration question.

U10 is used to invert the BLVPA and BLBERR signals for application to the VPA/BERR enable/disable circuitry.

U1G, U1F and U1H are used to selectively enable or disable the BLVPA and BLBERR signals from reaching the emulation processor LVPA and LBERR inputs. Basically, these lines will be enabled to reach the emulation processor during any memory cycle access to address ranges mapped as user memory (LUSER), or during an interrupt acknowledge cycle (LINTACK). LBERR can also be enabled even if these two conditions are false by setting the LBERREN bit in the pod control register. The lines will be disabled if the emulator is not plugged into a powered-on target system or if the current interrupt acknowledge cycle is the result of an emulation system break (LBRKACK).

U1N, R1, and C1 are used to detect whether the emulator is plugged into a powered-on target system. When this is true, the HICE signal output from U1N will also be true. If the target system is not powered on or if the emulator is not plugged in, R1 pulls the input to U1N low, negating HICE. This line is fused by picofuse F1.

The LRESET and LHALT lines to/from the processor are normally bidirectional lines when a 68000 is plugged into a system. The emulator must assert the RESET condition thru U2D (pins 9 and 11) in order to set the emulator to a known state. Therefore, the user may select two configurations for these signals: (1) The line may be switched unbuffered to the processor. R4 or R5 will pull these lines high in case the emulator is out-of-circuit and CR1/CR2 or CR3/CR4 will protect the lines. The user will, however, be exposed to activity on the part of the emulation system. (2) The line may be switched to U1D. U1D and U2D buffer and gate off the line when out-of-circuit. The user is thus shielded from any interference by the emulator. However, the processor will no longer be able to drive the line to the user.

U2G and U2F buffer and gate the interrupt lines to the processor. U2F is used to disable all interrupts from the target system. When LINTENB is high, the outputs of U2F will always be high, thus forcing emulation processor lines LIPL0-LIPL2 high. The user interrupts are disabled when the emulator is out-of-circuit or when selected by a bit in the Pod Control Register. The common inputs to U2G allow a level 7 interrupt from the emulation system to break the processor. When LINT7 is true, LIPL0-LIPL2 will all be asserted. U2G also allows various levels of interrupts from the target system to drive the emulation processor priority interrupt lines.

U4E, U4G, U3D and U5E flip-flops are used to form memory control signals to the emulation control board. U4G (qualified by AS and clocked by AS delayed 30 ns) latches the LLDS and LUDS control lines. One half of U4E latches the R/W line in the same manner. The data strobes are OR'ed by U1I (LDS). The falling edge of LDS is synchronized to the processor clock (HCLK) by one half of flip-flop U4E. U1J, U5I, U3D and U5B provide a latch signal for the address and status (LLATCH). Flip-flop U3D delays this latch signal one clock on read cycles and provides no delay on write cycles. One half of flip-flop U5E synchronizes the rising edge of LDS with the next falling edge of HCLK. The next rising edge of HCLK sets the other half of flip-flop U5E. This causes flip-flops U4G, U4E, U3D and U5E to be reset (assuming a jumper is position E2 - NORMAL mode). Jumper E1,E2 can be positioned in the NORMAL (E2) or TEST (E1) mode. In the NORMAL position, the normal emulator operation occurs. In the TEST position, the feedback loop on the memory control signals can be broken for troubleshooting. These flip-flops can also be cleared in one of three ways: 1) A DEFIB condition (assertion of LDEFIB through U5D, U1I and U5D); 2) A DMA Tagging condition (assertion of HHMAVEN through U5D, U1I and U5D); or 3) In response to the situation where an address strobe (AS) is asserted without a data strobe (DS). This condition occurs during address error cycles on the 68010 processor and some mask sets of the 68000. This reset is applied through U4H (Latched data strobe), U1M, U1M, U1I and U5D.

U2P buffers the READ/WRITE flip-flop U4E and sends it to the emulation control board as LEWRT. It also buffers the LLDS and the LUDS flip-flop (U4G) and sends them as LBYTE and LBUP. LBUP is also sent as HEA0.

The analysis strobe LANALIN is also buffered by U2P and is generated by U5E as routed through U4J. This strobe can also be generated by U5H (HHMAVEN and LALTDS) through U4J for DMA Tagging cycles.

U1K holds off the generation of EHMAV and LWDV for certain function code memory mappings. This only applies to the 68010 processor. In the 68000 version of the pod the "user hold-off" line (LUSERHO) is held high (this line is generated by the emulation pod control register located on the timing board). Therefore, the only path allowable through U1K is by way of pins 4, 5, and 6. Processor reads and writes develop EHMAV by way of a high on U5E pin 6 and a high on U4H pin 6 (Latched Data Strobe). This signal passes through U4J and is buffered by U2P. EHMAV can be held off in two ways: 1) During Interrupt Acknowledge Cycles (assertion of INTACK through U1J and U4H to U4J), and 2) During DMA Tagging (assertion of LALTSTB through U4H to U4J).

LWDV is generated through U4J and 3X50 Delay line U4I (with U4F and U4H) and is buffered by U2P. This signal is qualified by Writes (READ not asserted) and a EHMAV strobe by U4J.

U3P and U5P are the user mappers that provide directional information for the various data busses on the pod. Multiplexer U4P selects one of two modes of mapping. When LFCSEL (Low Function Code Select) is at a logic high, HEA 9-11 drive addresses A0-2 of user mappers U3P and U5P. This means that the LUSER line (U5I) is mapped as a function of HEA9-23 and that the function codes are not a part of this mapping. When LFCSEL is at a logic low, LESTAT4-6 (Low Emulation Status 4-6), an inverted and latched version of FC0-2, drive addresses A0-2 of the user mappers. This configuration maps the LUSER line as a function of FC0-2 and HEA12-23. The mapper rams are loaded via the control board: The HEA lines are driven up to the address lines of the mappers and the D0 line drives the Data In (DI) line of the rams. Target addresses are programmed with a logic high and emulation addresses are programmed with a logic low.

The control board ID is set to 68000 by placing a jumper in the "E4" position (ID = 0035H).

Processor clock generation is controlled by relay K1. When pod control bit HICLK (High Internal Clock) is programmed to a logic high, the internal 10MHz Oscillator drives the CLK line to the

processor. A low on HICLK routes the external (target) clock to the processor (12.5 MHz maximum clock rate). Diodes CR5 and CR6 provide clamping protection to the processor.

The six 4-bit counters 2L, 3L, 4M, 4L, 4O and 4N increment on each program fetch and the output of these counters are compared to the next processor address by comparators 3O, 3N and 3M to generate signals LEQ1 (Low Equal 1), LEQ2 (Low Equal 2), and LEQ3 (Low Equal 3). These three signals are and'ed through U4J and U5H and latched by U5O to generate AE (Address Equal) and buffered by U2K to generate BAE (Buffered Address Equal). BAE is driven down to the depipeline circuitry on the control board. Flip-flops U2M and one half of U2N control the loading (LLOAD) and counting (CNT) of these counters. If the input "cycle" is a program cycle the address counters will be incremented by two (actually incremented by one, but the LSB is A1). If the input "cycle" is a program cycle and the address is not equal to the address expected, the address counter is loaded with the new value. The address counter is still incremented by two after the new value is loaded. The address counter is loaded on the falling edge of CLKE. The counter is incremented on the falling edge of CLKA.

U1L and U1F decode A1-3 and assert LLEV7 (Low Level 7).

U3K and U5K through U2E generate LINTACK (Low Interrupt Acknowledge). This condition occurs when A4-23 and FC0-2 are all high, as in a processor interrupt acknowledge cycle.

U2J and U2F decode LIPL0, LIPL1 and LIPL2 and assert LIPL7 (Low Interrupt Priority Level 7).

U4K latches the following signals: LINTACK (latched and inverted to INTACK), FC0-2 (latched and inverted to LESTAT4-6) and HDMA (latched and inverted to LESTAT7). These signals are sent to the control board for analysis status.

One half of flip-flop U2N latches LBERR (processor Bus Error) on the rising edge of LDS and sends it to the control board through U2P for the depipeline circuitry.

There are two PV latch networks. The first series of latches (U2B, U3B and U4B) input the BA1-23 Buffered Address lines whenever a processor write occurs with an Upper Data Strobe (LUDS). The contents of latches U3B and U4B are output to BD0-15 by reading an address with A1=0 while LPV1 (Low Performance Verification 1) is asserted. Similarly, if LVP2 (Low Performance Verification 2) is asserted, reading an address with A1=1 will output the contents of latch U2B to BD0-7 and the contents of latch U1A to BD8-10. PV latch U1A inputs BAE (Buffered Address Equal) and LUSERMEM (User Memory). The contents of these latches can be output by reading a target (user) memory address while out of circuit.

68008 PROCESSOR BOARD A1

U3C is an MC68008L10, 10.0 MHz microprocessor. It forms the core around which the emulator is built.

U5F, U4F, and U3F latch the 20-bit address bus to the emulation control board. The control board contains termination resistors to suppress transmission line problems.

U2B, U3B, and U4B are the user address trceivers. These tristate buffers drive the address bus from the emulation processor onto the target system address bus. This will occur when LTSEN and HDMA (from the pod DMA circuitry) are both low. The target system will be allowed to drive the address bus into the emulator when HDMA is high, which will occur during a user bus

request/grant/acknowledge sequence. These buffers will go to the high impedance state in both directions if LTSEN is false (high).

U5A is the user data transceiver. This tristate buffer drives the data bus from the emulation processor onto the target system data bus. This will occur when LPDBEN and PDBDIR (from the timing board) are both low, typically during an emulation processor write cycle or DMA read cycle. The target system will be allowed to drive the data bus into the emulator when PDBDIR is high, which will occur during an emulation processor read cycle. This buffer will go to the high impedance state in both directions if LPDEN is false (high).

U6B is a transceiver used to buffer the function codes (FC0-FC2) to the target system. These lines carry status information from the emulation processor. During a DMA, HDMA reverses the direction of these transceivers so that the target system may provide the status information for possible capture by the analysis unit.

U5D is the memory control transceiver used to drive the memory control signals needed for processor read and write cycles. The signals are LAS, LUDS, LLDS, R/LW, and LVMA. The signals are driven to the target system during normal operation (HDMA low). The target system will be allowed to drive the control signals into the emulator when HDMA is high. These buffers will go to the high impedance state in both directions if LTSEN is high. The buffers will also go into the high impedance state during an interrupt acknowledge cycle, which is caused by an emulation system break, if the user answers "yes" to the "Enable Interrupt level 7 sharing?" configuration question.

U2G is used to invert the BLVPA and BLBERR signals for application to the VPA/BERR enable/disable circuitry.

U2H, U2I, and U2M are used to selectively enable or disable the BLVPA and BLBERR signals from reaching the emulation processor LVPA and LBERR inputs. Basically, these lines will be enabled to reach the emulation processor during any memory cycle access to address ranges mapped as user memory (LUSER), or during an interrupt acknowledge cycle (LINTACK). LBERR can also be enabled even if these two conditions are false by setting the LBERREN bit in the pod control register. The lines will be disabled if the emulator is not plugged into a powered-on target system or if the current interrupt acknowledge cycle is the result of an emulation system break (LBRKACK).

U2K, R2, and C13 are used to detect whether the emulator is plugged into a powered-on target system. When this is true, the HICE signal output from U2K will also be true. If the target system is not powered on or if the emulator is not plugged in, R2 pulls the input to U2K low, negating HICE. This line is fused by picofuse F1.

The LRESET and LHALT lines to/from the processor are normally bidirectional lines when a 68008 is plugged into a system. The emulator must assert the RESET condition thru U11 (pins 9 and 11) in order to set the emulator to a known state. Therefore, the user may select two configurations for these signals: (1) The line may be switched unbuffered to the processor. RP3-6 or RP3-7 will pull these lines high in case the emulator is out-of-circuit and CR3/CR5 or CR4/CR6 will protect the lines. The user will, however, be exposed to activity on the part of the emulation system. (2) The line may be switched to U1L. U1L and U11 buffer and gate off the line when out-of-circuit. The user is thus shielded from any interference by the emulator. However, the processor will no longer be able to drive the line to the user.

U5E and U2F buffer and gate the interrupt lines to the processor. U2F is used to disable all interrupts from the target system. When LINTENB is high, the outputs of U2F will always be high, thus forcing emulation processor lines LIPL0-LIPL2 high. The user interrupts are disabled when

the emulator is out-of-circuit or when selected by a bit in the Pod Control Register. The common inputs to U5E allow a level 7 interrupt from the emulation system to break the processor. When LINT7 is true, LIPL0-LIPL2 will all be asserted. U5E also allows various levels of interrupts from the target system to drive the emulation processor priority interrupt lines.

U5L, U5K, U6P and U4M flip-flops are used to form memory control signals to the emulation control board. U5L (qualified by AS and clocked by AS delayed 30 ns) latches the LDS control line. One half of U5L latches the R/W line in the same manner. The data strobe is inverted by U5M. The falling edge of LDS is synchronized to the processor clock (HCLK) by one half of flip-flop U5K. U5O, U6P, and U5J provide a latch signal for the address and status (LLATCH). Flip-flop U6P delays this latch signal one clock on read cycles and provides no delay on write cycles. One half of flip-flop U4M synchronizes the rising edge of LDS with the next falling edge of HCLK. The next rising edge of HCLK sets the other half of flip-flop U4M. This causes flip-flops U5L, U5K, U6P and U4M to be reset (assuming jumper E1 is in the NORMAL position). Jumper E1 can be positioned in the NORMAL or TEST mode. In the NORMAL position, the normal emulator operation occurs. In the TEST position, the feedback loop on the memory control signals can be broken for troubleshooting. These flip-flops can also be cleared in one of three ways: 1) A DEFIB condition (assertion of LDEFIB through U5N, U2N, and U5N); 2) A DMA Tagging condition (assertion of HHMAVEN through U5N, U2N and U5N); or 3) In response to the situation where an address strobe (AS) is asserted without a data strobe (DS). This condition occurs during address error cycles on the 68008 processor. This reset is applied through U5L (Latched data strobe), U2I, U2N, and U5N.

U2D buffers the READ/WRITE flip-flop U5L and sends it to the emulation control board as LEWRT. HEA0 is buffered to form LBUP. LBYTE is grounded, since the 68008 is an 8-bit processor.

The analysis strobe LANALIN is also buffered by U2D and is generated by U4M as routed through U5O. This strobe can also be generated by U5I (HHMAVEN and LALTDS) through U5O for DMA Tagging cycles.

Processor reads and writes develop EHMAV by way of a high on U4M pin 6 and a high on U5L pin 6 (Latched Data Strobe). This signal passes through U4E and is buffered by U2D. EHMAV can be held off in two ways: 1) During Interrupt Acknowledge Cycles (assertion of INTACK through U2G and U2M to U4E), and 2) During DMA Tagging (assertion of LALTSTB through U2M to U4E).

LWDV is generated through U5O and 3X50 Delay line U6Q (with U5M and U5J) and is buffered by U2D. This signal is qualified by Writes (READ not asserted) and a EHMAV strobe by U5O.

U4J is the user mapper that provides directional information for the various data buses on the pod. The mapper ram is loaded via the control board: The HEA lines are driven up to the address lines of the mapper and the D0 line drives the Data In (DI) line of the ram. Target addresses are programmed with a logic low and emulation addresses are programmed with a logic high.

Processor clock generation is controlled by relay K1. When pod control bit HICLK (High Internal Clock) is programmed to a logic high, the internal 10MHz Oscillator drives the CLK line to the processor. A low on HICLK routes the external (target) clock to the processor (10 MHz maximum clock rate). Diodes CR1 and CR2 provide clamping protection to the processor.

The five 4-bit counters 6H, 4I, 5H, 3H, and 4H increment on each program fetch and the output of these counters are compared to the next processor address by comparators 5G, 4G, and 3G to generate signals LEQ1 (Low Equal 1), LEQ2 (Low Equal 2), and LEQ3 (Low Equal 3). These three signals are and'ed through U4E and U5I and latched by U3F to generate AE (Address Equal) and buffered by U3N to generate BAE (Buffered Address Equal). BAE is driven down to the depipeline circuitry on the control board. Flip-flops U4N and one half of U4L control the loading (LLOAD) and

counting (CNT) of these counters. If the input "cycle" is a program cycle the address counters will be incremented by two (actually incremented by one, but the LSB is A1). If the input "cycle" is a program cycle and the address is not equal to the address expected, the address counter is loaded with the new value. The address counter is still incremented by two after the new value is loaded. The address counter is loaded on the falling edge of CLKE. The counter is incremented on the falling edge of CLKA.

U5E and U2M decode A1-3 and assert LLEV7 (Low Level 7).

U2C and U2E through U4E generate LINTACK (Low Interrupt Acknowledge). This condition occurs when A4-19 and FC0-2 are all high, as in a processor interrupt acknowledge cycle.

U2F decodes LIPL0, LIPL1 and LIPL2 and asserts LIPL7 (Low Interrupt Priority Level 7).

U2J latches the following signals: LINTACK (latched and inverted to INTACK), FC0-2 (latched and inverted to LESTAT4-6) and HDMA (latched and inverted to LESTAT7). These signals are sent to the control board for analysis status.

One half of flip-flop U5K latches LBERR (processor Bus Error) on the rising edge of LDS and sends it to the control board through U3N for the depipeline circuitry.

There is one PV latch. Latches U4A, U3A, and U2A input the BA0-19 Buffered Address lines whenever a processor write occurs with a Data Strobe (LDS). This PV latch also inputs BAE (Buffered Address Equal) and LUSERMEM (User Memory). The contents of these latches can be output by reading a target (user) memory address while out of circuit. Reading an address with A0=0 and A1=0 while LVP0 (Low Performance Verification 0) is asserted enables the contents of latch U2A to be output to BD0-7. Similarly, reading an address with A0=1 and A1=0 will output the contents of latch U3A to BD0-7, and reading an address with A0=0 and A1=1 will output the contents of latch U4A to BD0-7.

TIMING BOARD A2

Data Bus Transmission and Control

The data transceivers (U3M, U3N, U4M and U4N) are used to buffer and control the flow of data between the emulation processor data bus and the pod data bus going to the emulation control board. The timing board is common to the 68000/10 16-bit processors as well as the 68008 8-bit processor. Therefore, it interfaces an 8- or 16-bit emulation processor bus to the 16-bit emulation bus.

- a) Direction of transceivers U3M and U3N is controlled by LDXMT and LEVENSEL. When LDXMT is asserted, data bits D8-D15 are sent from the emulation pod to the control board; when LEVENSEL is asserted, data bits D8-D15 are received by the emulator pod from the control board. If both signals are deasserted, the outputs of the buffers in each direction are set to the high impedance state.
- b) Direction of transceivers U4M and U4N is controlled by LDXMT and LODDSEL. When LDXMT is asserted, data bits D0-D7 are sent from the emulation pod to the control board; when LODDSEL is asserted, data bits D0-D7 are received by the emulator pod from the control board. If both signals are deasserted, the outputs of the buffers in each direction are set to the high impedance state.

RP2 and RP3 are termination resistor packs which, along with an identical set of packs on the emulation control board, are used to minimize transmission line problems.

U3J and U4J latch the data sent from the emulation pod to the control board. The latches are made transparent when either the DS or TRANS signals are asserted. Data Strobe (DS), a signal which is generated on the processor board, is asserted (high) when either the LUDS and/or the LLDS signals are low. TRANS (transparent) is an output of the Vector Register Decoder PAL (U2K). TRANS is equal to the logical equation: $\overline{HDEFIB} + \overline{HRDPOD}$. LDEFIB (Low Defibrillate) is generated on the control board and passes through inverter U1O on the processor board to generate the active high signal HDEFIB. This signal is an active high reset signal for the emulator pod. LRDPOD is generated on the control board, passes through buffer U1N on the processor board, and then passes through inverter U3E on the timing board to generate the active high signal HRDPOD. When HRDPOD is valid, the addressed register on the emulator pod will be read by the mainframe CPU.

As previously mentioned, LEVENSEL and LODDSEL, are data bus transceiver directional controls. When these signals are asserted, the data is directed from the control board to the pod. These signals are outputs of U1D. LEVENSEL is asserted when EVEN and CDBB are asserted. In a manner, LODDSEL is asserted when ODD and CDBB are asserted. CDBB (Control Data Bus B) is the output of control logic including gates U2F, U3F, U4E, U4C, and U4D. CDBB is asserted when HDEFIB, HUSER, and WRITE are deasserted indicating that an emulation memory read is in progress. CDBB is also asserted when LDEFIB and LWRTPOD are asserted (i.e. when a write is made to any of the pod control registers). CDBB is deasserted when LINTACK or LINT is asserted and HDEFIB is deasserted (i.e. during break vector address jams or interrupt acknowledge cycles).

As previously mentioned, LDXMT is a data bus transceiver directional control. When LDXMT is asserted, data is directed from the pod to the control board. The gates which generate this control signal are U1E, U2F, U4F and U1G. LDXMT is asserted whenever HDEFIB is deasserted, WRITE is asserted, and LUSER is deasserted (i.e. a write to emulation memory). LDXMT is also asserted while HDEFIB is deasserted and either LUSER, LINTACK, or LALTSTB is asserted (i.e. any access to user space, an interrupt acknowledge cycle in progress, or certain types of DMA cycles are taking place). Furthermore, during performance verification, LDXMT is asserted. This enables the control register values to be passed from the pod to the control board. This occurs by the assertion of LREAD and LSEL through U1G.

U5E controls the enabling of the processor board user data bus transceivers (U2C, U3C). The output of U5E is LPDBEN (Low Probe Data Bus Enable). Any one of three sequences of events will cause these buffers to be enabled: 1. Normal (non-DMA) cycles: LRESET or LINT is deasserted, LDMA is asserted, and HUSER is asserted (i.e. a user access cycle). 2. DMA cycle: LRESET is deasserted, HDMA is asserted, and HBLKDMA is deasserted (i.e. a DMA device cycle). 3. A Target System Interrupt Acknowledge cycle: INTACK (Interrupt Acknowledge) and HMEMCYC (High Memory Cycle) are asserted and LBRKACK (Low Break Acknowledge) is deasserted.

U5F controls the direction of the processor board user data bus transceivers (U2C, U3C). The output of U5E is PDBDIR (Probe Data Bus Direction). When PDBDIR is a logical low, the buffers are directed to write to the target system, while a logic high turns the buffers around to perform a read from the target system. For normal (non- DMA) cycles, LDMA is deasserted and the R/LW line from the processor determines the value of PDBDIR. DMA cycles use the other path through U5F and essentially reverse the direction of the buffer on reads and writes.

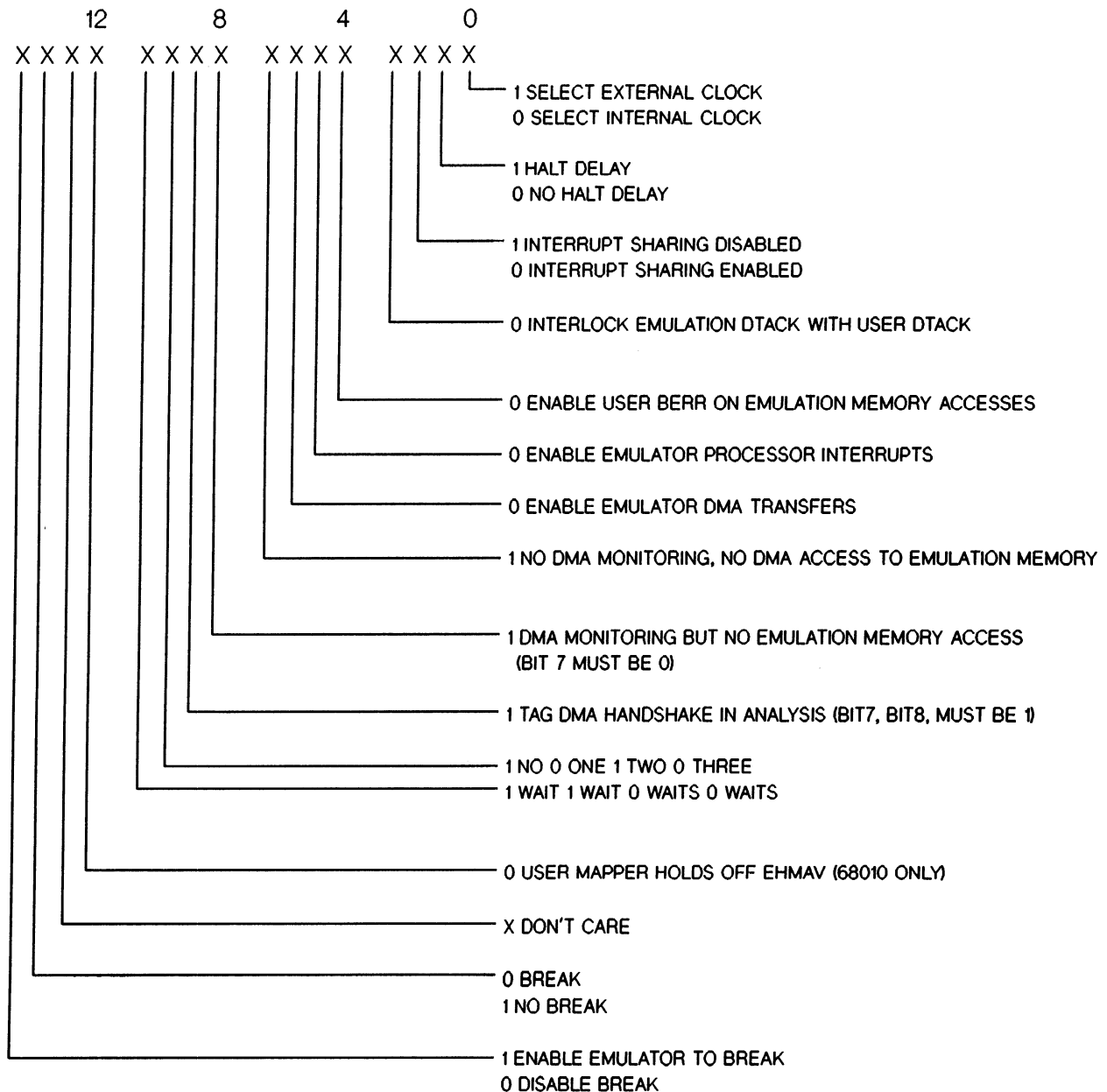
Pod Registers

POD INDEX REGISTER (U1L). This register is written to when LWRTPOD and HWPINDEX are both asserted. HWPINDEX (High Write Pod Index) and LWRTPOD (Low Write Pod) are outputs of the data direction logic on the control board. Writing to the Pod Index Register selects one of eight functions as follows:

<u>BIT</u>	<u>SIGNAL</u>	<u>FUNCTION</u>
D0	LSELC	Low Select Control register - asserting this bit directs all pod register transactions to the Pod Control Register (U1P and U2N).
D1	LSELV1	Low Select Vector register 1 - asserting this bit directs all pod register transactions to break vector address register 1 (U3K). This register contains the byte that translates to A31-A24 of the break address.
D2	LSELV2	Low Select Vector register 2 - asserting this bit directs all pod register transactions to break vector address register 2 (U4K). This register contains the byte that translates to A23-A16 of the break address.
D3	LSELV3	Low Select Vector register 3 - asserting this bit directs all pod register transactions to break vector address register 3 (U3L). This register contains the byte that translates to A15-A8 of the break address.
D4	LSELV4	Low Select Vector register 4 - asserting this bit directs all pod register transactions to break vector address register 4 (U4L). This register contains the byte that translates to A7-A0 of the break address.
D5	LWMAPEN	Low Write Mapper Enable - asserting this bit directs all pod register writes to the LUSER mapper (U3P, U5P) on the pod processor board.
D6	LFCSEL	Low Function Code Select - (68000/10 pod only) - asserting this bit selects the processor function codes as the least significant address lines to the LUSER mapper on the 68000/10 pod processor board. Deasserting this bit selects the processor address lines A9-11 as the least significant address lines to the LUSER mapper on the 68000/10 pod processor board.
D7	LDMARESET	Low DMA Reset - asserting this bit resets the DMA arbiter FPLA U2H.

The outputs of U1L are tied to the inputs of buffer U2L. This buffer is enabled during performance verification, allowing the contents of the pod index register to be read. U2L is enabled by asserting HDEFIB, HRDPOD, and LEHALT.

POD CONTROL REGISTER (U2N, U1P). This register is used to set up various modes of emulation operation, depending on the user's needs. The Pod Control Register is loaded when the emulation system places the data desired onto the pod data bus, selects the register by asserting LSELC and BLWRTPOD, and then clocks in the data by asserting HWSTB and HDEFIB. During Performance Verification, the register is read by asserting LSELC and LREAD. This causes the output of U1N to be low, which enables U2M and U1O so that the register can be read onto the pod data bus. A description of the bits in the Pod Control Register is shown in the following diagram.



BREAK ADDRESS REGISTER (U3K, U4K, U3L, U4L). U3K and U4K is the break address register that contains the most significant word (A31-16) of the address which is jammed onto the data bus when a break occurs. Likewise, U3L and U4L contain the least significant word (A15-0) of the break address. U3K and U4K are written to by first asserting LSELV1 (bit 1 of the pod index register) and then performing a pod write cycle (assert HWSTB and HDEFIB). In a similar manner, U3L and U4L are written to by setting LSELV3 and then performing a pod write cycle.

Vector Register Pal (U2K) controls the reading of these registers. While HDEFIB is asserted, these registers can be read by asserting the appropriate pod index bit (LSELV1, LSELV2, LSELV3 or LSELV4) and then performing a pod read cycle (assert HRDPOD). During Run Time (i.e. HDEFIB is not asserted), the register contents are jammed onto the data bus when the interrupt vector addresses are read by the processor. A break acknowledge cycle sets the LBRKACK (Low Break Acknowledge) line which sets the first flip-flop in U2G. The output of this flip-flop will assert LINT (Low Interrupt) by way of U1C whenever a processor read cycle occurs. Each assertion of LINT (two for the 68000/10 and four for the 68008) will cause a vector register jam as controlled by the vector register PAL U2K. The last jam asserts LBYTE3 which sets the second flip-flop in U2G which in turn disables LINT.

Following is a description of the vector register decoder PAL (U2K):

INPUTS

A = HDEFIB	B = ODD
C = EVEN	D = A1
E = LSELV1	F = LSELV2
G = LSELV3	H = LSELV4
I = HRDPOD	J = LINT
K = UNUSED	L = LEHALT

OUTPUTS

00 = TRANSPARENT	01 = LBYTE3
02 = LBYTE2	03 = LBYTE1
04 = LBYTE0	05 = LREAD

LOGICAL EQUATIONS ("/" means the signal is low.)

00 = /A+/I
01 = A*/H*I+/A*/J*B*D
02 = A*/G*I+/A*/J*C*D
03 = A*/F*I+/A*/J*B*/D
04 = A*/E*I+/A*/J*C*/D
05 = I*L*A

FUNCTION TABLE PIN LIST

INPUTS												OUTPUTS						
A	B	C	D	E	F	G	H	I	J	K	L	/00	/01	/02	/03	/04	/05	
H	X	X	X	L	H	H	H	H	X	X	X	H	H	H	L	X	X	; READ FROM BYTE0
H	X	X	X	H	L	H	H	H	X	X	X	H	H	L	H	X	X	; READ FROM BYTE1
H	X	X	X	H	H	L	H	H	X	X	X	H	L	H	H	X	X	; READ FROM BYTE2
H	X	X	X	H	H	H	L	H	X	X	X	L	H	H	H	X	X	; READ FROM BYTE3
L	H	H	L	X	X	X	X	X	L	X	X	H	H	L	L	X	X	; 68000 VECTOR JAM 0
L	H	H	H	X	X	X	X	X	L	X	X	L	L	H	H	X	X	; 68000 VECTOR JAM 1
L	L	H	L	X	X	X	X	X	L	X	X	H	H	H	L	X	X	; 68008 VECTOR JAM 0
L	H	L	L	X	X	X	X	X	L	X	X	H	H	L	H	X	X	; 68008 VECTOR JAM 1
L	L	H	H	X	X	X	X	X	L	X	X	H	L	H	H	X	X	; 68008 VECTOR JAM 2
L	H	L	H	X	X	X	X	X	L	X	X	L	H	H	H	X	X	; 68008 VECTOR JAM 3
H	X	X	X	X	X	X	X	H	X	X	H	X	X	X	X	X	X	; TRANSPARENT
H	X	X	X	X	X	X	X	H	X	X	X	X	X	X	X	L	H	; READ

NOTE

A "/" in the table above indicates that the output signals are inverted, i.e., /00 = not 00.

Break Method

There are two methods of performing a processor break. The method of breaking is determined by the user's answer to the "Enable Interrupt level 7 sharing?" configuration question as follows:

An answer of "no" to the configuration question places the user in the "non-sharing" mode. This method asserts a level 7 interrupt without arbitrating the target system level 7 interrupt (same method as the 64242A - first generation 68000 pod). This mode is set by deasserting the LSHAREINT (Low Share Interrupts) line in the pod control register. A break is initiated by asserting HBRK from the control board. This will clear U2D if the HBREAK and HBRKEN bits in the Pod Control Register are asserted. Clearing the first half of U2D will set the second half of U2D by way of U1B. The /Q output of U2D buffered by U1J and sent to the processor board where it initiates a level 7 interrupt (LINT7). Some time later, the processor will respond with a level 7 interrupt acknowledge cycle (break acknowledge cycle). When this occurs, the processor board will assert INTACK and LLEV7 (Low Level 7). These signals, as well as HMEMCYC, are gated together by one half of U1C to produce the LBRKACK or break acknowledge line. This in turn enables the the break address vectors onto the data bus as described previously. At the end of the acknowledge cycle, flip-flop U2E is clocked (cleared) and the break circuit is reset.

An answer of "yes" to the configuration question places the user in the Level 7 Sharing mode. This manner of breaking arbitrates between the target system NMI and the break induced NMI. Furthermore, the control lines are disabled to the target system during the break interrupt acknowledge cycle so it is shielded from these unrequested IACK cycles. Sharing level 7 interrupts requires that the break request be synchronized to the rising edge of LAS (LMEMCYC). This method works fine until a break request occurs during a STOP instruction (which will discontinue the toggling of LAS). In order to perform breaks reliably under these circumstances, a

watchdog timer is started when the break is asserted. After a period of time, if the break does not occur, external interrupts (from the target system) are disabled. If a processor cycle still does not take place after a period of time, the break request is performed asynchronously. This mode is set by asserting the LSHAREINT line in the pod control register. A break is initiated by asserting HBRK from the control board. This will clear U2D if the HBREAK and HBRKEN bits in the Pod Control Register are asserted. The output of this flip-flop is and'ed with the outputs of the two U2A flip-flops which debounce/synchronize the target system NMI. If a target system NMI is not in process, the second half of U2D will be set at the end of the next memory cycle. The /Q output of U2D is buffered by U1J and sent to the Processor board where it initiates a level 7 interrupt. At the same time, the control strobes to the target are disabled by the assertion of HDIS (High Disable) so that the target is shielded from the IACK cycle. Some time later, the processor will respond with a level 7 interrupt acknowledge cycle (break acknowledge cycle). When this occurs, the processor board will assert INTACK. This signal, as well as HMEMCYC (LAS), is gated together by one half of U1C to produce the LBRKACK or break acknowledge line. This in turn enables the break address vectors onto the data bus as described previously. At the end of the acknowledge cycle, flip-flop U2E is clocked (cleared) and the break circuit is reset. Note that LLEV7 does not qualify this mode of breaking.

Counter U1A performs the watchdog function as described previously. The pending break condition as well as LHALT is synchronized by U2J. U1M gates off the counter if the DMA controls LBG or BGACKL are asserted (as well as the LHALT). Assuming that the counter runs to completion, target system interrupts will be turned off when Qh of U1A is high (LINTENB will be deasserted). When both Qh and Qg are high, then the output of U3A will be high. Then, by way of U5D and U1B, the second half of U2D will be set asynchronously and the break will be asserted. This will guarantee that the asynchronous break will not interfere with a target system level 7 interrupt.

DTACK Circuitry

WAIT DETECTION CIRCUITRY. U4H, U2E, U1F, and U4H detect when the processor is in a memory cycle waiting for LDTACK. If a wait state is occurring, the signal HWAITIN (High Wait Input) will be asserted. The assertion of LDTACK or the negation of HMEMCYC (as in an MC6800 type cycle) will cause HWAITIN to be deasserted.

GENERATION OF LDTACK. DTACK timing generation circuitry sends a LDTACK signal to the emulation processor when an addressed device has data ready and is ready to terminate the bus cycle. The original DTACK may come from the user system or the emulation system, and is appropriately timed for the emulation processor depending on the user's needs. Each mode of operation is discussed in the following paragraphs.

- a) For accesses to user memory or I/O, the user will generate the the LDTACK signal. This will be passed to the emulation processor through U4C and U5G whenever LBRKACK is deasserted (break cycle not in progress), LICE is asserted (emulator is plugged into a powered on target system), and LUSERB or LINTACK is asserted (indicating that the current bus cycle refers to a user memory location or a user interrupt stimulus).
- b) If the emulator pod is not in circuit (HICE is not asserted), or a break acknowledge cycle is in process (LBRKACK is asserted), U4G pin 6 will be high, thus selecting a section of U5G for the passage of DTACK.
- c) If the DTACK's are interlocked, the following conditions must be met to generate a DTACK when an emulation memory access takes place:

- 1) HICE is asserted (emulator is plugged into a powered on target).
 - 2) BLDTACK is asserted (user has generated an LDTACK signal).
 - 3) LDTLOCK is asserted (DTACK's are interlocked).
 - 4) LUSER is not asserted (performing an emulation memory access).
 - 5) LBRKACK and LINTACK are not asserted.
 - 6) Output of U2C pin 12 is high (Wait state circuitry).
- d) If the DTACK's are not interlocked, the following conditions must be met to generate a DTACK when an emulation memory access takes place:
- 1) LDTLOCK is not asserted (DTACK's not interlocked).
 - 2) LUSER is not asserted (performing an emulation memory access).
 - 3) LINTACK is not asserted.
 - 4) Output of U2C pin 12 is high (Wait state circuitry).

GENERATION OF XDACK. The XDACK signal, which is output to the target system as an optional DTACK line, will be asserted whenever the DTACK timing is asserted and LUSER is deasserted, indicating an emulation access. This line will typically be used as a DTACK signal for DMA accesses to emulation memory.

DMA Circuitry

DMA ARBITRATION. The DMA arbiter (FPLA U2H) is used to latch bus requests from the user system and the emulation system, and assert the bus request line to the emulator when a request is generated by either. The FPLA contains a superset of the 68000 Bus Arbitration State Table. In non-emulation memory DMA cycles (i.e. DMA tracing and tagging), the FPLA provides the correctly timed DMA strobes so that the target system can perform DMA cycles while the host accesses the emulation memory (EHALT or Emulation Halt).

The target system Bus Request (BLBR) and Bus Grant Acknowledge (BLBGACK) are synchronized to the arbiter by U2I and U2J. U2J also synchronizes the EHALT, LSTART and LDMARESET lines. The address strobe line is synchronized by U4I. The system clock (HCLK) is the arbiter state clock. U3F gates the processor Bus Grant with the FPLA Bus Grant to insure synchronization with the processor. U3F and U4G perform similar functions for LBR and LBGACK. U5B is the HDMA flip-flop which provides an asynchronous indication of DMA operation. This is necessary because the synchronous signals from the arbiter FPLA are too delayed for some of the pod circuitry.

EMULATION HALT (EHALT). For the emulation system to receive bus control, it must pull the EHALT line low. EHALT is buffered and inverted before being input to U2J. U2J synchronizes the EHALT signal with HCLK and inputs the signal into the FPLA arbiter (U2H). The arbiter then causes the bus request line to the processor to be asserted. The emulation system needs to have the emulation processor release the bus so that the mainframe CPU can access into emulation memory without interference from the emulation processor.

DMA TAGGING. U2B, U3A, U3B, U3C, U3D, U4B and U5B implement a feature called "DMA tagging". The user may wish to observe the relative occurrence of DMA activity in his analysis trace. "DMA tagging" will place a state in the emulation analysis unit indicating that the DMA handshake has occurred. This feature requires that the user's LBGACK be asserted for at least 3 high-to-low processor clocks. The first high-to-low clock after LBGACK is asserted will clear the first half of U2B, releasing the clear on U5B and asserting thru U3B the LALTSTB signal which qualifies analysis-only cycles. The next two low-to-high clocks will cause U5B to generate one pulse thru U4B which clocks the analysis unit. The next high-to-low clock (third in sequence) will

reset the first half of U2B releasing LALTSTB and asserting clear to U5B. The second half of U2B will also clear at this time, remaining in this state until LBGACK is negated for at least one clock.

U4A and U4C generate HHMAVEN which prevents EHMAV on the processor board from being asserted. This occurs when the operator has elected to ignore DMA activity or when tagging has been selected.

U3B may also assert LALTSTB when the DMA-monitoring-only mode is selected by the operator.

Tristating the Bus

The 68000 family of processors do not have a halt acknowledge output to indicate that the processor has entered a halt mode and is tristating its address and control lines. The previous version of the 68000 emulator pod (64242A) tristated the address bus and control lines when the Halt line was asserted and Address Strobe was high. It is possible that the processor may decide to do one more memory cycle before it halts (if the halt is asserted late in the cycle). This could cause the pod to tristate the bus before the processor completes the memory cycle. To avoid this problem, tristate control signal LTSEN (Low Target System Enable) can be delayed by a configuration option which results in the assertion of Pod Control Register signal HENDELAY (High Enable Delay).

U1D and U3D are or'ed by U5D so that the output of U5D is high if LHALT or LRESET is asserted and LDMA and LMEMCYC are not asserted. In non-delayed mode, the TRISTATE line for the DMA arbiter is combined by U5D to produce the LTSEN signal. If the HENDELAY option is selected, shift register U5A is enabled and a delay of four clocks is inserted.

MNEMONICS

Table G8-1 is a list of mnemonics (signal names) which are used throughout the theory of operation and on the schematics. They are arranged by functional group names in alphabetical order. Most of the mnemonics have a prefix letter which indicates its active state. These prefixes may be one of the following:

H = Active High
L = Active Low

NOTE

The mnemonic descriptions in Table G8-1 are applicable to both the 68000 and the 68008. However, the reference designations for the IC's are essentially for the 68000 processor board and may not fully agree with an equivalent IC reference designation on the 68008 processor board.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
TO/FROM TARGET SYSTEM	
BA1-BA23	Buffered Address 1 through Buffered Address 23 - output to the target system from the emulation processor address bus. These lines are driven by transceivers U2A, U3A, and U4A. The target system may drive these lines into the emulator for DMA accesses to emulation memory.
BD0-BD15	Buffered Data 0 through Buffered Data 15 - 16 bits of data information to/from the target system data bus. These lines are driven by transceivers U2C, and U3C. Control of these buffers may be assumed by the target system during DMA cycles.
BE	Buffered Enable - output to the target system from the emulation processor through buffer U2K. This signal is used as an enable clock by MC6800-type peripherals.
BFC0-BFC2	Buffered Function Code 0 through Buffered Function Code 2 - output to the target system from the emulation processor through bidirectional buffer U5A. The buffer allows the target system to drive the function lines into the emulator during DMA cycles. The function codes indicate the mode of operation of the emulation processor and also indicate what type of bus cycle is being executed.
BLAS	Buffered Low Address Strobe - output to the target system from the emulation processor through bidirectional buffer U4C. The bidirectional buffer allows the target system to drive this line as an input to the emulator during DMA cycles. When low, BLAS indicates that the device with bus control has placed a valid address on the bus.
BLBERR	Buffered Low Bus Error - input from the target system to the BERR circuitry on the processor board. When low, indicates to the emulation processor that a problem exists with the current bus cycle.
BLBG	Buffered Low Bus Grant - output from the DMA circuitry on the pod timing board to the target system. When low, indicates that the emulation processor will give up the system bus to the requesting device.
BLBGACK	Buffered Low Bus Grant Acknowledge - input to the DMA circuitry on the pod timing board from the target system. When low, indicates that the device which requested and was granted the system bus has assumed bus control. Bus control will not be terminated by the device until BLBGACK is negated.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
BLBR	Buffered Low Bus Request - input to the DMA arbiter circuitry on the pod timing board from the target system. When low, indicates that a peripheral device in the target system wishes to have control of the system bus.
BLDTACK	Buffered Low Data Transfer Acknowledge - input to the DTACK circuitry on the emulator pod timing board. BLDTACK is pulsed low by the target system to indicate that the addressed device has data ready and the bus cycle is complete. This line may be used as an output if the DTACK lock switch on the emulator is closed. In this case, BLDTACK will be pulsed low by the emulator during a DMA access to emulation memory, indicating that the cycle is completed.
BLHALT	Buffered Low Halt - bidirectional line between the emulation processor and the target system. When this line is pulled low, the emulation processor will stop executing instructions at the end of the current bus cycle. The emulation processor may drive this line to the target system to indicate that it has stopped, such as during a bus fault. This line is only bidirectional when the halt switch on the emulator is set to "unbuffered". Otherwise, the user's system will be shielded from the processor's assertion of the signal.
BLIPL0-BLIPL2	Buffered Low Interrupt Priority Level 0 through Buffered Low Interrupt Priority Level 2 - input to the interrupt gating logic on the processor board from the user's target system. These lines indicate the binary priority level (active low true) of an interrupt request.
BLLDS	Buffered Low Lower Data Strobe - output from the emulation processor through bidirectional buffer U4C. The bidirectional buffer is used so that the target system may drive this line into the emulator during DMA cycles. When low during a write cycle, indicates that valid data is present on the lower 8 bits of the data bus. When low during a read cycle, indicates a request for data from the lower byte bank of memory.
BLRESET	Buffered Low Reset - bidirectional line between the emulation processor and the target system. When this line is pulled low, the emulation processor will start a reset initialization sequence. The emulation processor may drive this line to the target system by executing a RESET instruction. This line is only bidirectional when the reset switch on the emulator is set to "unbuffered". Otherwise, the user's system will be shielded from the processor's assertion of the signal.
BLUDS	Buffered Low Upper Data Strobe - output from the emulation processor through bidirectional buffer U4C. The bidirectional buffer is used so that the target system may drive this line into the emulator during DMA cycles. When low during a write cycle, indicates that valid data is present on the upper 8 bits of the data bus. When low during a read cycle, indicates a request for data from the upper byte bank of memory.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
BLVMA	Buffered Low Valid Memory Address - output from the emulation processor through bidirectional buffer U4C. The bidirectional buffer is used so that the target system may drive this line into the emulator during DMA cycles. When low, indicates that a valid address for an MC6800 type peripheral device has been placed on the bus. BLVMA will not be asserted until the emulation processor has received BLVPA.
BLVPA	Buffered Low Valid Peripheral Address - input to the emulator VPA gating on the processor board from the target system. When low, indicates that the address currently on the bus corresponds to an address range reserved for MC6800 type peripheral devices.
BR/LW	Buffered Read / Low Write - output to the target system from the processor through bidirectional buffer U4C. The bidirectional buffer is used so that the target system may drive this line into the emulator during DMA cycles. When high, the emulation processor wishes to read data from the addressed device; when low, the emulation processor wishes to write data to the addressed device.
BVCC	Buffered User Supply Voltage - this is the target system +5V supply voltage. It is sensed by the emulator pod to determine whether or not the emulator is plugged into a powered-on user target system.
BVSS	Buffered User Ground - connects emulation ground to target system ground.
LUSERMEM	Low User Memory - output provided for the user as an optional memory enable signal. When the mapper RAM on the processor board has determined that the current emulator address resides in area mapped as user memory space, this line will be driven low.
LXDTACK	Low External Data Transfer Acknowledge - output provided for the user as an optional DTACK line for DMA accesses. This output may be gated directly onto BLDTACK by closing a switch on the emulator or it may be left unconnected. When low, the addressed device (in this case emulation memory) has data ready and is ready to terminate the bus cycle.
XCLK	External Clock - input to the emulator clock select circuitry from the target system. This circuitry selects either UCLK or the output of a 10 MHz crystal (Y1) as the clock input of the emulation processor.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
TO/FROM CONTROL BOARD	
BAE	Buffered Address Equal - output of the depipeline circuitry on the pod processor board. The six 4-bit counters U2L, U3L, U4M, U4L, U4O, and U4N increment on each program fetch and the output of these counters is compared to the next processor address. If the two values are equal, then BAE is asserted. This signal is input to the depipeline circuitry on the control board.
BCDBA	Buffered Control Data Bus A - buffered version of CDBA. This signal is generated from the transceiver control logic on the pod timing board and input to the emulation control board data transceivers.
BCDDB	Buffered Control Data Bus A - buffered version of CDBB. This signal is generated from the transceiver control logic on the pod timing board and input to the emulation control board data transceivers.
BERRL	Bus Error Low - output of flip-flop U2N on the pod processor and input to the control board. This signal is the result of synchronizing the processor's bus error signal LBERR, with LDS.
BLDS	Buffered Low Data Strobe - active low signal which is the output of buffer U2P on the pod processor board. Buffered and inverted version of DS.
CLKE	Clock E - output of buffer U2K on the pod processor board and input to the depipeline circuitry on the control board.
EHALTB	Emulation Halt Buffered - output from the control board control register and input to the pod timing board (through buffer U1J). When high, asserts a bus request to the emulation processor through the DMA arbitration circuitry. This feature allows the mainframe CPU to access emulation memory while "holding-off" the emulation processor.
EHMAV	Emulator High Memory Available - output from U2P on the pod processor board and input to the memory available circuitry on the control board. When high, indicates that the emulation processor is not performing a bus cycle.
HBUSGTIN	High Bus Grant Input - input to control board from the DMA circuitry on the pod timing board. When high, indicates that the emulation processor has granted the bus to a requesting device. HBUSGTIN is the inverted version of LDMA.
HCLK	High Clock - output from buffer U2K on the pod processor board. HCLK is the buffered version of the processor CLK signal. Used to clock the data transfer acknowledge and DMA tracing options circuits on the pod timing board. HCLK is also input to the depipeline circuitry on the control board.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
HEA0-HEA23	High Emulation Address 0 through High Emulation Address 23 - 24 bits of emulation processor address information, active high. These lines are output from address transceivers U5M, U5N, and U5O on the pod processor board.
HRESETIN	High Reset Input - input to the control board from the pod timing board. When high, indicates that the LRESET line has been asserted by one of three methods: 1. The emulation processor has executed a RESET instruction. 2. LDEFIB has been asserted. 3. A target system device has asserted LRESET.
HWAITIN	High Wait Input - input to the control board from the wait state detector on the pod timing board. When high, indicates that the emulation processor is waiting for the LDTACK signal.
HWPINDEX	High Write Pod Index - output from the data direction logic on the control board to the pod timing board. When high, the Pod Index Register on the timing board will be written to by the mainframe CPU.
HVRTSTB	High Write Strobe - output from the data direction logic on the control board to the pod timing board. When high, the addressed register on the emulator pod will be written to by the mainframe CPU.
ID0-ID1	Identification 0 through Identification 1 - output to the control board indicating what type of emulator pod is connected to the control board.
LANALIN	Low Analysis Input - analysis strobe input to the control board. When a low-to-high transition occurs, the analysis board is instructed to record data present on the address, data, and status buses.
LBRK	Low Break - output from the break circuitry on the control board and input to inverter U10 on the pod processor board. When a high-to-low transition occurs on this line, the emulation system is requesting a level 7 (highest priority) interrupt of the emulation processor.
LBUP	Low Byte Upper - output from buffer U2P on the pod processor board and input to the upper byte enable logic on the control board. When low during a write cycle, indicates that data is valid on the upper 8 bits of the data bus. When low during a read cycle, indicates a request for data from the upper byte bank of memory.
LBYTE	Low Byte - output from buffer U2P on the pod processor board and input to the lower byte enable logic on the control board. When low during a write cycle, indicates that data is valid on the lower 8 bits of the data bus. When low during a read cycle, indicates a request for data from the lower byte bank of memory.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
LCLEAR	Low Clear - output from the depipeline circuitry on the control board and input to buffer U2K on the pod processor board. This signal resets the depipeline synchronization clocks.
LDE0-LDE15	Low Data Emulator 0 through Low Data Emulator 15 - bidirectional data bus between the pod timing board and the control board. The emulation processor uses this bus to communicate with emulation memory.
LDEFIB	Low Defibrillate - input to buffer U1N on the pod processor board from the control board. Used as an active low reset signal for the emulator pod.
LESTAT4-LESTAT7	Low Emulation Status 3 through Low Emulation Status 7 - output to the control board from status buffer U4K on the pod processor board. Included are status bits FC0-FC2, HDMA, and LINTACK.
LEWRT	Low Emulator Write - output from buffer U2P on the pod processor board and input to the control board. When low, indicates that the current emulation processor bus cycle is a write cycle.
LICE	Low In-Circuit Emulation - input to the emulation control board. When low, indicates that the emulator is plugged into a powered-on user target system.
LHALT	Low Halt - buffered version of BLHALT which is generated on the pod processor board. When this status line is low, the emulation processor will stop executing instructions at the end of the next bus cycle. This line can be asserted either when the user has requested a halt, when HDEFIB is asserted, or when a double bus fault has occurred. This signal is input to both the timing board and the control board.
LHALTIN	Low Halt Input - input to control board from the halt status decoder on the pod timing board. When low, indicates that the emulation processor has stopped execution of instructions in response to either a double bus fault; the LHALT line has been asserted through LDEFIB or by an external device; or the LEHALTB line has been asserted by the host processor.
LRDPOD	Low Read Pod - output from the data direction logic on the control board to the pod processor board. When low, the addressed register on the emulator pod will be read by the mainframe CPU.
LSTARTB	Low Start Buffered - output from gate U2E on the pod processor board and input to the depipeline circuitry on the control board. This signal is asserted when both LHALT and LRESET are asserted indicating a total system reset.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
LWDV	Low Write Data Valid - output from U2P on the pod processor board and input to the LWDV circuitry on the control board. When low, indicates that the emulation processor has placed valid data for a memory write on the bus.
LWRTPOD	Low Write Pod - output from the data direction logic on the control board to the pod timing board. When low, the addressed register on the emulator pod will be written to by the mainframe CPU.
P	Program - output from the depipeline circuitry on the control board and input to buffer U2K on the pod processor board. When asserted, this signal indicates that the current cycle is a program fetch.

TIMING BOARD TO/FROM PROCESSOR BOARD

AS	Address Strobe - inverse of LAS which indicates, when high, that the processor has started a bus cycle and has placed a valid address on the bus.
BBG	Buffered Bus Grant - output of the DMA arbitration circuitry on the pod timing board and input to the pod processor board. This signal is used to generate the user's Bus Grant (BLBG). When low, indicates that the emulation processor will release control of the bus at the end of the current bus cycle.
BLBGACK	Buffered Low Bus Grant Acknowledge - input to the DMA circuitry on the pod timing board from the target system. When low, indicates that the device which requested and was granted the system bus has assumed bus control. Bus control will not be terminated by the device until BLBGACK is negated.
BLBR	Buffered Low Bus Request - input to the DMA arbiter circuitry on the pod timing board from the target system. When low, indicates that a peripheral device in the target system wishes to have control of the system bus.
BLDTACK	Buffered Low Data Transfer Acknowledge - input to the DTACK circuitry on the emulator pod timing board. BLDTACK is pulsed low by the target system to indicate that the addressed device has data ready and the bus cycle is complete.
DO-D15	Data 0 through Data 15 - bidirectional data bus between the emulation processor and data transceivers U3M, U3N, U4M, and U4N on the pod timing board. The processor and the emulation system both use this bus to transfer data information.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
DS	Data Strobe - Inverted version of LDS. Active high signal which is true when either the LUDS and/or the LLDS signals are low.
DSLATCH	Data Strobe Latch - output of U4E on the pod processor board which is a latched version of DS. The falling edge of DS is synchronized to the processor clock (HCLK) to create this signal.
EVEN	Even Address - Active high signal generated by the 68008 uP board indicating an even address (A0 is low). On the 68000 and 68010 uP boards this signal is always high.
HBRK	High Break - inverted version of LBRK which is output from the control board, passes through inverter U1O on the pod processor board, and is then sent to the timing board's level 7 break logic. When a low-to-high transition occurs on this line, the emulation system is requesting a level 7 (highest priority) interrupt of the emulation processor.
HCLK	High Clock - output from buffer U2K on the pod processor board. HCLK is the buffered version of the processor CLK signal. Used to clock the data transfer acknowledge and DMA tracing options circuits on the pod timing board.
HDEFIB	High Defibrillate - Inverted version of LDEFIB. An active high signal generated on the processor board which is used as a reset signal for the emulation pod.
HDIS	High Disable - Output of the interrupt level 7 break circuitry of the pod timing board and input to the processor board which disables control strobes AS, UDS, LDS, R/LW, and VMA from reaching the target system. This prevents the target from receiving unwanted interrupt acknowledge cycles due to an emulation break.
HDMA	High Direct Memory Access - output of flip-flop U5B on the pod timing board which provides an asynchronous indication of DMA operation. When high, indicates that the emulation processor has released the bus to a requesting device in the user's target system. This signal is used on the pod processor board to change the direction of the address, data, and control buffers which interface to the target system so that the target system may drive these lines to the emulator.
HHMAVEN	High High Memory Available Enable - output from U4C in the DMA tracing options logic, and sent to the pod processor board. When high, EHMAV will be allowed to toggle normally; when low, EHMAV will be forced high. This is used to inhibit EHMAV from toggling the analysis clock during DMA operations unless it is desired to see the DMA handshake, in which case EHMAV will be allowed to toggle only during the handshake.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
HICE	High In-Circuit Emulation - input to the DTACK and DMA arbitration circuitry on the pod timing board. This signal is driven high when the user plug is inserted into a powered-on target system. Used to enable the user's DTACK and Bus Request.
HICLK	High Internal Clock - inverted version of LICLK which is output from the Pod Control Register on the pod timing board. This signal is input to the clock select relay on the pod processor board. When high, the output of the 10 MHz crystal is selected as the clock input to the MC68000 processor.
INTACK	Interrupt Acknowledge - Inverse of LINTACK. This active high signal is valid when A4-A23 and FC0-FC2 are all high as in a processor interrupt cycle. It is used to disable EHMAV on the pod processor board. It is also used to control data buffers on the pod timing board.
LALTDS	Low Alternate Data Strobe - output from the DMA tracing options circuitry. Used to toggle the EHMAV signal to the emulator low during a DMA handshake, which will cause the analysis unit to record the handshake.
LALTSTB	Low Alternate Strobe - output from the DMA tracing options circuitry on the pod timing board. This line is used to block DMA accesses to emulation memory, but allows analysis to record either the DMA handshakes or DMA bus cycles as desired by the user. Emulation memory accesses will be blocked when this line is low, which blocks EHMAV from reaching the emulation bus.
LAS	Low Address Strobe - this line is output by the emulation processor to indicate, when low, that the processor has started a bus cycle and has placed a valid address on the bus. LAS is used to derive HMEMCYC and LMEMCYC, which are used to synchronize various circuits on the pod timing board to the processor's bus cycles.
LBERRREN	Low Bus Error Enable - output from the pod control register on the pod timing board and input to the BERR enable/disable circuitry on the pod processor board. When this line is low, it allows the emulation processor to respond to the user BERR signal during emulation memory accesses, which is not the case when LBERRREN is false (high).
LBG	Low Bus Grant - output from the emulation processor and input to the DMA circuitry on the timing board. When low, indicates that the emulation processor will release control of the bus at the end of the current bus cycle.
LBGACK	Low Bus Grant Acknowledge - input to the emulation processor from the DMA arbiter on the pod timing board. U4G gates the user's BGACK with the BGACK generated from FPLA (U2H) to produce LBGACK. When low, indicates that a device in the target system that requested bus usage has assumed bus control and that the emulation processor should not attempt to use the bus until the signal is negated.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
LBR	Low Bus Request - input to the emulation processor from the DMA arbiter on the pod timing board. U4G gates the user's BR with the BR generated from FPLA (U2H) to produce LBR. When low, either the emulation system or the user system has requested use of the system bus.
LBRKACK	Low Break Acknowledge - output from the level 7 interrupt (break) circuitry. This line is used to indicate that an interrupt acknowledge cycle is in process in response to an emulation system break. LBRKACK is used to enable the output of the break address vectors onto the data bus; enable generation of LDTACK so that the emulation processor can latch the interrupt address; disable LPDBEN so that user data will not corrupt the interrupt vector data; and indicate to the VPA/BERR circuitry that the current interrupt acknowledge cycle is NOT in response to a user-generated interrupt (which effectively disables VPA and BERR).
LDMA	Low Direct Memory Access - inverted version of HDMA generated on the pod timing board which provides an asynchronous indication of DMA operation. When low, indicates that the emulation processor has released the bus to a requesting device in the user's target system.
LDTACK	Low Data Transfer Acknowledge - output from the data transfer acknowledge (DTACK) timing logic. Input to the emulation processor. When low, the addressed device is ready to complete the data transfer and finish the bus cycle.
LFCSEL	Low Function Code Select - output of bit D6 of the Pod Index Register on the timing board (U1L). When this signal is at a logic low, LESTAT4-6 (Low Emulation Status 4-6), an inverted and latched version of FC0-2 drive addresses A0-2 of the user mappers on the processor board (U3P, U5P). When at a logic high, HEA 9-11 drive addresses A0-2 of the user mappers.
LHALT	Low Halt - buffered version of BLHALT which is generated on the pod processor board. When this status line is low, the emulation processor will stop executing instructions at the end of the next bus cycle. This line can be asserted either when the user has requested a halt, when HDEFIB is asserted, or when a double bus fault has occurred. This signal is input to both the timing board and the control board.
LINT7	Low Interrupt 7 - output from the level 7 interrupt (break) logic on the pod timing board. This line is sent to the interrupt enable/disable logic on the pod processor board. When low, a level 7 (highest priority) interrupt will be asserted on emulation processor interrupt lines LIPL0-LIPL2.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
LINTACK	Low Interrupt Acknowledge - Active low signal generated by U2E on the pod processor board. This signal is valid when A4-A23 and FC0-FC2 are all high as in a processor interrupt cycle. It is used to enable VPA and BERR during an interrupt acknowledge cycle. It is also used to control the level 7 interrupt circuitry and the DTACK circuit.
LINTEN	Low Interrupt Enable - output from the Pod Control Register. This signal is asserted according to the user's answer to the "Enable emulator processor interrupts?" configuration question. An answer of "yes" results in the assertion of this signal. An answer of "no" will result in user interrupts (all levels) being masked off by hardware in the pod.
LINTENB	Low Interrupt Enable Buffered - output of U3G on the timing board. This signal is disabled when the the emulator is out-of-circuit, when selected by a bit in the Pod Control Register, or when an asynchronous break is taking place. This signal is used to allow interrupts from the user system to drive the interrupt inputs to the emulation processor.
LIPL7	Low Interrupt Priority Level 7 - output from U2J and U2F on the processor board and input to the level 7 interrupt circuitry on the pod timing board. This signal is derived from the LIPL0-LIPL2 lines and indicates that a level 7 interrupt is taking place. LIPL7 holds off an emulation system break until the user's level 7 interrupt has been serviced.
LLEV7	Low Level 7 - Output from U1L and U1F on the pod processor board and input to the level 7 interrupt circuitry on the pod timing board. When low, indicates that the address on the lower three bits of the emulation processor's address bus is 111, which, when gated with LINTACK, indicates that the emulation processor is in a level 7 interrupt acknowledge cycle.
LMAPEN	Low Write Mapper Enable - output of bit D5 of the Pod Index Register on the timing board (U1L). Asserting this signal directs all pod register writes to the USER mapper on the pod processor board (U3P, U5P).
LPDBEN	Low Probe Data Bus Enable - Active low signal generated by U5E of the pod timing board. When low, this signal enables the processor board's user data bus transceivers (U2C, U3C).
LRDPOD	Low Read Pod - output from the data direction logic on the control board to the pod processor and timing boards. Whe low, the addressed register on the emulator pod will be read by the mainframe CPU.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
LRESET	Low Reset - input to the status buffers, data rec/xmt and break vector control logic on the pod timing board. When low, indicates either that the BLRESET line from the user system has been asserted or the emulation processor has executed a RESET instruction.
LSTART	Low Start - output from gate U2E on the pod processor board and input to the DMA arbitration circuitry on the pod timing board. This signal is asserted when both LHALT and LRESET are asserted indicating a total system reset.
LTSEN	Low Target System Enable - output from the pod timing board which is used to control the user interface buffers for address and control on the pod processor board. When low, all user interface buffers will be enabled. This line may be negated by a reset or halt condition. It may also be negated when the emulation processor has bus control, but is not executing bus cycles.
LUSER	Low User - output to the timing board from the high speed mapper RAMs on the processor board. When low, indicates that the address present on the bus resides within an address range mapped to user memory or I/O. Used to properly orient data buffers for the bus transaction.
LUSERHO	Low User Hold-off - output of the pod control register on the timing board and input to U1K on the pod processor board. When asserted, EHMAV will be held off for all target memory accesses (applies only to the MC68010 microprocessor).
LWSTB	Low Write Strobe - inverted version of the control board generated signal HWRTSTB. LWSTB is output from inverter U1K on the pod timing board and input to the user mapper control logic on the pod processor board. When low, the memory mappers (U3P and U5P) will be written to by the mainframe CPU.
ODD	Odd Address - Active high signal generated by the 68008 uP board indicating an odd address (A0 is high). This signal is always high on the 68000 and 68010 uP boards.
PDBDIR	Probe Data Bus Direction - user data bus control signal generated by U5E of the pod timing board. When low, this signal directs the flow of data through the user data bus transceivers (U2C, U3C) from the emulation processor onto the target system address bus as in a data write to the target system. But when this signal is high, the data bus is driven from the target system to the emulation processor as in a data read from the target system.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
READ	Read - Active high signal generated on the processor board. This output of U4E indicates that a read cycle is taking place.
R/LW	Read/ Low Write - output from the emulation processor and input to the data direction control logic on the pod timing board (user data xmt/rec transceiver controls). When high, the emulation processor wishes to read data from the addressed device, and when low, the emulation processor wishes to write data to the addressed device.
WRITE	Write - Active high signal generated on the processor board. This output of U4E indicates that a write cycle is taking place.
XDTACK	External Data Transfer Acknowledge - output from the DTACK circuitry to the target system through open collector inverter U2D. When high, indicates that emulation memory has completed the data transfer in response to the read or write operation and is ready to terminate the bus cycle. This signal is provided as an extra DTACK for DMA accesses by target system devices to emulation memory.

INTERNAL SIGNALS - PROCESSOR BOARD

A0-A23	Address 0 through Address 23 - address information output from the emulation processor. These lines are driven to the emulation system through latches U5M, U5N, and U5O. They are driven to the target system through transceivers U2A, U3A, and U4A. The target system can drive these lines through the target system transceivers whenever the emulation processor has granted the bus to a target system device.
CLK	Clock - input to the emulation processor from the clock select circuitry. This line may either be driven by the 10 MHz crystal oscillator internal to the emulator or by the user clock input to the emulator.
E	Enable - output from the emulation processor to the target system through buffer U2K. The period for this output is ten MC68000 clock periods. E is used as an enable clock for MC6800 type peripheral devices.
FC0-FC2	Function Code 0 through Function Code 2 - input to the interrupt acknowledge decoder and user buffers on the pod processor board. These lines are driven by the emulation processor to indicate what type of bus cycle the processor is executing and what mode of operation the processor is in.
LBERR	Low Bus Error - input to the emulation processor from the BERR enable/disable circuitry. When low, the target system is indicating to the emulation processor that there is a problem with the current bus cycle.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
LDS	Low Data Strobe - output of U1I on the pod processor board. Active low signal which is true when either the LUDS and/or the LLDS signals are low.
LIPL0-LIPL2	Low Interrupt Priority Level 0 through Low Interrupt Priority input to the emulation processor from the interrupt enable/ disable logic, these lines indicate the binary encoded priority level of an interrupt request of the emulation processor.
LLDS	Low Lower Data Strobe - output from the emulation processor to bidirectional buffer U4C. The bidirectional buffer is used so that the target system may drive this line into the emulator during DMA cycles. When low during a write cycle, indicates that valid data is present on the lower 8 bits of the data bus. When low during a read cycle, indicates a request for data from the lower byte bank of memory.
LPV1	Low Performance Verification 1 - a PV latch network control signal. If LPV1 is asserted, the contents of PV latches U3B and U4B are output to BD0-15 when an address with A1=0 is read.
LPV2	Low Performance Verification 2 - a PV latch network control signal. If LPV2 is asserted, the contents of PV latch U2B is output to BD0-7 and the contents of latch U1A to BD8-10 when an address with A1=1 is read.
LUDS	Low Upper Data Strobe - output from the emulation processor to bidirectional buffer U4C. The bidirectional buffer is used so that the target system may drive this line into the emulator during DMA cycles. When low during a write cycle, indicates that valid data is present on the upper 8 bits of the data bus. When low during a read cycle, indicates a request for data from the upper byte bank of memory.
LVPA	Low Valid Peripheral Address - input to the emulation processor from the VPA enable/disable circuitry. When low, indicates that the address sent to the target system is within an address range used exclusively for MC6800 type peripheral devices.
SA0-SA2	Select Address 0 through Select Address 2 - outputs of multiplexor U4P which select one of two modes of memory mapping. When LFCSEL is deasserted, SA0-SA2 equal HEA9-HEA11. When LFCSEL is asserted, SA0-SA2 equal LESTAT4-6. SA0-SA2 drive addresses A0-2 of user mappers U3P and U5P.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
INTERNAL SIGNALS - TIMING BOARD	
CDBA	Control Data Bus A - a transceiver control signal which is buffered by U1J to produce BCDBA, an input to the emulation control board data transceivers.
CDBB	Control Data Bus B - a transceiver control signal which is input to U1D to qualify the value of LEVENSEL and LODDSEL. CDBB has to be high for data to be received by the emulation pod from the emulation control board. CDBB is buffered by U1J to produce BCDBB, an input to the emulation control board data transceivers.
HBLKDMA	High Block Direct Memory Address - output from the pod control register on the pod timing board and input to the DMA tracing options logic. When high, DMA accesses to the emulation bus will not be seen by analysis and will not be responded to by emulation memory.
HBREAK	High Break - output from the Pod Control Register and input to the break circuitry. This signal is set according to the user's answer to the "Enable emulator use of INT7?" configuration question. An answer of "yes" results in the assertion of this signal. When this signal is asserted, a low will be clocked into the interrupt flipflop U2D by the HBRK signal from the emulation system, thus sending a level 7 interrupt to the emulation processor. If this signal is low, a high will be clocked into the flipflop and no interrupt will occur.
HBRKEN	High Break Enable - output from the Pod Control Register and input to the break circuitry. This signal is always active high.
HDMAMON	High Direct Memory Access Monitor - output from the pod control register and input to the DMA tracing options circuitry. When high, the analysis module will be able to see all DMA transactions, however, emulation memory will not respond to any of these transactions.
HENDELAY	Halt Enable Delay - output from the pod control register and input to the halt delay logic. When asserted, the LHALT line will be delayed from reaching the processor. This option is available because there is the possibility of pulling HALT active late in the current bus cycle and one more cycle may execute before the processor halts. In this case, a delay is desirable to allow the processor to halt before any activity occurs on the bus.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
HMEMCYC	High Memory Cycle - derived from LAS. When high, indicates that the emulation processor (or DMA device) is performing a bus cycle.
HRDPOD	High Read Pod - inverted version of LRDPOD. When high, the addressed register on the emulator pod will be read by the mainframe CPU.
HTAGEN	High Tag Enable - output from the pod control register and input to the DMA tracing options logic. When high, the analysis unit will be able to see the DMA handshake sequence, but not DMA bus cycles. Emulation memory is not allowed to respond in either case.
HUSER	High User - inverted version of LUSER. When high, indicates that the address present on the bus resides within an address range mapped to user memory or I/O. Used to properly orient data buffers for the bus transaction.
LBYTE0	Low Byte 0 - output of the Vector Register PAL (U2K) on the timing board. This signal enables a byte of the Break Address Register (U3K) to be read. This register contains the byte that translates to A31-A24 of the break address. LBYTE0 causes the register contents to be jammed onto the bus when the emulation processor is attempting to read the interrupt vector address.
LBYTE1	Low Byte 1 - output of the Vector Register PAL (U2K) on the timing board. This signal enables a byte of the Break Address Register (U4K) to be read. This register contains the byte that translates to A23-A16 of the break address. LBYTE1 causes the register contents to be jammed onto the bus when the emulation processor is attempting to read the interrupt vector address.
LBYTE2	Low Byte 2 - output of the Vector Register PAL (U2K) on the timing board. This signal enables a byte of the Break Address Register (U3L) to be read. This register contains the byte that translates to A15-A8 of the break address. LBYTE2 causes the register contents to be jammed onto the bus when the emulation processor is attempting to read the interrupt vector address.
LBYTE3	Low Byte 3 - output of the Vector Register PAL (U2K) on the timing board. This signal enables a byte of the Break Address Register (U4K) to be read. This register contains the byte that translates to A7-A0 of the break address. LBYTE3 causes the register contents to be jammed onto the bus when the emulation processor is attempting to read the interrupt vector address.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
LDMAEN	Low Direct Memory Access Enable - output from the pod control register and input to the DMA arbiter. When low, bus request signals from the user target system will be allowed to enter the arbiter; when high, bus requests from the user system will be ignored.
LDMARESET	Low DMA Reset - output of bit D7 of the Pod Index Register. Asserting this signal resets the DMA arbiter FPLA U2H.
LDTLOCK	Low Data Transfer Lock - output from the Pod Control Register. This signal is asserted according to the user's answer to the "Interlock emulation memory DTACK with user DTACK?" configuration question. An answer of "yes" results in the assertion of this signal. If asserted, the user must assert BLDTACK for accesses to emulation memory. This allows LDTACK to be received by the emulation processor. When not asserted, the LDTACK signal will be supplied by emulation memory.
LDXMT	Low Data Transmit - output from the transceiver control logic and input to the data transceivers. When low, data will be transmitted to the emulation control board by the emulation pod.
LEVENSEL	Low Even Select - output from the transceiver control logic and input to data transceivers U3M and U3N. When low, the emulation pod will receive data bits D8 - D15 from the emulation control board.
LICE	Low In-Circuit Emulation - input to the DMA arbiter and break circuitry. When low, indicates that the emulator is plugged into a powered-on user target system.
LICLK	Low Internal Clock - output from the Pod Control Register on the pod timing board. When low, the output of the 10 MHz crystal is selected as the clock input to the MC68000 microprocessor.
LINT	Low Interrupt - output of U1C and input into the the Vector Register PAL (U2K) for break vector jamming.
LINTEN	Low Interrupt Enable - output from the Pod Control Register. This signal is asserted according to the user's answer to the "Enable emulator processor interrupts?" configuration question. An answer of "yes" results in the assertion of this signal. An answer of "no" will result in user interrupts (all levels) being masked off by hardware in the pod.
LMEMCYC	Low Memory Cycle - inverse of HMEMCYC. When low, indicates that the emulation processor (or DMA device) is performing a bus cycle.
LODDSEL	Low Odd Select - output from the transceiver control logic and input to data transceivers U4M and U4N. When low, the emulation pod will receive data bits D0-D7 from the emulation control board.

Table G8-1. Mnemonics

<u>Mnemonic</u>	<u>Description</u>
LREAD	Low Read - Active low signal which is the output of the Vector Register Decoder PAL (U2K). This signal is equal to the logical Equation: $\text{HRDPOD} * \text{LEHALT} * \text{HDEFIB}$. When LREAD is low, one of the pod registers is read by the mainframe CPU.
LSELC	Low Select Control Register - output of bit D0 of the Pod Index Register. Asserting this signal directs all pod register transactions to the Pod Control Register (U1P and U2N).
LSELV1	Low Select Vector Register 1 - output of bit D1 of the Pod Index Register. Asserting this signal directs all pod register transactions to the Break Vector Address Register 1 (U3K). This register contains the byte that translates to A31-A24 of the break address.
LSELV2	Low Select Vector Register 2 - output of bit D2 of the Pod Index Register. Asserting this signal directs all pod register transactions to the Break Vector Address Register 2 (U4K). This register contains the byte that translates to A23-A16 of the break address.
LSELV3	Low Select Vector Register 3 - output of bit D3 of the Pod Index Register. Asserting this signal directs all pod register transactions to the Break Vector Address Register 3 (U3L). This register contains the byte that translates to A15-A8 of the break address.
LSELV4	Low Select Vector Register 4 - output of bit D4 of the Pod Index Register. Asserting this signal directs all pod register transactions to the Break Vector Address Register 4 (U4L). This register contains the byte that translates to A7-A0 of the break address.
LSHAREINT	Low Share Interrupts - output from the Pod Control Register and input to the break circuitry. This signal is set according to the user's answer to the "Enable Interrupt Level 7 Sharing?" configuration question. An answer of "yes" results in the assertion of this signal.
LW0-LW1	Low Wait 0 through Low Wait 1 - output from the pod control register and input to the DTACK wait state generator. These control bits determine the number of wait states inserted before a DTACK signal is sent to the emulation processor.
TRANS	Transparent - Active high signal which is the output of the Vector Register Decoder PAL (U2K). This signal is equal to the logical equation: $\text{HDEFIB} + \text{HRDPOD}$. When TRANS is high, data latches U3J and U4J are made transparent.

NOTES

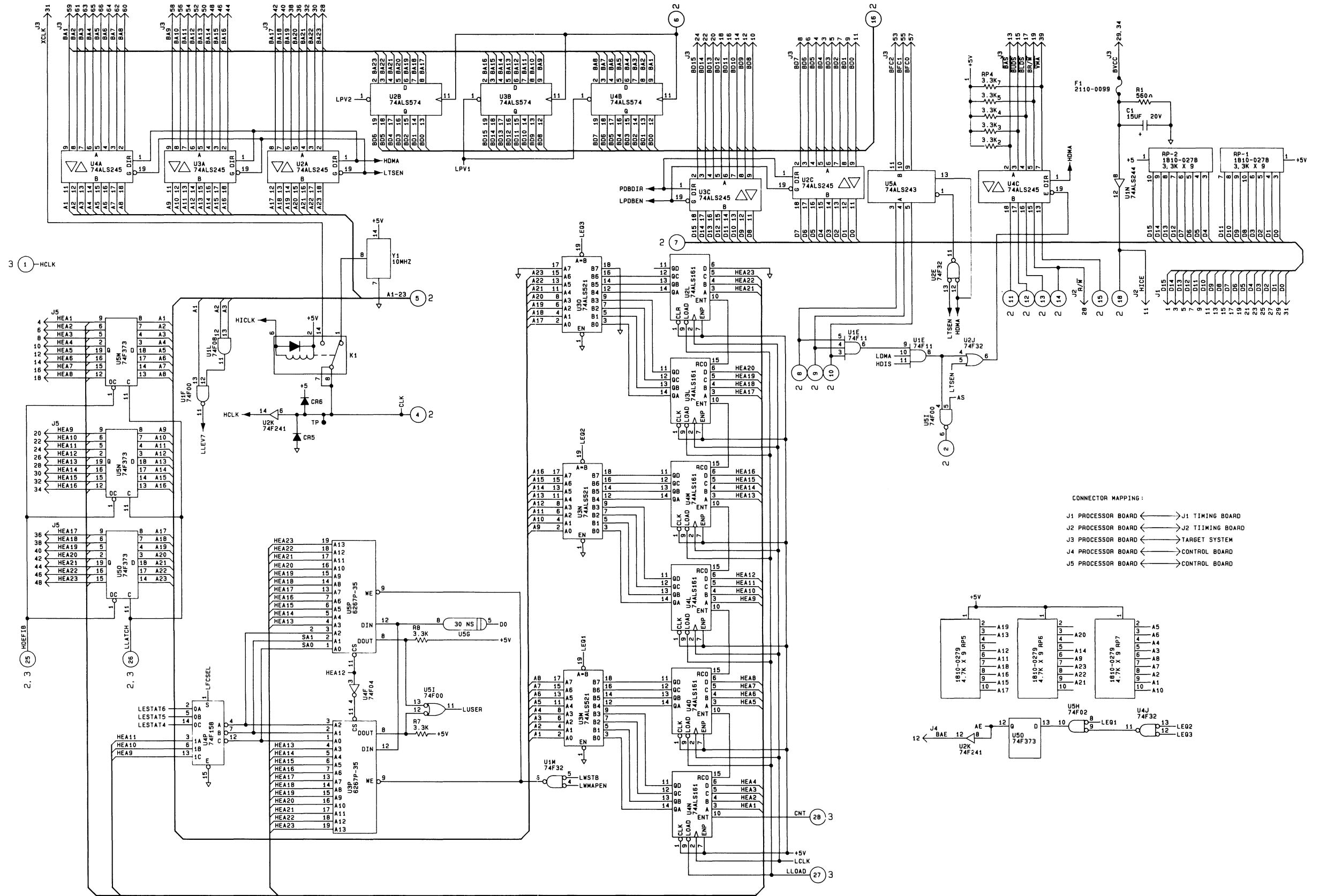


Figure G8-1.
 68000 Emulator Pod Processor Board Schematic (Sheet 1 of 3)
 G8-35

NOTES

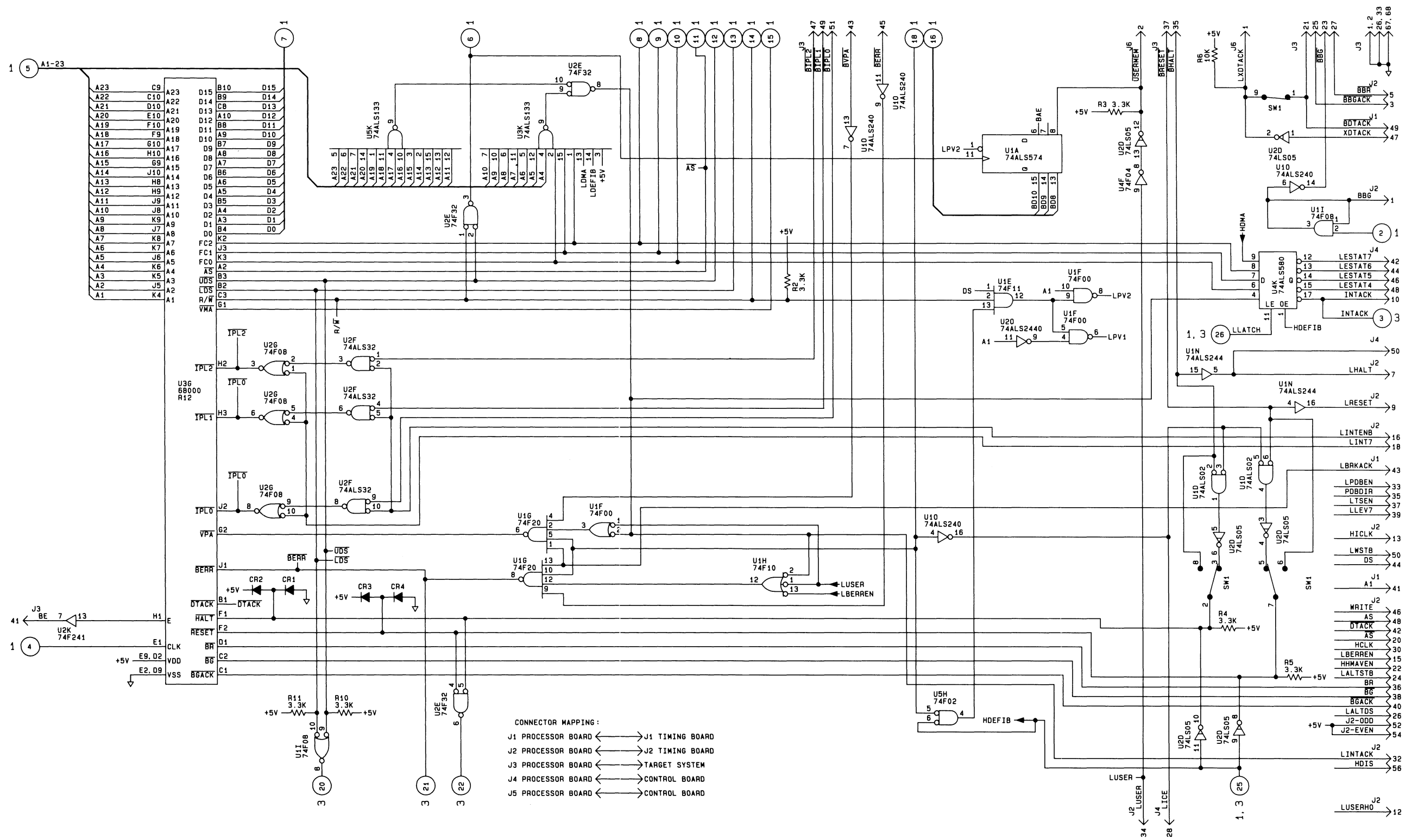
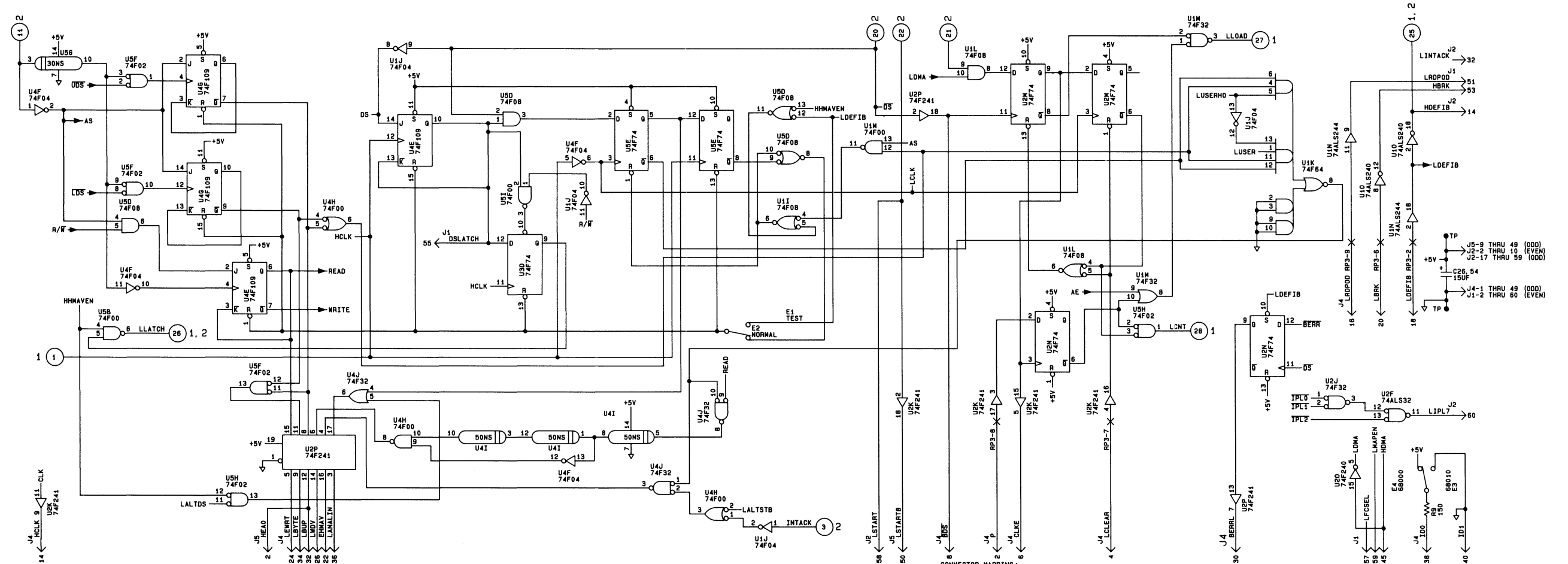


Figure G8-1.
 68000 Emulator Pod Processor Board Schematic (Sheet 2 of 3)
 G8-37

NOTES



CONNECTOR MAPPING:

- J1 PROCESSOR BOARD ← J1 TIMING BOARD
- J2 PROCESSOR BOARD ← J2 TIMING BOARD
- J3 PROCESSOR BOARD ← TARGET SYSTEM
- J4 PROCESSOR BOARD ← CONTROL BOARD
- J5 PROCESSOR BOARD ← CONTROL BOARD

Figure G8-1.
68000 Emulator Pod Processor Board Schematic (Sheet 3 of 3)
G8-39

NOTES

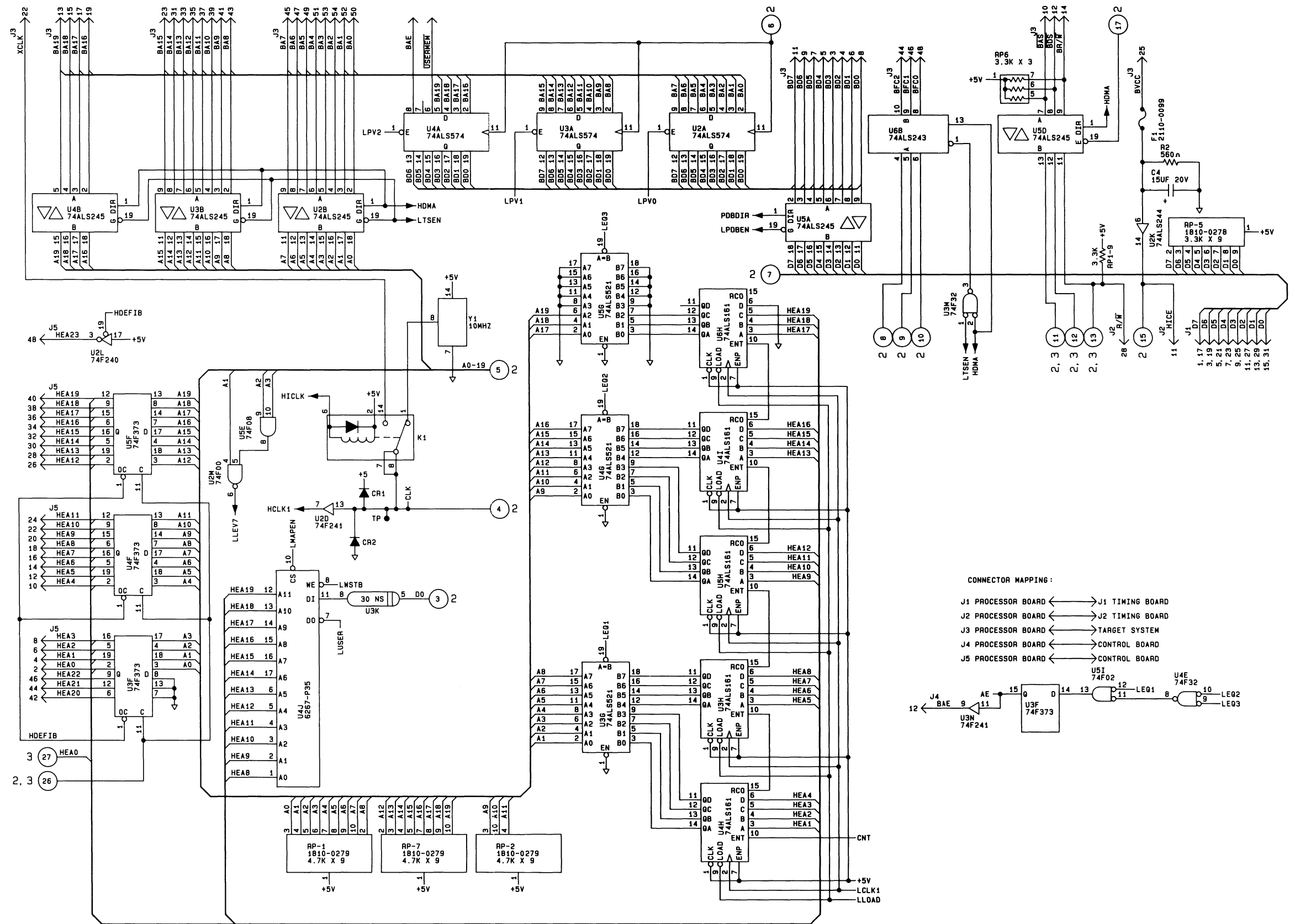


Figure G8-2.
68008 Emulator Pod Processor Board Schematic (Sheet 1 of 3)
G8-41

NOTES

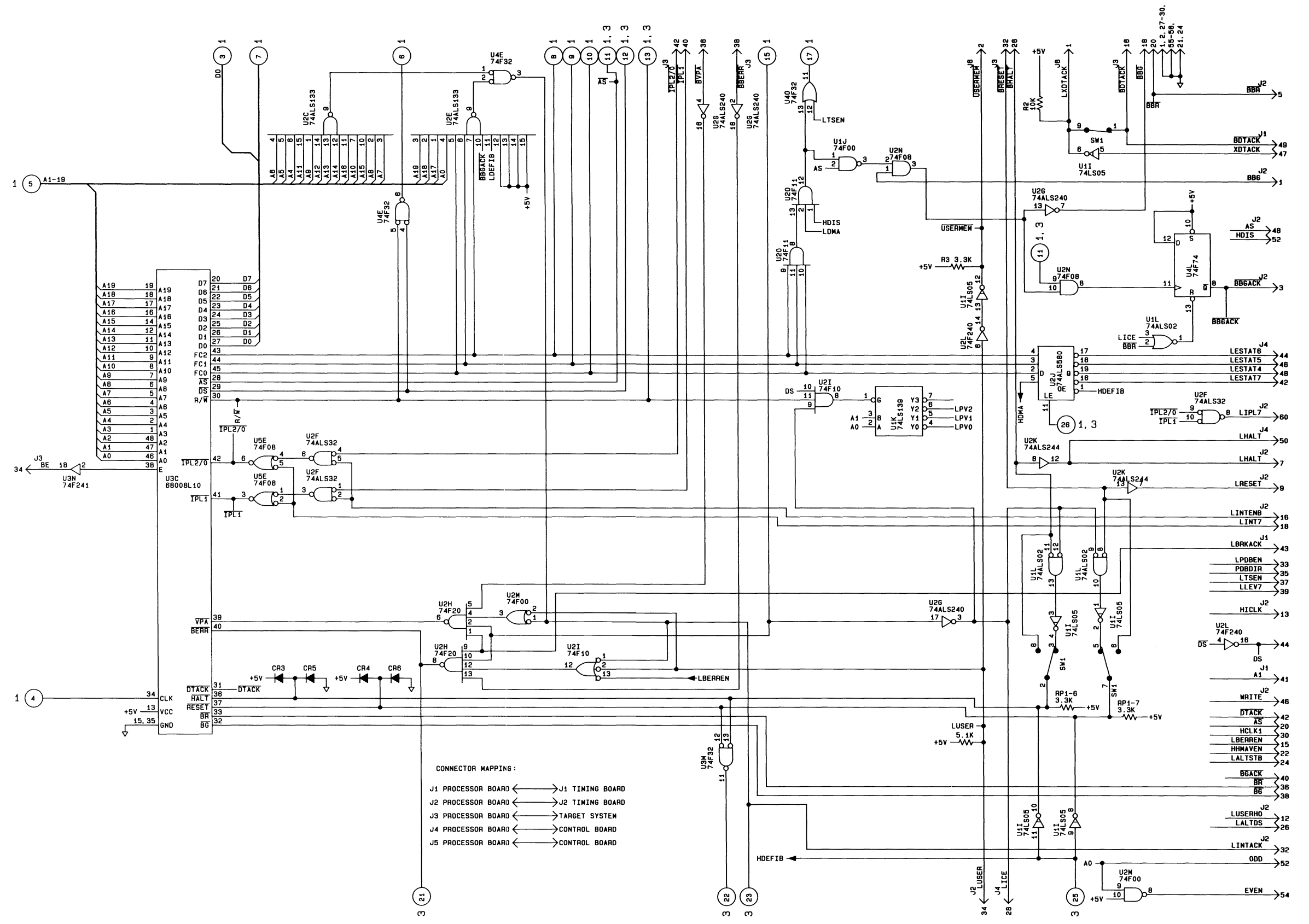


Figure G8-2.
 68008 Emulator Pod Processor Board Schematic (Sheet 2 of 3)
 G8-43

NOTES

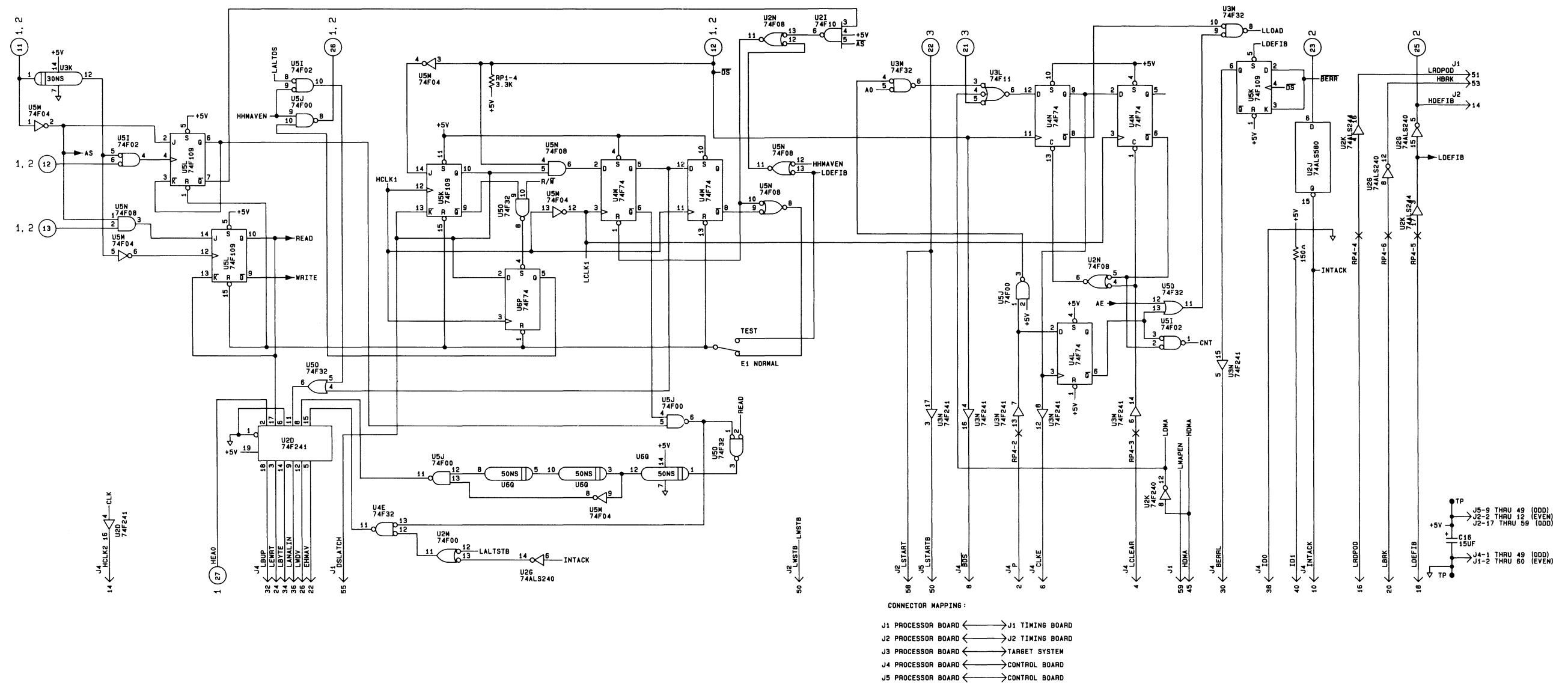


Figure G8-2.
 68008 Emulator Pod Processor Board Schematic (Sheet 3 of 3)
 G8-45

NOTES

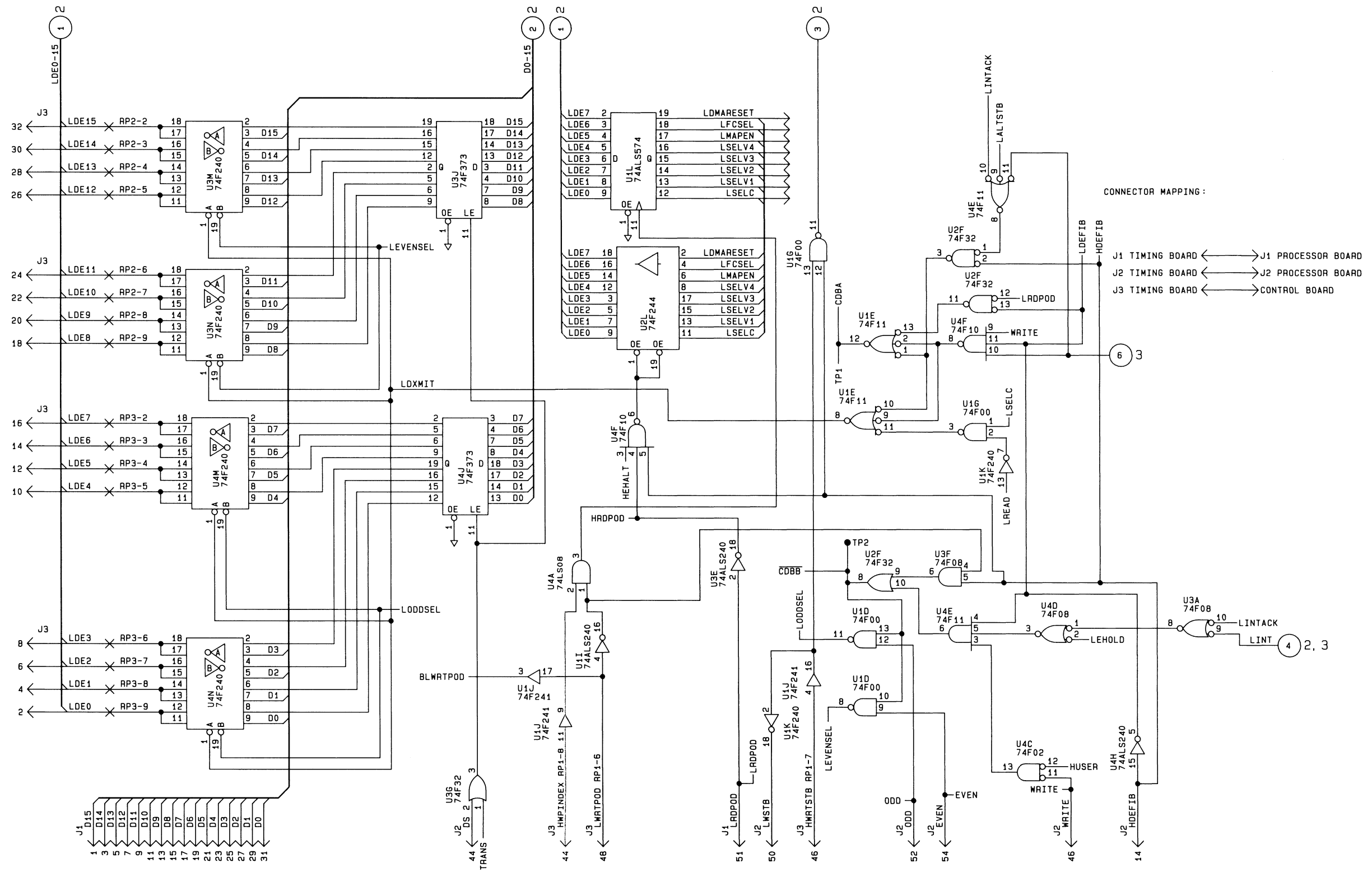


Figure G8-3.
 Emulator Pod Timing Board Schematic (Sheet 1 of 4)
 G8-47

NOTES

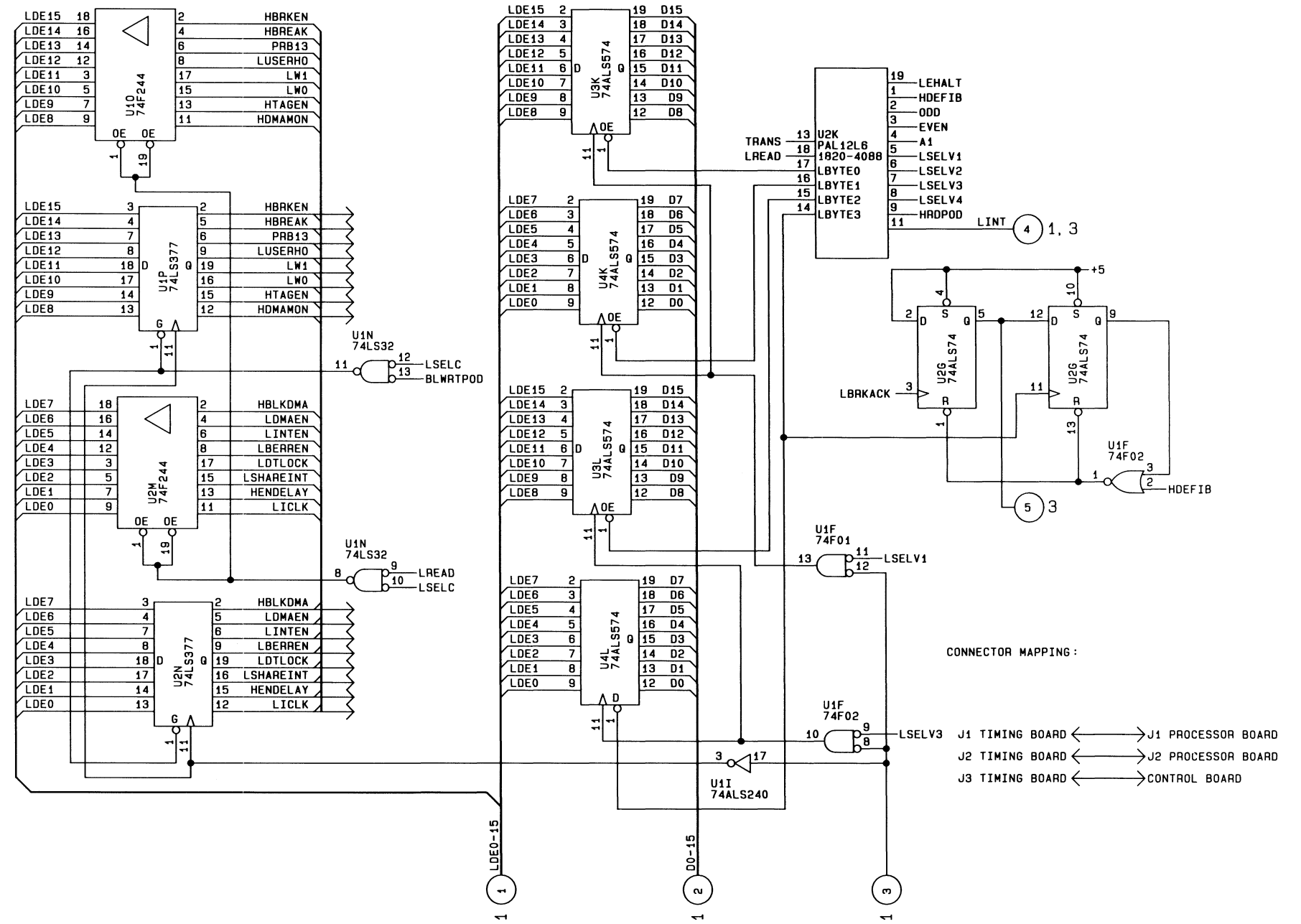


Figure G8-3.
 Emulator Pod Timing Board Schematic (Sheet 2 of 4)
 G8-49

NOTES

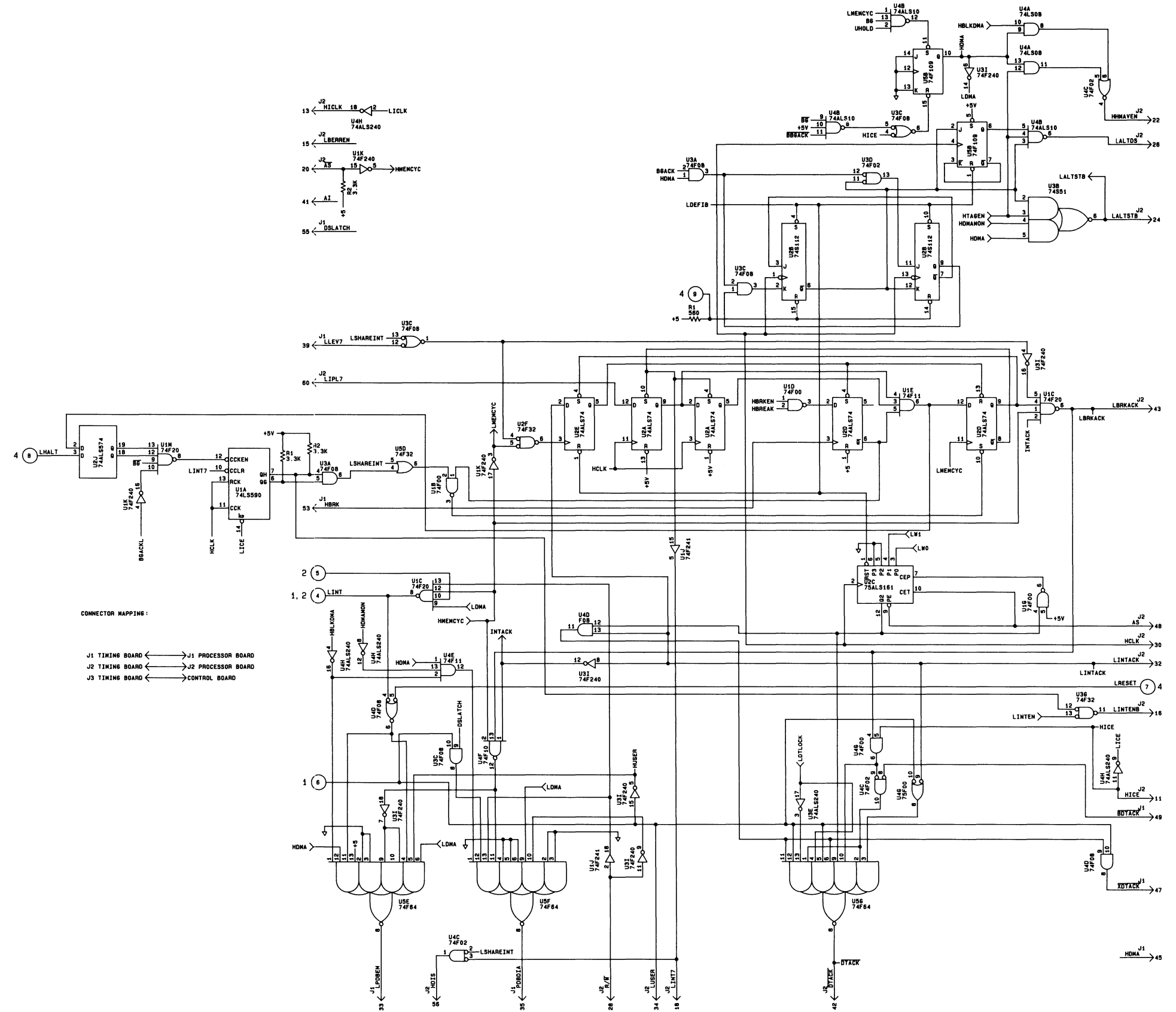


Figure G8-3.
Emulator Pod Timing Board Schematic (Sheet 3 of 4)
G8-51

NOTES

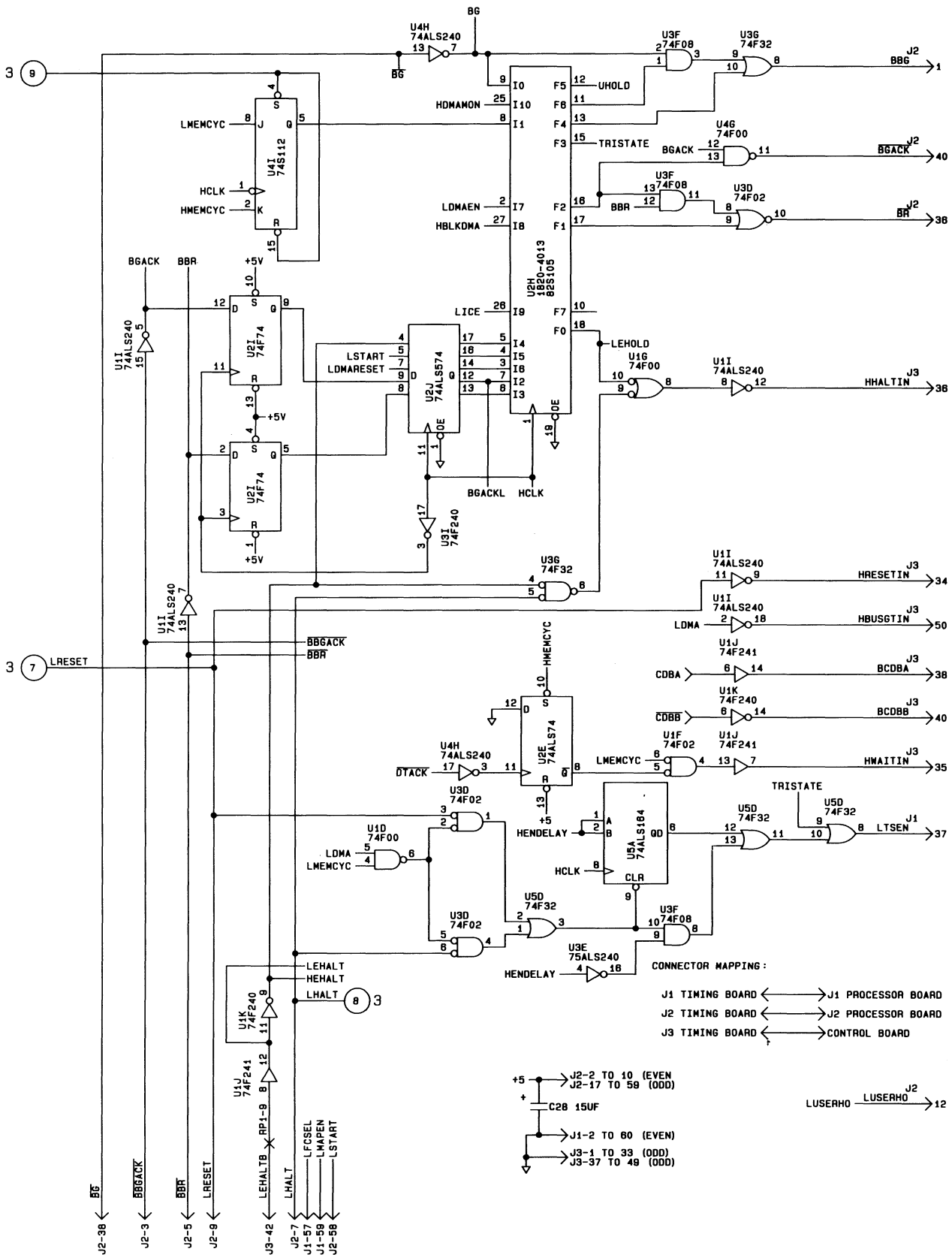


Figure G8-3.
 Emulator Pod Timing Board Schematic (Sheet 4 of 4)
 G8-53

NOTES

Appendix G

INSTALLATION AND SERVICE INFORMATION

SERVICE INFORMATION INDEX

a

abbreviations	G4-1
adjustments	G3-1
analysis data buffers	G2-11
analysis status buffer	G2-11
analysis stimulus test	FGG2-6
analysis tests	FGG2-7
awaiting command status	FGG2-1

b

backing up your software	G1-24
bus arbitration timing diagram - active bus case (68000)	FGG6-4
bus arbitration timing diagram - active bus case (68008)	FGG6-11
bus arbitration timing diagram - idle bus case (68000)	FGG6-3
bus arbitration timing diagram - idle bus case (68008)	FGG6-10
bus arbitration timing diagram -multiple bus requests (68000)	FGG6-5
bus arbitration timing diagram -multiple bus requests (68008; 52-pin version only)	FGG6-12
board installation sequence	G1-16
break address register (U3K, U4K, U3L, U4L)	G8-11
break method	G8-12

c

characteristics, electrical and environmental	G6-1
characteristics, performance	G6-2
clock selection/clock generation circuit	G2-14
configuration of boards, with state or timing analyzers	G1-7
configuration of the boards in the station	G1-6
connector compatibility, target system microprocessor	G1-21
control card replaceable parts	TG4-2

d

data bus transmission and control	G8-7
data transceivers, development station	G2-11
diagrams, schematic	G8-1
direct mail order system	G4-2
DMA arbitration	G8-14
DMA circuitry	G8-14
DMA tagging	G8-14

SERVICE INFORMATION INDEX (Cont'd)

e

electrical and environmental characteristics	G6-1
electrical specifications - read and write cycles (68000)	TG6-3
electrical specifications - read and write cycles (68008)	TG6-4
emulation control board connections	FGG1-5
emulation control card	G2-10
emulation control card, removing	G1-31
emulation halt (EHALT)	G8-14
emulation memory controller	G2-10
emulation probe interface	G2-14
emulation system block diagram	FGG2-8
emulation subsystem overview	G2-10
emulator block diagrams	G2-10
emulator control card	G2-10
emulator control card block diagram	FGG2-9
emulator control card parts locator	FGG4-1
emulator electrical and environmental characteristics	G6-1
emulator pod	G2-10,G2-13
emulator pod block diagram	FGG2-10
emulator pod bottom view	FGG5-3
emulator pod disassembly	G5-1
emulator pod illustrated parts breakdown	FGG4-2
emulator pod reassembly	G5-5
emulator pod replaceable parts	TG4-3
emulator pod side view	FGG5-1
emulator pod top view	FGG5-2
emulator troubleshooting flow chart	FGG2-11
exchange assemblies	G4-1
external hardware features of the development stations	G1-4

f

function drivers	G2-16
function latch	G2-16

g

generation of LDTACK	G8-13
generation of XDTACK	G8-13
ground bar clamp installation (HP 64100A)	FGG1-6
ground bar clamp installation (HP 64110A)	FGG1-8

h

hex comparator	G2-16
hold/hlda controller	G2-16

SERVICE INFORMATION INDEX (Cont'd)

i

I/O bus configuration display	FGG1-12
inspecting the equipment	G1-4
installation details HP 64110	FGG1-7
installation instructions	G1-8,G1-15
installation sequence	G1-8
installing emulation hardware into a 64100 logic development station	G1-6
installing emulation hardware into a 64110 logic development station	G1-15
installing RFI ground bracket	G1-9
installing the emulation probe into a DIP socket	FGG1-10
installing the emulation probe into a PGA socket	FGG1-11
installing the emulation probe into the target system	G1-20
installing your emulation hardware and software	G1-1
instruction fetch latches	G2-12
internal analysis	G2-10

j

jam address latch	G2-17
-------------------------	-------

k

key features, HP 64100	FGG1-1
key features, HP 64110	FGG1-2

l

loading emulation system software	G1-24
loading software in clustered stations configuration	G1-24

m

mapper RAM	G2-10
MC68008 to M6800 peripheral timing diagram - best case	FGG6-8
MC68008 to M6800 peripheral timing diagram - worst case	FGG6-9
memory data bus transceivers	G2-11
memory, emulation, and intermodule bus cabling, HP 64110	FGG1-9
memory strobe generator	G2-11
mnemonics	G8-15,TG8-1

n

NMI/break control circuit	G2-17
---------------------------------	-------

SERVICE INFORMATION INDEX (Cont'd)

o

operating emulation system in stand-alone configuration	G1-30
operating environment	G6-3
operating verification, performing	G1-29
operation, theory of	G2-8, G8-1
option_test card slot listing	FGG2-2
option test display	FGG1-13
ordering information	G4-2

p

performance characteristics	G6-3
performance verification	FGG2-3
performance verification (PV) softkey descriptions	G2-2
performance verification and troubleshooting	G2-1
pod bus and board interconnect cables	FGG5-5
pod functionality test	FGG2-5
pod index register (U1L)	G8-9
pod control register (U2N, U1P)	G8-10
pod registers	G8-9
pod register tests (static)	FGG2-4
pod to control board transceiver controls circuit	G2-16
power supply requirements	G6-1
preinstallation inspection	G1-4
processor board A1, 68000	G8-1
processor board A1, 68008	G8-4
program counter image	G2-12
PV/function latch control	G2-13

r

radio frequency interference	G7-1
read cycle timing diagram (68000)	FGG6-1
read cycle timing diagram (68008)	FGG6-6
reducing r.f. noise	G7-1
reference designators and abbreviations	TG4-1
repackaging for shipment	G1-32
removal and replacement procedures	G5-1
removing emulation software from the operating system	G1-29
removing emulator control board	G1-31
replacable parts	G4-1
required, recommended, and additional optional equipment,	G1-4
RFI ground bracket assembly parts	FGG1-3
RFI ground bracket assembly parts list	TG1-1
RFI ground bracket installation (64100A)	FGG1-4
RFI ground bracket, installation	G1-9

SERVICE INFORMATION INDEX (Cont'd)

S

schematic diagrams	G8-1
schematic diagram, emulator pod processor board, 68000	FGG8-1
schematic diagram, emulator pod processor board, 68008	FGG8-2
schematic diagram, emulator pod timing board	FGG8-3
stand-alone configuration, operating in	G1-30
status decoder/latch circuitry	G2-16
storage and shipment	G6-3

T

tagging, DMA	G8-14
theory of operation	G2-8,G8-1
timing board A2	G8-7
timing comparison - MC68000R12 vs 12.5 Mhz emulator	TG6-1
timing comparison - MC68000L10 vs 10.0 Mhz emulator	TG6-2
tristating the bus	G8-15
troubleshooting	G2-8
troubleshooting flow chart	G2-17
turning on the development station	G1-23

U

unpacking the equipment	G1-4
upper address latch	G2-17
user interface	G6-1
user transceiver control logic	G2-16

W

wait detection circuitry	G8-13
write cycle timing diagram (68000)	FGG6-2
write cycle timing diagram (68008)	FGG6-7

NOTES

INDEX

a

<ABSFILE>	A-1
absolute file	3-10, 3-12
accepting keystrokes from the keyboard	12-1
access	
internal processor memory	1-3
internal processor registers	1-3
port	1-5
existing files - opening the file (81H)	13-4
the configuration questions	5-2
address	
conventions	5-2
length	5-2
range information for memory mapping	3-10
range overlays	5-13
<ADDRESS>	A-1
<ADR_LST>	A-1
advancing N records (84H)	13-5
analysis	
(with optional internal analyzer board)	1-7
and interactive commands	7-7
card	5-7
examples	6-19
modes	7-10
and additional optional equipment	2-4
answering the emulation configuration questions	3-15, 5-7
assembling	
and linking the program modules	3-9
modules	3-9, 4-1
assigning your userid	3-2
asynchronous mode	14-5
asynchronous mode instruction format (figure)	14-5
automatic selection of record 1,2,3...etc.	13-5
available block numbers	5-13
available memory for simulated I/O	5-19

b

backing up your software	2-24
backup N records (85H)	13-5
base numbers	5-2
beginning and ending emulation	6-6
block size	5-12
board installation sequence	2-15
break	1-8
processor on write to ROM?	5-10

break (cont'd)	
syntax	7-16
into the monitor	1-10
generation.	1-7
(NMI)	5-8
building system command files	3-16
bus_cycle_data	
mode	7-10
trace page 1 (figure)	7-11
trace page 2 (figure)	7-12
bytes available	2-7

c

changing the file name (8AH)	13-6
clock source selection.	1-7
closing	
the printer file	10-1
simulated I/O keyboard file	12-2
the display file	11-1, 11-4
the file (82H)	13-4
the keyboard file (81H)	12-3
the keyboard interface	12-1
the printer file (81H)	10-3
the read buffer (8BH)	14-17
the RS-232 file (81H)	14-17
the write buffer (88H)	14-12
<CMDFILE>	A-1
code access, instruction-by-instruction	1-3
command file	
file name?	5-26
file, emulation configuration	5-26
file, emulation configuration demo	3-15
mode instruction format (figure)	14-4
to 8251 USART (83H)	14-7
word codes (figure)	12-6
configuration	
of boards, with state or timing analyzers	2-7
of the boards in the station	2-6, 2-15
of the memory boards	2-7, 2-15
switches (figure)	5-27
switches	5-27
memory board jumpers	2-7
a flexible disc for stand-alone operation	2-25
for interactive measurements	5-24
simulated I/O	5-19
the emulator pod	5-21
/reconfiguring the emulation pod	5-3
controlling flow of data and code	5-11
copy the modules onto new discs	2-26
copying the 680XX programs to your userid	3-6

creating a new file (80H)	13-3
current block number values	5-13
customizing the emulation monitor	8-5

d

data area memory requirements	5-13
data transfer	1-5
debugging target system	1-3
default command syntax	5-19
default interactive measurements	5-24
default response to emulation configuration questions	5-7
define disc software configurations	2-26
definition, status word	6-3
deleting blocks	5-19
deleting the file (83H)	13-7
demo	
absolute file	3-10
absolute file	3-17
absolute file	3-19
command file	3-12
emulation configuration command file	3-15
emulation configuration command file	3-16
link command file	3-16
link command file	3-17
listing file	3-10
system command file	3-19
system command file	3-19
demo_cmd	3-9
demo_cmd system command file	3-17
demo_cmd system command file	3-18
demonstration configuration	6-7
development station display	11-4
development system tools (figure)	1-6
disc	
drive interface diagram (figure)	13-1
file CA address	13-2
file control addresses (CA)	13-1
file control codes (table)	13-8
file example program	13-15
file I/O codes	13-2
file simulated I/O	5-1, 13-1
file type numbers and names (figure)	13-2
<DISC#>	A-2
display	
CA address	11-2
control address	5-20, 11-1
control codes (table)	11-5
example program	11-7
interface diagram (figure)	11-1
memory - absolute byte:	6-13

display (cont'd)	
memory - blocked byte:	6-11
memory - mnemonic:	6-12
memory - real:	6-14
memory blocked byte offset:	6-15
registers:	6-9
simulated I/O	11-1
techniques (figure)	11-4
trace absolute:	6-21
trace status binary:	6-22
trace status mnemonic:	6-23
display/ keyboard simulated I/O program (figure)	11-10
display/list	
global_symbols syntax	7-19
io_port syntax	7-20
loc_sym syntax	7-22
memory syntax	7-23
registers syntax	7-26
software_breakpoints syntax	7-28
syntax	7-17
trace syntax	7-29
dividing the processor address space	5-11
DMA controls	5-8
DMA cycles using execution_mode	F-3
during emulation monitor program control	1-9
during normal flow of the program	1-8

e

editing log command files	3-17
editing modules	3-17
effects of execution_data mode	F-1
electrical transparency	1-10
emul cntrl bd-interface between mem pod and mem/analy busses	1-3
emulate	
command	5-2
command syntax	5-3
softkey	3-3, 3-16
softkey label	3-12
syntax	5-6, 7-32
emulation	
configuration command file	5-26
configuration questions	3-12, 3-15
control board	1-3, 1-8
control board connections (figure)	2-12
examples	6-1, 6-7
map for test program (figure)	5-12
memory	1-5, 5-13
memory and target system memory	6-5
memory configuration	1-5
memory load operations	5-13

emulation (cont'd)	
memory overlays	5-16
monitor	8-2
monitor flowchart (figure)	8-11
monitor flowchart	8-10
monitor memory requirements (68000)	4-4, 8-8
monitor memory requirements (68008)	4-5, 8-9
monitor memory requirements	5-13
monitor overview	8-1
monitor source program	8-10
probe pod	1-3
processor	1-3
subsystem hardware	1-2
system	1-1
system components	3-3
system functional block diagram (figure)	1-4
system hardware	1-2, 2-4
system manual	1-2
system softkeys	3-4
system software	1-2
emulator	
monitor	1-8
pod configuration default conditions	5-21
function of	1-3, 1-9
steps to using	1-10
emulator/analysis independent from HP 64000 operating system	1-3
enable	
bus error on emulation memory accesses?	5-22
DMA transfers to emulation memory ?	5-22
emulator DMA transfers?	5-22
emulator processor interrupts?	5-22
emulator use of INT7 ?	5-23
interrupt level 7 sharing ?	5-23
tracing of DMA memory transfers ?	5-22
tracing of DMA tags ?	5-23
tristate delay on halts ? no (yes)	5-23
end syntax	7-33
ending the "log" command file	3-16
ending the mapping session	5-19
entering memory blocks	5-15
error codes, simulated I/O	E-1
error messages (table)	D-3
error messages	D-1
example	
link programs (figure)	4-3
link_sym listing (figure)	4-11
memory map (figure)	4-12
one word branch-around code (figure)	F-2
program	5-7
program module	4-1
of using the emulator	6-1
exception vector table (commented) (figure)	3-7
exception vector table (un-commented) (figure)	3-8
execute syntax	7-34

execution cycle data page 1 (figure)	7-13
execution data trace page 2 (figure)	7-14
execution_data mode	7-10
external	
analysis systems	1-2
clock	5-8
hardware features of the development stations	2-4
timing systems	1-2

f

<FILE NAME>	A-2
file names used in this manual	3-5
<FILE>	A-1
files you will create	3-5
format a new flexible disc	2-25
from the operating system	2-29
functional description	1-3
functional transparency	1-9

g

gaining access to the emulator	3-12
general information	1-1
general user information	1-11
getting started	3-1
global and local symbols display.	1-7
<GLOBAL SYMBOL>	A-3
glossary of softkey labels	C-1
guarded memory	5-13
guarded memory access	5-2
guidelines for in-circuit emulation	6-5
guidelines for out-of-circuit emulation	6-5

h

halt syntax	7-35
hardware	
breakpoints	1-5
debug	1-5
integration	1-5
mapper	1-5
host processor	1-3
how are these tasks implemented?	1-5
HP 64000 logic development system simplified block diagram	9-3

HP 64100 key features (figure) 2-2
HP 64110 installation details (figure) 2-17
HP 64110 key features (figure) 2-3
HP 64110 memory, emulation, and intermodule bus cabling(figure) 2-19

i

I/O bus configuration display (figure) 2-23
I/O ports display/modification. 1-7
illegal conditions 5-2
in stand-alone configuration 2-30
in-circuit emulation 6-5
initializing the 8251 USART (82H) 14-3
initiating a system "log" command file 3-9
inspecting the equipment 2-4
installation
 instructions 2-8, 2-15
 overview 2-1
 sequence 2-8
installing
 emulation system hardware 2-6, 2-15
 into a DIP socket (figure) 2-21
 into a PGA socket (figure) 2-22
 into an HP 64100 logic development station 2-6
 into an HP 64110 logic development station 2-15
 the emulation probe microprocessor (figure) 2-21, 2-22
 your emulation software 2-1
instruction-by-instruction code access 1-3
integrating program modules 1-3
interactive measurements 1-7, 9-5
interactive measurements, configuring for 5-24
interactivity with other HP 64000 system modules 1-1
interlock emulation memory DTACK with user DTACK? 5-21
internal
 analysis board function 1-5, 1-9
 analysis module 5-24
 clock 5-8
 processor memory 1-8
 processor registers 1-8
 reset (ir) 14-4
 resource display/modification. 1-7
introduction 6-1
jumpers, memory board 2-7

k

keyboard	
CA address	12-2
control address (CA)	5-20, 12-1
control codes (figure)	12-4
interface diagram (figure)	12-1
interface sequence (figure)	12-7
simulated I/O sample program	12-1

l

leading zeros	5-16
leading zeros in address identifier	5-2
line & column	11-3
linker absolute file	3-12
linker output listing (figure)	3-11
linker questions	3-10
linking	4-1
linking modules	3-10
linking the emulation monitor	8-9
load syntax	7-36
loading emulation memory	3-16
loading emulation system software	2-24
loading software in clustered stations configuration	2-24
<LOCAL SYMBOL>	A-3
log commands, reason for using	3-9
logging commands to a command file	3-17
looking at your sample program	3-9

m

manual contents guide	x
mapped block numbers	5-13
mapper blocks, syntax for entering	5-16
mapping display softkey labels	5-14
mapping memory	5-11
maximum clock rate	5-8
meas_sys softkey	3-3, 3-16
meas_sys softkey label	3-12
measurement_system	
/emulate command syntax	5-3
(using options continue) syntax	5-4
(with options continue) syntax	7-37
(without options continue) syntax	7-37
(without using options continue) syntax	5-4
command	5-2

measurement_system (cont'd)	
command syntax	5-3
memory	
accesses--display, list, load, modify, and store.	1-8
available for simulated I/O	5-19
board capacity	1-5
board functions	1-5
board interface	1-5
board jumpers	2-7
board model number mixes	2-8
board option	1-5
board space	2-7
boards	1-5
characterization.	1-7
configuration modification	5-10
configuration review	5-10
contents - hexadecimal and ascii (figure)	7-4
default	5-12
display/modification.	1-5
map	5-7
map definition	5-12
map display (figure)	5-14
map display entries	5-14
map for test (figure)	4-7
mapping example	5-7
resource identification	1-8
resources during emulation	6-4
space	1-5
microprocessor clock source?	5-8
microprocessor replacement probe	1-3
modify	
access_size_to syntax	7-40
analysis_mode_to	7-41
and display memory - byte:	6-16
configuration syntax	7-42
interactive measurement specifications?	5-24
internal processor registers	1-3
io_port syntax	7-43
memory configuration?	5-10
memory syntax	7-44
register syntax	7-46
simulated disc files?	5-20
simulated I/O?	5-20
software_breakpoints syntax	7-47
syntax	7-39
a current configuration	5-10
the monitor to use software breakpoints	8-4
the monitor to use the vector address table	3-6
module file	3-10
<MODULE>	A-3
Mon_68000 relocatable file - record #1 (figure)	4-4, 8-8
Mon_68008 relocatable file - record #1 (figure)	4-5, 8-9
Mon_680XX monitor program	3-10

monitor	
level softkeys	3-4
program	1-9, 3-6
program construction	1-9
program description	1-8
program memory location	1-9
program, function of	1-8
multi-module emulation and analysis	9-5
multi-module systems	3-3, 3-17
multi-module systems, interactive measurements	5-24
multi-processor designs	1-7, 9-5

n

NMI	8-2
nonreal-time mode	1-8
number	
of blocks currently mapped	5-13
of bytes available	2-7
of emulation memory blocks available	5-13
of mapper blocks vs. available memory (table)	5-15
of significant address bits?	5-10
<NUMBER>	A-3

o

opening	
the printer file	10-1
the display file	11-1
printer interface	10-2
the display file (80H)	11-2
the keyboard file (80H)	12-2
the keyboard interface	12-1
the printer file (80H)	10-2
the RS-232 file (80H)	14-2
operating the emulation system	2-30
operational overview	3-1
option test display (figure)	2-30
organizing your program modules	3-6
out-of-circuit emulation	5-1, 6-4
output available (00)-HP 64000 response	12-3
overlay addresses	5-13
overlay errors	5-18

p

partitioning the processor address space	5-12
performing operation verification	2-29
physical address	5-2
physical description	1-1
position to record N (86H)	13-5
pre-installation inspection	2-4
preparing program modules for emulation	4-1
printer	
CA address	10-2
control address (CA)	5-20, 10-1
control codes (table)	10-4
example program	10-5
interface (figure)	10-1
simulated I/O	10-1
procedure delay;	11-8
processor reset	6-2
processor status messages	6-2
program	
disc_1	13-16
disc_1 (figure)	13-16
loading and execution.	1-5
module	3-6
PRINT_SIO (figure)	10-5
stepping.	1-7

r

read in process (82H) - HP 64000 response	12-2
read record (8AH)	
reading	
a single byte (85H)	14-13
from the 8251 USART	14-13
RS-232 record interface sequence (figure)	14-15
the record (87H)	13-6
real-time	1-8
real-time mode	1-8
<REAL>	A-2
reconfigure emulator pod?	5-21
reference manual updates	1-11
register accesses--display, list, and modify.	1-8
register format and names	B-1
relocatable program area memory requirements	5-13
removing emulation software	2-29
removing the emulator control board	2-31
repackaging for shipment	2-32
replace target system processor	1-3
required equipment, recommended additional equipment,	2-4
reset	1-3

reset syntax	7-48
resource mapping	1-7
restrict to real-time runs	1-10, 5-8
rewind to record one (88H)	13-6
roll to/write line 18 (82H)	11-2
RS-232	
control address (CA)	14-1
control codes (table)	14-18
interface diagram (figure)	14-2
simulated I/O	14-1
control address?	5-20
run	
emulation command file trace 1 (figure)	3-23
emulation command file trace 2 (figure)	3-24
syntax	7-49
/stop controls	1-5
run_until	1-5
runs not restricted to real time	5-8

S

safety considerations	1-1
sample modified IO_FUNCTION (figure)	8-8
sample overlay mapping #1 (figure)	5-17
sample overlay mapping #2 (figure)	5-18
scrolling	11-2
see real-time tracing of processor operations	6-19
seizing control of the emulation processor	1-9
selecting	
a record	13-5
nonreal-time runs	5-8
real-time runs	5-8
real-time/nonreal-time runs	5-8
the card slot	5-7
the clock	5-8
the memory configuration	5-10
the starting line/column (83H)	11-3
/modifying a memory configuration	5-9
setting the default	5-18
simulated I/O	
devices minimum memory space	5-19
devices, maximum open at one time	5-19
error codes - general definitions (table)	E-1
error codes	E-1
memory availability	5-19
questions	5-20
single-module systems	3-3, 3-17
size of the memory blocks	5-13
slot number	5-7
of analysis card?	5-7
of memory controller card?	5-7
softkey labels and mnemonic status definitions (figure)	6-4

softkey	
labels, glossary of	C-1
prompt messages (table)	D-5
prompt messages	D-1
templates	3-4
softkeys	3-4
software	
debug	1-5
integration	1-5
materials subscription	1-11
problem reporting	1-11
release bulletins	1-11
status bulletins	1-11
updates	1-11
specify syntax	7-51
starting address of a block boundary	5-16
<STATE>	A-2
static memory	1-5
status	
definition	6-3
from 8251 (84H)	14-8
line messages	D-1
messages (table)	D-1
register format (figure)	B-2
word definition	6-4
step registers <#STEPS> from <ADDRESS>:	6-10
step syntax	7-52
steps to using the emulator	1-10
stop_trace syntax	7-53
stopping processor for real-time runs	5-8
store syntax	7-54
<STRING>	A-3
symbol accesses--display and list	1-8
synchronous mode instruction format (figure)	14-6
synchronous mode/double sync character	14-7
synchronous mode/single sync character	14-6
syntactical variable definitions	A-1
syntax for entering mapper blocks	5-16
syntax for the default command	5-19
system memory capacity	5-14

t

target	
memory	5-13
memory display operations	5-13
memory load operations	5-13
system memory	1-5
system microprocessor connector compatibility	2-21
system program interrupt	8-2
the break function and the emulation monitor	8-2
the relationship of linker mapping and emulation memory mapping	4-2

timing transparency	1-9
to break into emulation monitor and see registers	3-21
to see real-time tracing of processor activity	3-20
trace	
after 002000H (figure)	3-21
after <ADDRESS>:	6-20
data	1-5
memory display (figure)	7-7
syntax	7-56
transferring control to a monitor program	1-9
transparency	1-9
trap number for software break (0..0FH)?	5-23
turning on the development station	2-23
typing a system command file	3-19
typing an emulation command file	3-20
typing the commands into a command file	3-18

u

understanding the examples	3-2
understanding the examples used in this manual	x
unpacking the equipment	2-4
update read buffer (8CH)	14-13
update write buffer (89H)	14-9
updating read/write buffers (8DH)	14-17
<USERID>	A-2
using	
a buffer to read multiple bytes	
a buffer to write multiple bytes	
analysis commands	6-18
the "trace function" of the 680XX	F-2
the 680XX emulation monitor program	4-4
the assembler	4-2
the command files	3-23
the emulator	3-20
the emulator--examples	6-1
this manual	x
unimplemented instructions	F-3
utility keys used for transportation (from "emulate") (figure)	3-13
utility keys used for transportation (from "meas_sys") (figure)	3-14

v

<VALUE>	A-3
---------------	-----

w

wait syntax	7-63
what	
can software materials subscription do for you?	1-1
does an emulator allow you to do?	1-1, 1-5
does the emulator do to your microprocessor system?	1-1
effect will the emulator have on your program?	1-8
is an emulation system?	1-1
is happening while your program is running?	1-1
is your role in the emulation process?	1-1
where to locate the monitor	4-12
why so many files have the same name	4-11
will the emulator system run interactively	1-7
will using an emulator have an effect on your program?	1-1
with other HP 64000 system modules?	1-7
write	
record (87H)	
to printer	10-1
to the display	11-1
write-to-printer code	10-2
writing	
a record (89H)	13-3
a single byte (86H)	14-9
additional records	13-4
first record	13-4
from the starting line/column (84H)	11-3
RS-232 record interface sequence (figure)	14-10
to the 8251 USART	14-9
to the printer (82H)	10-2

y

your part in the emulation process	1-10
your program module	3-5

6

68000/68008 Emulation Systems (figure)	1-0
--	-----

8

8251 Initialization Formats (figure)	14-3
8251 Status Word Format (figure)	14-8

NOTES

SALES & SUPPORT OFFICES

Arranged alphabetically by country



Product Line Sales/Support Key

Key Product Line

A Analytical

CM Components

C Computer Systems Sales only

CH Computer Systems Hardware Sales and Services

CS Computer Systems Software Sales and Services

E Electronic Instruments & Measurement Systems

M Medical Products

P Personal Computation Products

* Sales only for specific product line

** Support only for specific product line

IMPORTANT: These symbols designate general product line capability. They do not insure sales or support availability for all products within a line, at all locations. Contact your local sales office for information regarding locations where HP support is available for specific products.

HP distributors are printed in italics.

HEADQUARTERS OFFICES

If there is no sales office listed for your area, contact one of these headquarters offices.

AFRICA AND MIDDLE EAST

Hewlett-Packard S.A.
Mediterranean and Middle East
Operations

Atrina Centre
32 Kifissias Ave.
Paradissos-Amarousion, **ATHENS**
Greece

Tel: 682 88 11
Telex: 21-6588 HPAT GR
Cable: HEWPACKSA Athens

NORTH/CENTRAL AFRICA

Hewlett-Packard S.A.
7, Rue du Bois-du-Lan
CH-1217 **MEYRIN** 2, Switzerland
Tel: (022) 83 12 12
Telex: 27835 hpse
Cable: HEWPACKSA Geneve

ASIA

Hewlett-Packard Asia Ltd.
47/F, 26 Harbour Rd.,
Wanchai, **HONG KONG**
G.P.O. Box 863, Hong Kong
Tel: 5-8330833
Telex: 76793 HPA HX
Cable: HPASIAL TD

CANADA

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
Telex: 610-492-4246

EASTERN EUROPE

Hewlett-Packard Ges.m.b.h.
Lieblgasse 1
P.O.Box 72
A-1222 **VIENNA**, Austria
Tel: (222) 2365110
Telex: 1 3 4425 HEPA A

NORTHERN EUROPE

Hewlett-Packard S.A.
Uilenstede 475
P.O.Box 999
NL-1180 AZ AMSTELVEEN
The Netherlands
Tel: 20 437771

SOUTH EAST EUROPE

Hewlett-Packard S.A.
World Trade Center
110 Avenue Louis Carol
1215 Cointrin, **GENEVA**, Switzerland
Tel: (022) 98 96 51
Telex: 27225 hpse.

EASTERN USA

Hewlett-Packard Co.
4 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 258-2000

MIDWESTERN USA

Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800

SOUTHERN USA

Hewlett-Packard Co.
2000 South Park Place
P.O. Box 105005
ATLANTA, GA 30348
Tel: (404) 955-1500

WESTERN USA

Hewlett-Packard Co.
3939 Lankershim Blvd.
P.O. Box 3919
LOS ANGELES, CA 91604
Tel: (213) 506-3700

OTHER INTERNATIONAL AREAS

Hewlett-Packard Co.
Intercontinental Headquarters
3495 Deer Creek Road
PALO ALTO, CA 94304
Tel: (415) 857-1501
Telex: 034-8300
Cable: HEWPACK

ANGOLA

Telectra
Empresa TAEcnica de Equipamentos
R. Barbosa Rodrigues, 41-I DT.
Caixa Postal 6487
LUANDA
Tel: 35515,35516
E,P

ARGENTINA

Hewlett-Packard Argentina S.A.
Avenida Santa Fe 2035
Martinez 1640 **BUENOS AIRES**
Tel: 798-5735, 792-1293
Cable: HEWPACKARG
A,E,CH,CS,P

AUSTRALIA

Adelaide, South Australia

Office
Hewlett-Packard Australia Ltd.
153 Greenhill Road
PARKSIDE, S.A. 5063
Tel: 272-5911
Telex: 82536
Cable: HEWPARD Adelaide
A*,CH,CM,CS,E,M,P

Brisbane, Queensland

Office
Hewlett-Packard Australia Ltd.
10 Payne Road
THE GAP, Queensland 4061
Tel: 30-4133
Telex: 42133
Cable: HEWPARD Brisbane
A,CH,CS,CM,E,M,P

Canberra, Australia

Capital Territory Office
Hewlett-Packard Australia Ltd.
121 Wollongong Street
FYSHWICK, A.C.T. 2609
Tel: 80 4244
Telex: 62650
Cable: HEWPARD Canberra
C,CH,CM,CS,E,P

Melbourne, Victoria

Office
Hewlett-Packard Australia Ltd.
31-41 Joseph Street
BLACKBURN, Victoria 3130
Tel: 895-2895
Telex: 31-024
Cable: HEWPARD Melbourne
A,CH,CM,CS,E,M,P

Perth, Western Australia

Office
Hewlett-Packard Australia Ltd.
261 Stirling Highway
CLAREMONT, W.A. 6010
Tel: 383-2188
Telex: 93859
Cable: HEWPARD Perth
A,CH,CM,CS,E,M,P

Sydney, New South

Wales Office

Hewlett-Packard Australia Ltd.
17-23 Talavera Road
P.O. Box 308
NORTH RYDE, N.S.W. 2113
Tel: 888-4444
Telex: 21561
Cable: HEWPARD Sydney
A,CH,CM,CS,E,M,P

AUSTRIA

Hewlett-Packard Ges.m.b.h.
Grottenhofstrasse 94
A-8052 **GRAZ**
Tel: (0316) 291 5 66
Telex: 32375
CH,E

Hewlett-Packard Ges.m.b.h.

Lieblgasse 1
P.O. Box 72
A-1222 **VIENNA**
Tel: (0222) 23 65 11-0
Telex: 134425 HEPA A
A,CH,CM,CS,E,M,P

BAHRAIN

Green Salon
P.O. Box 557
Manama
BAHRAIN
Tel: 255503-255950
Telex: 844 19
P

Wael Pharmacy
P.O. Box 648

BAHRAIN
Tel: 256123
Telex: 8550 WAEL BN
E,M

BELGIUM

Hewlett-Packard Belgium S.A./N.V.
Blvd de la Woluwe, 100
Woluwedal
B-1200 **BRUSSELS**
Tel: (02) 762-32-00
Telex: 23-494 paloben bru
A,CH,CM,CS,E,M,P

BERMUDA

Applied Computer Technologies
Atlantic House Building
Par-La-Ville Road
Hamilton 5
Tel: 295-1616
P

BRAZIL

Hewlett-Packard do Brasil
I.e.C. Ltda.
Alameda Rio Negro, 750
Alphaville
06400 BARUERI SP
Tel: (011) 421.1311
Telex: (011) 33872 HPBR-BR
Cable: HEWPACK Sao Paulo
A,CH,CM,CS,E,M,P



SALES & SUPPORT OFFICES

Arranged alphabetically by country

BRAZIL (Cont'd)

Hewlett-Packard do Brasil
I.e.C. Ltda.
Avenida Epitacio Pessoa, 4664
22471 RIO DE JANEIRO-RJ
Tel: (021) 286.0237
Telex: 021-21905 HPBR-BR
Cable: HEWPACK Rio de Janeiro
A,CH,CM,E,M,P*

Convex/Van Den
Rua Jose Bonifacio
458 Todos Os Santos
CEP 20771
RIO DE JANEIRO, RJ
Tel: 249-7121, 591-4946
Telex: 33487
ANAMED I.C.E.I. Ltda.
Rua Bage, 103
04012 SAO PAULO
Tel: (011) 570-5726
Telex: 021-21905 HPBR-BR
M

CANADA

Alberta

Hewlett-Packard (Canada) Ltd.
3030 3rd Avenue N.E.
CALGARY, Alberta T2A 6T7
Tel: (403) 235-3100
A,CH,CM,E*,M,P*

Hewlett-Packard (Canada) Ltd.
11120-178th Street
EDMONTON, Alberta T5S 1P2
Tel: (403) 486-6666
A,CH,CM,CS,E,M,P

British Columbia

Hewlett-Packard (Canada) Ltd.
10691 Shellbridge Way
RICHMOND,
British Columbia V6X 2W7
Tel: (604) 270-2277
Telex: 610-922-5059
A,CH,CM,CS,E*,M,P*

Hewlett-Packard (Canada) Ltd.
121 - 3350 Douglas Street
VICTORIA, British Columbia V8Z 3L1
Tel: (604) 381-6616
CH,CS

Manitoba

Hewlett-Packard (Canada) Ltd.
1825 Inkster Blvd.
WINNIPEG, Manitoba R3H 0Y1
Tel: (204) 786-6701
A,CH,CM,E,M,P*

New Brunswick

Hewlett-Packard (Canada) Ltd.
37 Shediac Road
MONCTON, New Brunswick E1A 2R6
Tel: (506) 855-2841
CH,CS

Nova Scotia

Hewlett-Packard (Canada) Ltd.
Suite 111
900 Windmill Road
DARTMOUTH, Nova Scotia B2Y 3Z6
Tel: (902) 469-7820
CH,CM,CS,E*,M,P*

Ontario

Hewlett-Packard (Canada) Ltd.
3325 N. Service Rd., Unit 6
BURLINGTON, Ontario P3A 2A3
Tel: (416) 335-8644
CS,M*

Hewlett-Packard (Canada) Ltd.
496 Days Road
KINGSTON, Ontario K7M 5R4
Tel: (613) 384-2088
CH,CS

Hewlett-Packard (Canada) Ltd.
552 Newbold Street
LONDON, Ontario N6E 2S5
Tel: (519) 686-9181
A,CH,CM,E*,M,P*

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
A,CH,CM,CS,E,M,P

Hewlett-Packard (Canada) Ltd.
2670 Queensview Dr.
OTTAWA, Ontario K2B 8K1
Tel: (613) 820-6483
A,CH,CM,CS,E*,MS,P*

Hewlett-Packard (Canada) Ltd.
1855 Lasalle Boulevard
SUDBURY, Ontario, P3A 2A3
Tel: (705) 560-5450
CH

Hewlett-Packard (Canada) Ltd.
220 Yorkland Blvd. Unit #11
WILLOWDALE, Ontario M2J 1R5
Tel: (416) 499-9333
CH

Quebec
Hewlett-Packard (Canada) Ltd.
17500 South Service Road
Trans-Canada Highway
KIRKLAND, Quebec H9J 2M5
Tel: (514) 697-4232
A,CH,CM,CS,E,M,P*

Hewlett-Packard (Canada) Ltd.
1150 Rue Claire Fontaine
QUEBEC CITY, Quebec G1R 5G4
Tel: (418) 648-0726
CH,CS

Hewlett-Packard (Canada) Ltd.
#7-130 Robin Crescent
SASKATOON, Saskatchewan S7L 6M7
Tel: (306) 242-3702
CH,CS

CHILE
ASC Ltda.
Austria 2041
SANTIAGO
Tel: 223-5946, 223-6148
Telex: 340192 ASC CK
P,C

Jorge Calcagni y Cia. Ltda.
Av. Italia 634 Santiago
Casilla 16475
SANTIAGO 9
Tel: 222-0222
Telex: 440283 JCYCL CZ
CM,E,M

Metrolab S.A.

Monjitas 454 of. 206
SANTIAGO
Tel: 395752, 398296
Telex: 340866 METLAB CK
A

Olympia (Chile) Ltda.
Av. Rodrigo de Araya 1045
Casilla 256-V
SANTIAGO 21
Tel: 225-5044
Telex: 340892 OLYMP
Cable: Olympiachile Santiagochile
CH,CS,P

CHINA, People's Republic of

China Hewlett-Packard Co., Ltd.
6th Floor, Sun Hung Kai Centre
30 Harbour Road
HONG KONG

Tel: 5-8323211
Telex: 36678 HEWPA HX
A,C,CH,CS,E,M,P

China Hewlett-Packard Rep. Office
P.O. Box 418

1A Lane 2, Luchang St.
Beiwei Rd., Xuanwu District
BEIJING

Tel: 33-1947, 33-7426
Telex: 22601 CTSHP CN
Cable: 1920
A,CH,CM,CS,E,P

COLOMBIA

Instrumentacia On
H. A. Langebaek & Kier S.A.
Carrera 4A No. 52A-26
Apartado Aereo 6287
BOGOTA 1, D.E.

Tel: 212-1466
Telex: 44400 INST CO
Cable: AARIS Bogota
CM,E,M

Nefromedicas Ltda.
Calle 123 No. 9B-31
Apartado Aereo 100-958
BOGOTA D.E., 10

Tel: 213-5267, 213-1615
Telex: 43415 HEGAS CO
A

Procesa, S.A.
CRA 7 No. 24-89 Piso 25
Torre Colpatria
Apartado Aereo No. 49667
BOGOTA D.E.

Tel: 2344925, 2344958, 2344742
Telex: 43127 COVER CO
C,P

Compumundo
Avenida 15 # 107-80
BOGOTA D.E.

Tel: 214-4458
Telex: 45466 MARICO
P

COSTA RICA

Cientifica Costarricense S.A.
Avenida 2, Calle 5
San Pedro de Montes de Oca
Apartado, 10159
SAN JOSÉ
Tel: 24-38-20, 24-08-19
Telex: 2367 GALGUR CR
CM,E,M

CYPRUS

Telexer S Ltd.
P.O. Box 4809
14C Stassinou Avenue
NICOSIA
Tel: 62698
Telex: 2894 LEVIDO CY
E,M,P

DENMARK

Hewlett-Packard A/S
Datavej 52
DK-3460 BIRKEROD
Tel: (02) 81-66-40
Telex: 37409 hpas dk
A,CH,CM,CS,E,M,P

Hewlett-Packard A/S
Rolighedsvej 32
DK-8240 RISSKOV, Aarhus
Tel: (06) 17-60-00
Telex: 37409 hpas dk
CH,E

DOMINICAN REPUBLIC

Microprog S.A.
Juan Tomás Mejía y Cotes No. 60
Arroyo Hondo
SANTO DOMINGO
Tel: 565-6268
Telex: 4510 ARENTA DR (RCA)
P

ECUADOR

CYEDE Cia. Ltda.
Avenida Eloy Alfaro 1749
y Belgica
Casilla 6423 CCI
QUITO

Tel: 450-975, 243-052
Telex: 2548 CYEDE ED
CM,E,P

Hospitalar S.A.

Robles 625
Casilla 3590
QUITO

Tel: 545-250, 545-122
Telex: 2485 HOSPTL ED
Cable: HOSPITALAR-Quito
M

QUITO

Tel: 2-238-951
Telex: 2298 ECUAME ED

EGYPT

Egyptian International
Office for Foreign Trade
P.O. Box 2558
42 El-Zahraa Street
Dokki, CAIRO,
Tel: 712230
Telex: 93337 EGPOR UN
Cable: EGYPOR
P,A

EGYPT (Cont'd)
INFORMATIC FOR SYSTEMS

22 Talaat Harb Street
CAIRO,
Tel: 759006
Telex: 93697 SAFLM UN
CS

International Engineering Associates
24 Hussein Hegazi Street
Kasr-el-Aini

CAIRO,
Tel: 23829, 21641
Telex: 93830 IEA UN
Cable: INTEGASSO
E

S.S.C. Medical
40 Gezerat El Arab Street
Mohandessin
CAIRO,
Tel: 803844, 805998, 810263
Telex: 20503 SSC UN
M*

EL SALVADOR

IPESA de El Salvador S.A.
29 Avenida Norte 1216
SAN SALVADOR
Tel: 26-6858, 26-6868
Telex: 20539 IPESASAL
A,CH,CM,CS,E,P

FINLAND

Hewlett-Packard Oy
Piispankalliontie 17
02200 **ESPOO**
Tel: 00358-0-88721
Telex: 121563 HEWPA SF
CH,CM,SS,P

Hewlett-Packard Oy
(Olarinluoma 7)
PL 24
02101 **ESPOO** 10
Tel: (90) 4521022
A,E,M

Hewlett-Packard Oy
Aatoksenkatu 10-C
SF-40720-72 **JYVASKYLA**
Tel: (941) 216318
CH

Hewlett-Packard Oy
Kainvuntie 1-C
SF-90140-14 **OULU**
Tel: (981) 338785
CH

FRANCE

Hewlett-Packard France
Z.I. Mercure B
Rue Berthelot
F-13763 Les Milles Cedex
AIX-EN-PROVENCE
Tel: (42) 59-41-02
Telex: 410770F
A,CH,E,M,P*

Hewlett-Packard France
64, rue Marchand Saillant
F-61000 **ALENCON**
Tel: (33) 29 04 42

Hewlett-Packard France
Boite Postale 503
F-25026 **BESANCON**
28 rue de la Republique
F-25000 **BESANCON**
Tel: (81) 83-16-22
Telex: 361157
CH,M

Hewlett-Packard France
13, Place Napoleon III
F-29000 **BREST**
Tel: (98) 03-38-35

Hewlett-Packard France
Chemin des Mouilles
Boite Postale 162
F-69130 **ECULLY** Cedex (Lyon)
Tel: (78) 833-81-25
Telex: 310617F
A,CH,CS,E,M

Hewlett-Packard France
Parc d'Activite du Bois Briard
Ave. du Lac
F-91040 **EVRY** Cedex
Tel: 6 077-8383
Telex: 692315F
E

Hewlett-Packard France
5, Avenue Raymond Chanas
F-38320 **EYBENS** (Grenoble)
Tel: (76) 62-67-98
Telex: 980124 HP GRENOB EYBE
CH

Hewlett-Packard France
Centre d'Affaire Paris-Nord
Bâtiment Ampère 5 étage
Rue de la Commune de Paris
Boite Postale 300
F-93153 **LE BLANC MESNIL**
Tel: (1) 865-44-52
Telex: 211032F
CH,CS,E,M

Hewlett-Packard France
Parc d'Activités Cadere
Quartier Jean Mermoz
Avenue du Président JF Kennedy
F-33700 **MERIGNAC** (Bordeaux)
Tel: (56) 34-00-84
Telex: 550105F
CH,E,M

Hewlett-Packard France
Immueble "Les 3 B"
Nouveau Chemin de la Garde
ZAC de Bois Briand
F-44085 **NANTES** Cedex
Tel: (40) 50-32-22
Telex: 711085F
CH**

Hewlett-Packard France
125, rue du Faubourg Bannier
F-45000 **ORLEANS**
Tel: (38) 68 01 63

Hewlett-Packard France
Zone Industrielle de Courtaboef
Avenue des Tropiques
F-91947 Les Ulis Cedex **ORSAY**
Tel: (6) 907-78-25
Telex: 600048F
A,CH,CM,CS,E,M,P

Hewlett-Packard France
Paris Porte-Maillot
15, Avenue de L'Amiral Bruix
F-75782 **PARIS** CEDEX 16
Tel: (1) 502-12-20
Telex: 613663F
CH,M,P

Hewlett-Packard France
124, Boulevard Tourasse
F-64000 **PAU**
Tel: (59) 80 38 02

Hewlett-Packard France
2 All'EE de la Bourgonnette
F-35100 **RENNES**
Tel: (99) 51-42-44
Telex: 740912F
CH,CM,E,M,P*

Hewlett-Packard France
98 Avenue de Bretagne
F-76100 **ROUEN**
Tel: (35) 63-57-66
Telex: 770035F
CH**,CS

Hewlett-Packard France
4 Rue Thomas Mann
Boite Postale 56
F-67033 **STRASBOURG** Cedex
Tel: (88) 28-56-46
Telex: 890141F
CH,E,M,P*

Hewlett-Packard France
Le PAEripole
20, Chemin du Pigeonnier de la
CAEpiGEere
F-31083 **TOULOUSE** Cedex
Tel: (61) 40-11-12
Telex: 531639F
A,CH,CS,E,P*

Hewlett-Packard France
9, rue Baudin
F-26000 **VALENCE**
Tel: (75) 42 76 16

Hewlett-Packard France
Carolor
ZAC de Bois Briand
F-57640 **VIGY** (Metz)
Tel: (8) 771 20 22
CH

Hewlett-Packard France
Immeuble P'Ericentre
F-59658 **VILLENEUVE D'ASCQ** Cedex
Tel: (20) 91-41-25
Telex: 160124F
CH,E,M,P*

**GERMAN FEDERAL
REPUBLIC**

Hewlett-Packard GmbH
Geschäftsstelle
Keithstrasse 2-4
D-1000 **BERLIN** 30
Tel: (030) 24-90-86
Telex: 018 3405 hpbjn d
A,CH,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Herrenberger Strasse 130
D-7030 **BÖBLINGEN**
Tel: (7031) 14-0
Telex: 07265739
A,CH,CM,CS,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Emanuel-Leutze-Strasse 1
D-4000 **DUSSELDORF**
Tel: (0211) 5971-1
Telex: 085/86 533 hpdd d
A,CH,CS,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Schleefstr. 28a
D-4600 **DORTMUND-Aplerbeck**
Tel: (0231) 45001

Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Berner Strasse 117
Postfach 560 140
D-6000 **FRANKFURT** 56
Tel: (0611) 50-04-1
Telex: 04 13249 hpffm d
A,CH,CM,CS,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Aussenstelle Bad Homburg
Louisenstrasse 115
D-6380 **BAD HAMBURG**
Tel: (06172) 109-0

Hewlett-Packard GmbH
Geschäftsstelle
Kapstadtring 5
D-2000 **HAMBURG** 60
Tel: (040) 63804-1
Telex: 021 63 032 hphh d
A,CH,CS,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Heidering 37-39
D-3000 **HANNOVER** 61
Tel: (0511) 5706-0
Telex: 092 3259
A,CH,CM,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Rosslauer Weg 2-4
D-6800 **MANNHEIM**
Tel: (0621) 70050
Telex: 0462105
A,C,E

Hewlett-Packard GmbH
Geschäftsstelle
Messerschmittstrasse 7
D-7910 **NEU ULM**
Tel: 0731-70241
Telex: 0712816 HP ULM-D
A,C,E*

Hewlett-Packard GmbH
Geschäftsstelle
Ehlicherstr. 13
D-8500 **NÜRNBERG** 10
Tel: (0911) 5205-0
Telex: 0623 860
CH,CM,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Eschenstrasse 5
D-8028 **TAUFKIRCHEN**
Tel: (089) 6117-1
Telex: 0524985
A,CH,CM,E,M,P

GREAT BRITAIN
See United Kingdom



SALES & SUPPORT OFFICES

Arranged alphabetically by country

GREECE

Hewlett-Packard A.E.
178, Kifissias Avenue
6th Floor
Halandri-ATHENS
Greece
Tel: 647 1673, 647 1543, 647 2971
A,CH,CM**,CS**,E,M,P
Kostas Karaynnis S.A.
8 Omirou Street
ATHENS 133
Tel: 32 30 303, 32 37 371
Telex: 215962 RKAR GR
A,CH,CM,CS,E,M,P
PLAISIO S.A.
Eliopoulos Brohers Ltd.
11854
ATHENS
Tel: 34-51-911
Telex: 216286
P

GUATEMALA

IPESA
Avenida Reforma 3-48, Zona 9
GUATEMALA CITY
Tel: 316627, 314786
Telex: 4192 TELTRO GU
A,CH,CM,CS,E,M,P

HONG KONG

Hewlett-Packard Hong Kong, Ltd.
G.P.O. Box 795
5th Floor, Sun Hung Kai Centre
30 Harbour Road
HONG KONG
Tel: 5-8323211
Telex: 66678 HEWPA HX
Cable: HEWPAK HONG KONG
E,CH,CS,P
CET Ltd.
10th Floor, Hua Asia
Bldg. Gloucester
64-66 Gloucester Road
HONG KONG
Tel: (5) 200922
Telex: 85148 CET HX
CM

Schmidt & Co. (Hong Kong) Ltd.
18th Floor, Great Eagle Centre
23 Harbour Road, Wanchai

HONG KONG

Tel: 5-8330222
Telex: 74766 SCHMC HX
A,M

ICELAND

Elding Trading Company Inc.
Hafnarnvoli-Tryggvagotu
P.O. Box 895
IS-REYKJAVIK
Tel: 1-58-20, 1-63-03
M

INDIA

Computer products are sold through
Blue Star Ltd. All computer repairs and
maintenance service is done through
Computer Maintenance Corp.

Blue Star Ltd.
Sabri Complex II Floor
24 Residency Rd.
BANGALORE 560 025

Tel: 55660
Telex: 0845-430
Cable: BLUESTAR
A,CH*,CM,CS*,E

Blue Star Ltd.
Band Box House
Prabhadevi
BOMBAY 400 025

Tel: 422-3101
Telex: 011-3751
Cable: BLUESTAR
A,M

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi

BOMBAY 400 025

Tel: 422-6155
Telex: 011-71193
Cable: FROSTBLUE
A,CH*,CM,CS*,E,M

Blue Star Ltd.
Kalyan, 19 Vishwas Colony
Alkapuri, **BORODA**, 390 005
Tel: 65235
Cable: BLUE STAR

A

Blue Star Ltd.
7 Hare Street
CALCUTTA 700 001

Tel: 12-01-31
Telex: 021-7655
Cable: BLUESTAR
A,M

Blue Star Ltd.
133 Kodambakkam High Road
MADRAS 600 034

Tel: 82057
Telex: 041-379
Cable: BLUESTAR
A,M

Blue Star Ltd.
Bhandari House, 7th/8th Floors
91 Nehru Place

NEW DELHI 110 024
Tel: 682547
Telex: 031-2463
Cable: BLUESTAR
A,CH*,CM,CS*,E,M

Blue Star Ltd.
15/16:C Wellesley Rd.
PUNE 411 011

Tel: 22775
Cable: BLUE STAR
A

Blue Star Ltd.
2-2-47/1108 Bolarum Rd.
SECUNDERABAD 500 003

Tel: 72057
Telex: 0155-459
Cable: BLUEFROST
A,E

Blue Star Ltd.
T.C. 7/603 Poornima
Maruthankuzhi
TRIVANDRUM 695 013

Tel: 65799
Telex: 0884-259
Cable: BLUESTAR
E

Computer Maintenance Corporation Ltd.
115, Sarojini Devi Road
SECUNDERABAD 500 003

Tel: 310-184, 345-774
Telex: 031-2960
CH**

INDONESIA

BERCA Indonesia P.T.
P.O.Box 496/Jkt.
Jl. Abdul Muis 62

JAKARTA

Tel: 21-373009
Telex: 46748 BERSAL IA
Cable: BERSAL JAKARTA
P

BERCA Indonesia P.T.
P.O.Box 2497/Jkt
Antara Bldg., 17th Floor
Jl. Medan Merdeka Selatan 17

JAKARTA-PUSAT

Tel: 21-344-181
Telex: BERSAL IA
A,CS,E,M

BERCA Indonesia P.T.
P.O. Box 174/SBY.

Jl. Kutei No. 11

SURABAYA

Tel: 68172
Telex: 31146 BERSAL SB
Cable: BERSAL-SURABAYA
A*,E,M,P

IRAQ

Hewlett-Packard Trading S.A.
Service Operation
Al Mansoor City 9B/3/7

BAGHDAD

Tel: 551-49-73
Telex: 212-455 HEPAIRAQ IK
CH,CS

IRELAND

Hewlett-Packard Ireland Ltd.
82/83 Lower Leeson Street
DUBLIN 2

Tel: 0001 608800
Telex: 30439
A,CH,CM,CS,E,M,P

Cardiac Services Ltd.
Kilmore Road
Artane

DUBLIN 5

Tel: (01) 351820
Telex: 30439
M

ISRAEL

Eldan Electronic Instrument Ltd.
P.O.Box 1270
JERUSALEM 91000

16, Ohaliav St.
JERUSALEM 94467

Tel: 533 221, 553 242
Telex: 25231 AB/PAKRD IL
A,M

Computation and Measurement
Systems (CMS) Ltd.
11 Masad Street
67060

TEL-AVIV

Tel: 388 388
Telex: 33569 Motil IL
CH,CM,CS,E,P

ITALY

Hewlett-Packard Italiana S.p.A.
Traversa 99C
Via Giulio Petroni, 19

I-70124 BARI

Tel: (080) 41-07-44
M,CH

Hewlett-Packard Italiana S.p.A.
Via Martin Luther King, 38/III
I-40132 **BOLOGNA**

Tel: (051) 402394

Telex: 511630

CH,CS,E,M

Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43G/C

I-95126 CATANIA

Tel: (095) 37-10-87

Telex: 970291

CH

Hewlett-Packard Italiana S.p.A.

Via G. Di Vittorio 9

I-20063 CERNUSCO SUL

NAVIGLIO

(Milano)

Tel: (02) 923691

Telex: 334632

A,CH,CM,CS,E,M,P

Hewlett-Packard Italiana S.p.A.

Via C. Colombo 49

I-20090 TREZZANO SUL

NAVIGLIO

(Milano)

Tel: (02) 4459041

Telex: 322116

CH,CS

Hewlett-Packard Italiana S.p.A.

Via Nuova San Rocco a

Capodimonte, 62/A

I-80131 NAPOLI

Tel: (081) 7413544

Telex: 710698

A**,CH,CS,E,M

Hewlett-Packard Italiana S.p.A.

Viale G. Modugno 33

I-16156 GENOVA PEGLI

Tel: (010) 68-37-07

Telex: 215238

E,C

Hewlett-Packard Italiana S.p.A.

Via Pelizzo 15

I-35128 PADOVA

Tel: (049) 664888

Telex: 430315

A,CH,CS,E,M

Hewlett-Packard Italiana S.p.A.

Viale C. Pavese 340

I-00144 ROMA EUR

Tel: (06) 54831

Telex: 610514

A,CH,CS,E,M,P*



ITALY (Cont'd)

Hewlett-Packard Italiana S.p.A.
Via di Casellina 57/C
I-50018 **SCANDICCI-FIRENZE**
Tel: (055) 753863
CH,E,M

Hewlett-Packard Italiana S.p.A.
Corso Svizzera, 185
I-10144 **TORINO**
Tel: (011) 74 4044
Telex: 221079
A*,CS,CH,E

JAPAN

Yokogawa-Hewlett-Packard Ltd.
152-1, Onna
ATSUGI, Kanagawa, 243
Tel: (0462) 28-0451
CM,C*,E

Yokogawa-Hewlett-Packard Ltd.
Meiji-Seimei Bldg. 6F
3-1 Hon Chiba-Cho
CHIBA, 280
Tel: 472 25 7701
E,CH,CS

Yokogawa-Hewlett-Packard Ltd.
Yasuda-Seimei Hiroshima Bldg.
6-11, Hon-dori, Naka-ku
HIROSHIMA, 730
Tel: 82-241-0611

Yokogawa-Hewlett-Packard Ltd.
Towa Building
2-3, Kaigan-dori, 2 Chome Chuo-ku
KOBE, 650
Tel: (078) 392-4791
C,E

Yokogawa-Hewlett-Packard Ltd.
Kumagaya Asahi 82 Bldg
3-4 Tsukuba
KUMAGAYA, Saitama 360
Tel: (0485) 24-6563
CH,CM,E

Yokogawa-Hewlett-Packard Ltd.
Asahi Shinbun Daiichi Seimei Bldg.
4-7, Hanabata-cho
KUMAMOTO, 860
Tel: (0963) 54-7311
CH,E

Yokogawa-Hewlett-Packard Ltd.
Shin-Kyoto Center Bldg.
614, Higashi-Shiokoji-cho
Karasuma-Nishiiru
Shiokoji-dori, Shimogyo-ku
KYOTO, 600
Tel: 075-343-0921
CH,E

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsui Bldg
4-73, Sanno-maru, 1 Chome
MITO, Ibaraki 310
Tel: (0292) 25-7470
CH,CM,E

Yokogawa-Hewlett-Packard Ltd.
Meiji-Seimei Kokubun Bldg. 7-8
Kokubun, 1 Chome, Sendai
MIYAGI, 980
Tel: (0222) 25-1011
Telex:
C,E

Yokogawa-Hewlett-Packard Ltd.
Sumitomo Seimei 14-9 Bldg.
Meiki-Minami, 2 Chome
Nakamura-ku
NAGOYA, 450
Tel: (052) 571-5171
CH,CM,CS,E,M
Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg.,
4-20 Nishinakajima, 5 Chome
Yodogawa-ku
OSAKA, 532
Tel: (06) 304-6021
Telex: YHPOSA 523-3624
A,CH,CM,CS,E,M,P*

Yokogawa-Hewlett-Packard Ltd.
27-15, Yabe, 1 Chome
SAGAMIHARA Kanagawa, 229
Tel: 0427 59-1311
Yokogawa-Hewlett-Packard Ltd.
Daiichi Seimei Bldg.
7-1, Nishi Shinjuku, 2 Chome
Shinjuku-ku, **TOKYO** 160
Tel: 03-348-4611
CH,E

Yokogawa-Hewlett-Packard Ltd.
29-21 Takaido-Higashi, 3 Chome
Suginami-ku **TOKYO** 168
Tel: (03) 331-6111
Telex: 232-2024 YHPTOK
A,CH,CM,CS,E,M,P*

Yokogawa-Hewlett-Packard Ltd.
Daiichi Asano Building
2-8, Odori, 5 Chome
UTSUNOMIYA, Tochigi 320
Tel: (0286) 25-7155
CH,CS,E

Yokogawa-Hewlett-Packard Ltd.
Yasuda Seimei Nishiguchi Bldg.
30-4 Tsuruya-cho, 3 Chome
YOKOHAMA 221
Tel: (045) 312-1252
CH,CM,E

JORDAN

Scientific and Medical Supplies Co.
P.O. Box 1387

AMMAN

Tel: 24907, 39907
Telex: 21456 SABCO JO
CH,E,M,P

KENYA

ADCOM Ltd., Inc., Kenya
P.O.Box 30070

NAIROBI

Tel: 331955
Telex: 22639
E,M

KOREA

Samsung Hewlett-Packard Co. Ltd.
12 Fl. Kinam Bldg.
San 75-31, Yeoksam-Dong
Kangnam-Ku
Yeongdong P.O. Box 72
SEOUL
Tel: 555-7555, 555-5447
Telex: K27364 SAMSAN
A,CH,CM,CS,E,M,P

KUWAIT

Al-Khaldiya Trading & Contracting
P.O. Box 830
SAFAT
Tel: 424910, 411726
Telex: 22481 AREEG KT
Cable: VISCOUNT
E,M,A
Photo & Cine Equipment
P.O. Box 270

SAFAT

Tel: 2445111
Telex: 22247 MATIN KT
Cable: MATIN KUWAIT
P

W.J. Towell Computer Services
P.O. Box 75

SAFAT

Tel: 2462640/1
Telex: 30336 TOWELL KT
C

LEBANON

Computer Information Systems
P.O. Box 11-6274

BEIRUT

Tel: 89 40 73
Telex: 42309
C,E,M,P

LUXEMBOURG

Hewlett-Packard Belgium S.A./N.V.
Blvd de la Woluwe, 100
Woluwedal
B-1200 **BRUSSELS**
Tel: (02) 762-32-00
Telex: 23-494 paloben bru
A,CH,CM,CS,E,M,P

MALAYSIA

Hewlett-Packard Sales (Malaysia)
Sdn. Bhd.
1st Floor, Bangunan British
American
Jalan Semantan, Damansara Heights
KUALA LUMPUR 23-03

Tel: 943022

Telex: MA31011

A,CH,E,M,P*

Protel Engineering

P.O.Box 1917
Lot 6624, Section 64
23/4 Pending Road
Kuching, **SARAWAK**

Tel: 36299

Telex: MA 70904 PROMAL

Cable: PROTELENG

A,E,M

MALTA

Philip Toledo Ltd.
Notabile Rd.

Tel: 447 47, 455 66
Telex: Media MW 649
E,P,M

MEXICO

Hewlett-Packard Mexicana, S.A.
de C.V.
Av. Periferico Sur No. 6501
Tepepan, Xochimilco
16020 **MEXICO D.F.**
Tel: 6-76-46-00
Telex: 17-74-507 HEWPACK MEX
A,CH,CS,E,M,P

Hewlett-Packard Mexicana, S.A.
de C.V.
Czda. del Valle
409 Ote. 1 ° Piso
Colonia del Valle
Municipio de Garza Garcia
66220 **MONTERREY**, Nuevo LaOon
Tel: 78 42 41
Telex: 038 410
CH

Equipos Cientificos de Occidente, S.A.
Av. Lazaro Cardenas 3540

GUADALAJARA

Tel: 21-66-91
Telex: 0684186 ECOME
A

Infograficas y Sistemas del Noreste, S.A.
Rio Orinoco #171 Oriente
Despacho 2001
Colonia Del Valle
MONTERREY
Tel: 782499, 781259A
A

MOROCCO

Dolbeau
81 rue Karatchi
CASABLANCA
Tel: 3041-82, 3068-38
Telex: 23051, 22822
E

Gerep

2 rue d'Agadir
Boite Postale 156
CASABLANCA
Tel: 272093, 272095
Telex: 23 739
P

Sema-Maroc

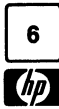
Rue Lapebie
CASABLANCA
Tel: 26.09.80
CH,CS,P

NETHERLANDS

Hewlett-Packard Nederland B.V.
Van Heuven Goedhartlaan 121
NL 1181KK **AMSTELVEEN**
P.O. Box 667
NL1180 AR **AMSTELVEEN**
Tel: (020) 47-20-21
Telex: 13 216 HEPAC NL
A,CH,CM,CS,E,M,P

Hewlett-Packard Nederland B.V.
Bongerd 2
NL 2906VK **CAPELLE A/D IJSSEL**
P.O. Box 41
NL 2900AA **CAPELLE A/D IJSSEL**
Tel: (10) 51-64-44
Telex: 21261 HEPAC NL
A,CH,CS,E

Hewlett-Packard Nederland B.V.
Pastoor Petersstraat 134-136
NL 5612 LV **EINDHOVEN**
P.O. Box 2342
NL 5600 CH **EINDHOVEN**
Tel: (040) 326911
Telex: 51484 hepae nl
A,CH**,E,M



SALES & SUPPORT OFFICES

Arranged alphabetically by country

NEW ZEALAND

Hewlett-Packard (N.Z.) Ltd.
5 Owens Road
P.O. Box 26-189

EPSOM, AUCKLAND

Tel: 687-159
Cable: HEWPAK Auckland
CH,CS,CM,E,P*

Hewlett-Packard (N.Z.) Ltd.
4-12 Cruickshank Street
Kilbirnie, WELLINGTON 3

P.O. Box 9443
Courtenay Place, WELLINGTON 3
Tel: 877-199

Cable: HEWPACK Wellington
CH,CS,CM,E,P

Northrop Instruments & Systems Ltd.
369 Khyber Pass Road
P.O. Box 8602

AUCKLAND

Tel: 794-091
Telex: 60605

A,M

Northrop Instruments & Systems Ltd.
110 Mandeville St.

P.O. Box 8388

CHRISTCHURCH

Tel: 488-873

Telex: 4203

A,M

Northrop Instruments & Systems Ltd.
Sturdee House

85-87 Ghuznee Street

P.O. Box 2406

WELLINGTON

Tel: 850-091

Telex: NZ 3380

A,M

NORTHERN IRELAND

See United Kingdom

NORWAY

Hewlett-Packard Norge A/S
Folke Bernadottes vei 50
P.O. Box 3558

N-5033 FYLLINGSDALEN (Bergen)

Tel: 0047/75/16 55 40

Telex: 16621 hpnas n

CH,CS,E,M

Hewlett-Packard Norge A/S

UCOsterndalen 16-18

P.O. Box 34

N-1345 OCUSTERÅS

Tel: 0047/2/17 11 80

Telex: 16621 hpnas n

A,CH,CM,CS,E,M,P

OMAN

Khimjil Ramdas

P.O. Box 19

MUSCAT

Tel: 722225, 745601

Telex: 3289 BROKER MB MUSCAT

P

Suhail & Saud Bahwan

P.O. Box 169

MUSCAT

Tel: 734 201-3

Telex: 3274 BAHWAN MB

E

Imtac LLC

P.O. Box 8676

MUTRAH

Tel: 601695

Telex: 5741 Tawoos On

A,C,M

PAKISTAN

Mushko & Company Ltd.

House No. 16, Street No. 16

Sector F-6/3

ISLAMABAD

Tel: 824545

Cable: FEMUS Islamabad

A,E,M,P*

Mushko & Company Ltd.

Oosman Chambers

Abdullah Haroon Road

KARACHI 0302

Tel: 524131, 524132

Telex: 2894 MUSKO PK

Cable: COOPERATOR Karachi

A,E,M,P*

PANAMA

ElectroOnico Balboa, S.A.

Calle Samuel Lewis, Ed. Alfa

Apartado 4929

PANAMA 5

Tel: 63-6613, 63-6748

Telex: 3483 ELECTRON PG

A,CM,E,M,P

PERU

Cía Electro Médica S.A.

Los Flamencos 145, San Isidro

Casilla 1030

LIMA 1

Tel: 41-4325, 41-3703

Telex: Pub. Booth 25306

CM,E,M,P

SAMS

Río De La Plata 305

SAN ISIDRO

Tel: 419928

Telex: 394 20450 PELIBERTAD

P

PHILIPPINES

The Online Advanced Systems

Corporation

Rico House, Amorsolo Cor. Herrera

Street

Legaspi Village, Makati

P.O. Box 1510

Metro MANILA

Tel: 815-38-11 (up to 16)

Telex: 63274 Online PN

A,CH,CS,E,M

Electronic Specialists and

Proponents Inc.

690-B Epifanio de los Santos

Avenue

Cubao, QUEZON CITY

P.O. Box 2649 Manila

Tel: 98-96-81, 98-96-82, 98-96-83

Telex: 40018, 42000 ITT GLOBE MAC-

KAY BOOTH

P

PORTUGAL

Mundinter

Intercambio Mundial de ComAercio

S.A.R.L.

P.O. Box 2761

Av. Antonio Augusto de Aguiar 138

P-LISBON

Tel: (19) 53-21-31, 53-21-37

Telex: 16691 munter p

M

Soquimica

Av. da Liberdade, 220-2

1298 LISBOA Codex

Tel: 56 21 81/2/3

Telex: 13316 SABASA

P

Telectra-Empresa Técnica de

Equipamentos Eléctricos S.A.R.L.

Rua Rodrigo da Fonseca 103

P.O. Box 2531

P-LISBON 1

Tel: (19) 68-60-72

Telex: 12598

CM,E

Rarcentro Ltda

R. Costa Cabral 575

4200 PORTO

Tel: 499174/495173

Telex: 26054

CH,CS

PUERTO RICO

Hewlett-Packard Puerto Rico

101 MuAnoz Rivera Av

Esu. Calle Ochoa

HATO REY, Puerto Rico 00918

Tel: (809) 754-7800

A,CH,CS,CM,M,E,P

QATAR

Computer Arabia

P.O. Box 2750

DOHA

Tel: 883555

Telex: 4806 CHPARB

P

Nasser Trading & Contracting

P.O. Box 1563

DOHA

Tel: 422170

Telex: 4439 NASSER DH

M

SAUDI ARABIA

Modern Electronic Establishment

Hewlett-Packard Division

P.O. Box 281

Thuobah

AL-KHOBAR

Tel: 895-1760, 895-1764

Telex: 671 106 HPMEEK SJ

Cable: ELECTA AL-KHOBAR

CH,CS,E,M

Modern Electronic Establishment

Hewlett-Packard Division

P.O. Box 1228

Redec Plaza, 6th Floor

JEDDAH

Tel: 644 38 48

Telex: 4027 12 FARNAS SJ

Cable: ELECTA JEDDAH

A,CH,CS,CM,E,M,P

Modern Electronic Establishment

Hewlett-Packard Division

P.O. Box 22015

RIYADH

Tel: 491-97 15, 491-63 87

Telex: 202049 MEERYD SJ

CH,CS,E,M

Abdul Ghani El Ajou

P.O. Box 78

RIYADH

Tel: 40 41 717

Telex: 200 932 EL AJOU

P

SCOTLAND

See United Kingdom

SINGAPORE

Hewlett-Packard Singapore (Sales)

Pte. Ltd.

#08-00 Inchcape House

450-2 Alexandra Road

P.O. Box 58 Alexandra Rd. Post

Office

SINGAPORE, 9115

Tel: 631788

Telex: HPSGSO RS 34209

Cable: HEWPACK, Singapore

A,CH,CS,E,MS,P

Dynamar International Ltd.

Unit 05-11Block 6

Kolam Ayer Industrial Estate

SINGAPORE 1334

Tel: 747-6188

Telex: RS 26283

CM

SOUTH AFRICA

Hewlett-Packard So Africa (Pty.) Ltd.

P.O. Box 120

Howard Place CAPE PROVINCE 7450

Pine Park Center, Forest Drive, Pine-

lands

CAPE PROVINCE 7405

Tel: 53-7954

Telex: 57-20006

A,CH,CM,E,M,P

Hewlett-Packard So Africa (Pty.) Ltd.

P.O. Box 37099

Overport Drive 92

6067

Tel: 28-4178

Telex: 6-22954

CH,CM

Hewlett-Packard So Africa (Pty.) Ltd.

6 Linton Arcade

511 Cape Road

Linton Grange

PORT ELIZABETH 6001

Tel: 041-301201

CH

Hewlett-Packard So Africa (Pty.) Ltd.

Fountain Center

Kalkden Str.

Monument Park

Ext 2

PRETORIA 0105

Tel: 45-5723

Telex: 32163

CH,E

SOUTH AFRICA (Cont'd)

Hewlett-Packard So Africa (Pty.) Ltd.
Private Bag Wendywood
SANDTON 2144
Tel: 802-5111, 802-5125
Telex: 4-20877
Cable: HEWPACK Johannesburg
A,CH,CM,CS,E,M,P

SPAIN

Hewlett-Packard Española S.A.
Calle Entenza, 321
E-BARCELONA 29
Tel: 322.24.51, 321.73.54
Telex: 52603 hpbbee
A,CH,CS,E,M,P

Hewlett-Packard Española S.A.
Calle San Vicente S/No
Edificio Albia II 7B
E-BILBAO 1
Tel: 423.83.06
A,CH,E,M

Hewlett-Packard Española S.A.
Crta. de la Coruña, Km. 16, 400
Las Rozas
E-MADRID
Tel: (1) 637.00.11
Telex: 23515 HPE
CH,CS,M

Hewlett-Packard Española S.A.
Avda. S. Francisco Javier, S/no
Planta 10. Edificio Sevilla 2,
E-SEVILLA 5
Tel: 64.44.54
Telex: 72933
A,CS,M,P

Hewlett-Packard Española S.A.
C/Isabel La Católica, 8
E-46004 **VALENCIA**
Tel: 0034/6/351 59 44
CH,P

SWEDEN

Hewlett-Packard Sverige AB
Sunnanvagen 14K
S-22226 **LUND**
Tel: (046) 13-69-79
Telex: (854) 17886 (via Spånga office)
CH

Hewlett-Packard Sverige AB
Östra Tullgatan 3
S-21128 **MALMÖ**
Tel: (040) 70270
Telex: (854) 17886 (via Spånga office)
CH

Hewlett-Packard Sverige AB
Våstra Vintergatan 9
S-70344 **ÖREBRO**
Tel: (19) 10-48-80
Telex: (854) 17886 (via Spånga office)
CH

Hewlett-Packard Sverige AB
Skalholtsgatan 9, Kista
Box 19
S-16393 **SPÅNGA**
Tel: (08) 750-2000
Telex: (854) 17886
Telefax: (08) 7527781
A,CH,CM,CS,E,M,P

Hewlett-Packard Sverige AB
Frötlisgatan 30
S-42132 **VÄSTRA-FRÖLUNDA**
Tel: (031) 49-09-50
Telex: (854) 17886 (via Spånga office)
CH,E,P

SWITZERLAND

Hewlett-Packard (Schweiz) AG
Clarastrasse 12
CH-4058 **BASEL**
Tel: (61) 33-59-20
A

Hewlett-Packard (Schweiz) AG
7, rue du Bois-du-Lan
Case Postale 365
CH-1217 **MEYRIN 2**
Tel: (0041) 22-83-11-11
Telex: 27333 HPAG CH
CH,CM,CS

Hewlett-Packard (Schweiz) AG
Allmend 2
CH-8967 **WIDEN**
Tel: (0041) 57 31 21 11
Telex: 53933 hpag ch
Cable: HPAG CH
A,CH,CM,CS,E,M,P

SYRIA

General Electronic Inc.
Nuri Basha Ahnaf Ebn Kays Street
P.O. Box 5781

DAMASCUS
Tel: 33-24-87
Telex: 411 215
Cable: **ELECTROBOR DAMASCUS E**

Middle East Electronics
P.O.Box 2308
Abu Rummaneh
DAMASCUS
Tel: 33 45 92
Telex: 411 304
M

TAIWAN

Hewlett-Packard Taiwan
Kaohsiung Office
11/F 456, Chung Hsiao 1st Road

KAOSIUNG
Tel: (07) 2412318
CH,CS,E

Hewlett-Packard Taiwan
8th Floor Hewlett-Packard Building
337 Fu Hsing North Road

TAIPEI
Tel: (02) 712-0404
Telex: 24439 HEWPACK
Cable: HEWPACK Taipei
A,CH,CM,CS,E,M,P
Ing Lih Trading Co.
3rd Floor, 7 Jen-Ai Road, Sec. 2
TAIPEI 100
Tel: (02) 3948191
Cable: **INGLIH TAIPEI**
A

THAILAND

Unimesa
30 Patpong Ave., Suriwong
BANGKOK 5
Tel: 235-5727
Telex: 84439 Simonco TH
Cable: **UNIMESA Bangkok**
A,CH,CS,E,M

Bangkok Business Equipment Ltd.
5/5-6 Dejo Road

BANGKOK
Tel: 234-8670, 234-8671
Telex: 87669-BEQUIPT TH
Cable: **BUSIQUIPT Bangkok P**

TOGO

Societe Africaine De
Promotion
B.P. 12271

LOME
Tel: 21-62-88
Telex: 5304
P

TRINIDAD & TOBAGO

Caribbean Telecoms Ltd.
Corner McAllister Street &
Eastern Main Road, Laventille
P.O. Box 732

PORT-OF-SPAIN
Tel: 624-4213
Telex: 22561 CARTEL WG
Cable: **CARTEL, PORT OF SPAIN CM,E,M,P**
Computer and Controls Ltd.
P.O. Box 51
66 Independence Square
PORT-OF-SPAIN
Tel: 623-4472
Telex: 3000 POSTLX WG
P

TUNISIA

Tunisie Electronique
31 Avenue de la Liberté
TUNIS
Tel: 280-144
CH,CS,E,P

Corema
1 ter. Av. de Carthage

TUNIS
Tel: 253-821
Telex: 12319 CABAM TN
M

TURKEY

E.M.A
Mediha Eldem Sokak No. 41/6
Yenisehir
ANKARA
Tel: 319175
Telex: 42321 KTX TR
Cable: **EMATRADE ANKARA M**

Kurt & Kurt A.S.
Mithatpasa Caddesi No. 75
Kat 4 Kizilay
ANKARA
Tel: 318875/6/7/8
Telex: 42490 MESR TR
A

Saniva Bilgisayar Sistemleri A.S.
Buyukdere Caddesi 103/6
Gayrettepe

ISTANBUL
Tel: 1673180
Telex: 26345 SANI TR
C,P

Teknim Company Ltd.
Iran Caddesi No. 7
Kavaklidere

ANKARA
Tel: 275800
Telex: 42155 TKNM TR
E,CM

UNITED ARAB EMIRATES

Emitac Ltd.
P.O. Box 1641

SHARJAH,
Tel: 591181
Telex: 68136 EMITAC EM
Cable: **EMITAC SHARJAH E,C,M,P,A**

Emitac Ltd.
P.O. Box 2711

ABU DHABI,
Tel: 820419-20
Cable: **EMITACH ABUDHABI**

Emitac Ltd.
P.O. Box 8391

DUBAI,
Tel: 377951

Emitac Ltd.
P.O. Box 473

RAS AL KHAIMAH,
Tel: 28133, 21270

UNITED KINGDOM

GREAT BRITAIN
Hewlett-Packard Ltd.

Trafalgar House
Navigation Road
ALTRINCHAM
Cheshire WA14 1NU
Tel: 061 928 6422
Telex: 668068
A,CH,CS,E,M,M,P

Hewlett-Packard Ltd.
Miller House
The Ring, **BRACKNELL**
Berks RG12 1XN
Tel: 44344 424898
Telex: 848733
E

Hewlett-Packard Ltd.
Elstree House, Elstree Way
BOREHAMWOOD, Herts WD6 1SG
Tel: 01 207 5000
Telex: 8952716
E,CH,CS,P

Hewlett-Packard Ltd.
Oakfield House, Oakfield Grove
Clifton **BRISTOL,** Avon BS8 2BN
Tel: 0272 736806
Telex: 444302
CH,CS,E,P



SALES & SUPPORT OFFICES

Arranged alphabetically by country

GREAT BRITAIN (Cont'd)

Hewlett-Packard Ltd.
Bridewell House
Bridewell Place
LONDON EC4V 6BS
Tel: 01 583 6565
Telex: 298163
CH,CS,P

Hewlett-Packard Ltd.
Fourier House
257-263 High Street
LONDON COLNEY
Herts. AL2 1HA, St. Albans
Tel: 0727 24400
Telex: 1-8952716
CH,CS

Hewlett-Packard Ltd.
Pontefract Road
NORMANTON, West Yorkshire WF6 1RN
Tel: 0924 895566
Telex: 557355
CH,CS,P

Hewlett-Packard Ltd.
The Quadrangle
106-118 Station Road
REDHILL, Surrey RH1 1PS
Tel: 0737 68655
Telex: 947234
CH,CS,E,P

Hewlett-Packard Ltd.
Avon House
435 Stratford Road
Shirley, **SOLIHULL**, West Midlands
B90 4BL
Tel: 021 745 8800
Telex: 339105
CH,CS,E,P

Hewlett-Packard Ltd.
West End House
41 High Street, West End
SOUTHAMPTON
Hampshire SO3 3DQ
Tel: 04218 6767
Telex: 477138
CH,CS,P

Hewlett-Packard Ltd.
King Street Lane
Winnersh, **WOKINGHAM**
Berkshire RG11 5AR
Tel: 0734 784774
Telex: 847178
A,CH,CS,E,M,P

Hewlett-Packard Ltd.
Nine Mile Ride
Easthampstead, **WOKINGHAM**
Berkshire, 3RG11 3LL
Tel: 0344 773100
Telex: 848805
CH,CS,E,P

IRELAND

NORTHERN IRELAND

Hewlett-Packard Ltd.
Cardiac Services Building
95A Finaghy Road South
BELFAST BT 10 OBY
Tel: 0232 625-566
Telex: 747626
CH,CS

SCOTLAND

Hewlett-Packard Ltd.
SOUTH QUEENSFERRY
West Lothian, EH30 9TG
Tel: 031 331 1188
Telex: 72682
CH,CM,CS,E,M,P

UNITED STATES

Alabama

Hewlett-Packard Co.
700 Century Park South, Suite 128
BIRMINGHAM, AL 35226
Tel: (205) 822-6802
C,CH,CS,P*

Hewlett-Packard Co.
420 Wynn Drive
P.O. Box 7700
HUNTSVILLE, AL 35807
Tel: (205) 830-2000
C,CH,CM,CS,E,M*

Alaska

Hewlett-Packard Co.
3601 C St., Suite 1234
ANCHORAGE, AK 99503
Tel: (907) 563-8855
CH,CS,E

Arizona

Hewlett-Packard Co.
8080 Pointe Parkway West
PHOENIX, AZ 85044
Tel: (602) 273-8000
A,CH,CM,CS,E,M

Hewlett-Packard Co.
2424 East Aragon Road
TUCSON, AZ 85706
Tel: (602) 573-7400
CH,E,M**

California

Hewlett-Packard Co.
99 South Hill Dr.
BRISBANE, CA 94005
Tel: (415) 330-2500
CH,CS

Hewlett-Packard Co.
P.O. Box 7830 (93747)
5060 E. Clinton Avenue, Suite 102
FRESNO, CA 93727
Tel: (209) 252-9652
CH,CS,M

Hewlett-Packard Co.
1421 S. Manhattan Av.
FULLERTON, CA 92631
Tel: (714) 999-6700
CH,CM,CS,E,M

Hewlett-Packard Co.
320 S. Kellogg, Suite B
GOLETA, CA 93117
Tel: (805) 967-3405
CH

Hewlett-Packard Co.
5400 W. Rosecrans Blvd.
LAWDALE, CA 90260
P.O. Box 92105
LOS ANGELES, CA 90009
Tel: (213) 643-7500
Telex: 910-325-6608
CH,CM,CS,M

Hewlett-Packard Co.
3155 Porter Drive
PALO ALTO, CA 94304
Tel: (415) 857-8000
CH,CS,E

Hewlett-Packard Co.
4244 So. Market Court, Suite A
P.O. Box 15976
SACRAMENTO, CA 95813
Tel: (916) 929-7222
A*,CH,CS,E,M

Hewlett-Packard Co.
9606 Aero Drive
P.O. Box 23333
SAN DIEGO, CA 92123
Tel: (619) 279-3200
CH,CM,CS,E,M

Hewlett-Packard Co.
2305 Camino Ramon 'C'
SAN RAMON, CA 94583
Tel: (415) 838-5900
CH,CS

Hewlett-Packard Co.
3005 Scott Boulevard
SANTA CLARA, CA 95050
Tel: (408) 988-7000
Telex: 910-338-0586
A,CH,CM,CS,E,M

Hewlett-Packard Co.
5703 Corsa Avenue
WESTLAKE VILLAGE, CA 91362
Tel: (213) 706-6800
E*,CH*,CS*

Colorado

Hewlett-Packard Co.
24 Inverness Place, East
ENGLEWOOD, CO 80112
Tel: (303) 649-5000
A,CH,CM,CS,E,M

Connecticut

Eff. Dec. 1, 1984
Hewlett-Packard Co.
500 Sylvan Av.
BRIDGEPORT, CT 06606
Tel: (203) 371-6454
CH,CS,E

Hewlett-Packard Co.
47 Barnes Industrial Road South
P.O. Box 5007
WALLINGFORD, CT 06492
Tel: (203) 265-7801
A,CH,CM,CS,E,M

Florida

Hewlett-Packard Co.
2901 N.W. 62nd Street
P.O. Box 24210
FORT LAUDERDALE, FL 33307
Tel: (305) 973-2600
CH,CS,E,M,P*

Hewlett-Packard Co.
4080 Woodcock Drive, Suite 132
JACKSONVILLE, FL 32207
Tel: (904) 398-0663
C*,CH*,M**

Hewlett-Packard Co.
6177 Lake Ellenor Drive
P.O. Box 13910
ORLANDO, FL 32859
Tel: (305) 859-2900
A,C,CH,CM,CS,E,P*

Hewlett-Packard Co.
4700 Bayoue Blvd.
Building 5
PENSACOLA, FL 32505
Tel: (904) 476-8422
A,C,CH,CM,CS,M

Hewlett-Packard Co.
5550 Idlewild, #150
P.O. Box 15200
TAMPA, FL 33684
Tel: (813) 884-3282
A*,C,CH,CS,E*,M*,P*

Georgia

Hewlett-Packard Co.
2000 South Park Place
P.O. Box 105005
ATLANTA, GA 30348
Tel: (404) 955-1500
Telex: 810-766-4890
A,C,CH,CM,CS,E,M,P*

Hawaii

Hewlett-Packard Co.
Kawaiahao Plaza, Suite 190
567 South King Street
HONOLULU, HI 96813
Tel: (808) 526-1555
A,CH,E,M

Illinois

Hewlett-Packard Co.
304 Eldorado Road
P.O. Box 1607
BLOOMINGTON, IL 61701
Tel: (309) 662-9411
CH,M**

Hewlett-Packard Co.
525 W. Monroe, #1300
CHICAGO, IL 60606
Tel: (312) 930-0010
CH,CS

Hewlett-Packard Co.
1200 Diehl
NAPERVILLE, IL 60566
Tel: (312) 357-8800
CH*,CS

Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800
Telex: 910-687-1066
A,CH,CM,CS,E,M

Indiana

Hewlett-Packard Co.
11911 N. Meridian St.
CARMEL, IN 46032
Tel: (317) 844-4100
A,CH,CM,CS,E,M

Iowa

Hewlett-Packard Co.
4070 22nd Av. SW
CEDAR RAPIDS, IA 52404
Tel: (319) 390-4250
CH,CS,E,M



UNITED STATES (Cont'd)

Hewlett-Packard Co.
4201 Corporate Dr.
WEST DES MOINES, IA 50265
Tel: (515) 224-1435
A**,CH,M**

Kentucky

Hewlett-Packard Co.
10300 Linn Station Road, #100
LOUISVILLE, KY 40223
Tel: (502) 426-0100
A,CH,CS,M

Louisiana

Hewlett-Packard Co.
160 James Drive East
ST. ROSE, LA 70087
P.O. Box 1449
KENNER, LA 70063
Tel: (504) 467-4100
A,C,CH,E,M,P*

Maryland

Hewlett-Packard Co.
3701 Koppers Street
BALTIMORE, MD 21227
Tel: (301) 644-5800
Telex: 710-862-1943
A,CH,CM,CS,E,M
Hewlett-Packard Co.
2 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 948-6370
A,CH,CM,CS,E,M

Massachusetts

Hewlett-Packard Co.
1775 Minuteman Road
ANDOVER, MA 01810
Tel: (617) 682-1500
A,C,CH,CS,CM,E,M,P*
Hewlett-Packard Co.
32 Hartwell Avenue
LEXINGTON, MA 02173
Tel: (617) 861-8960
CH,CS,E

Michigan

Hewlett-Packard Co.
4326 Cascade Road S.E.
GRAND RAPIDS, MI 49506
Tel: (616) 957-1970
CH,CS,M
Hewlett-Packard Co.
39550 Orchard Hill Place Drive
NOVI, MI 48050
Tel: (313) 349-9200
A,CH,CS,E,M
Hewlett-Packard Co.
1771 W. Big Beaver Road
TROY, MI 48084
Tel: (313) 643-6474
CH,CS

Minnesota

Hewlett-Packard Co.
2025 W. Larpentour Ave.
ST. PAUL, MN 55113
Tel: (612) 644-1100
A,CH,CM,CS,E,M

Missouri

Hewlett-Packard Co.
1001 E. 101st Terrace
KANSAS CITY, MO 64131
Tel: (816) 941-0411
A,CH,CM,CS,E,M
Hewlett-Packard Co.
13001 Hollenberg Drive
BRIDGETON, MO 63044
Tel: (314) 344-5100
A,CH,CS,E,M

Nebraska

Hewlett-Packard
10824 Old Mill Rd., Suite 3
OMAHA, NE 68154
Tel: (402) 334-1813
CM,M

New Jersey

Hewlett-Packard Co.
120 W. Century Road
PARAMUS, NJ 07652
Tel: (201) 265-5000
A,CH,CM,CS,E,M
Hewlett-Packard Co.
20 New England Av. West
PISCATAWAY, NJ 08854
Tel: (201) 981-1199
A,CH,CM,CS,E

New Mexico

Hewlett-Packard Co.
11300 Lomas Blvd.,N.E.
P.O. Box 11634
ALBUQUERQUE, NM 87112
Tel: (505) 292-1330
CH,CS,E,M

New York

Hewlett-Packard Co.
5 Computer Drive South
ALBANY, NY 12205
Tel: (518) 458-1550
A,CH,E,M
Hewlett-Packard Co.
9600 Main Street
P.O. Box AC
CLARENCE, NY 14031
Tel: (716) 759-8621
CH,CS,E
Hewlett-Packard Co.
200 Cross Keys Office Park
FAIRPORT, NY 14450
Tel: (716) 223-9950
A,CH,CM,CS,E,M
Hewlett-Packard Co.
7641 Henry Clay Blvd.
LIVERPOOL, NY 13088
Tel: (315) 451-1820
A,CH,CM,CS,E,M
Hewlett-Packard Co.
No. 1 Pennsylvania Plaza
55th Floor
34th Street & 8th Avenue
MANHATTAN NY 10119
Tel: (212) 971-0800
CH,CS,M*
Hewlett-Packard Co.
15 Myers Corner Rd.
WAPPINGER FALLS, NY 12590
CM,E

Hewlett-Packard Co.
250 Westchester Avenue
WHITE PLAINS, NY 10604
Tel: (914) 684-6100
CM,CH,CS,E
Hewlett-Packard Co.
3 Crossways Park West
WOODBURY, NY 11797
Tel: (516) 921-0300
A,CH,CM,CS,E,M

North Carolina

Hewlett-Packard Co.
305 Gregson Dr.
CARY, NC 27511
Tel: (919) 467-6600
C,CH,CM,CS,E,M,P*
Hewlett-Packard Co.
9600-H Southern Pine Blvd.
CHARLOTTE, NC 28210
Tel: (704) 527-8780
CH*,CS*
Hewlett-Packard Co.
5605 Roanne Way
P.O. Box 26500
GREENSBORO, NC 27420
Tel: (919) 852-1800
A,C,CH,CM,CS,E,M,P*

Ohio

Hewlett-Packard Co.
9920 Carver Road
CINCINNATI, OH 45242
Tel: (513) 891-9870
CH,CS,M
Hewlett-Packard Co.
16500 Sprague Road
CLEVELAND, OH 44130
Tel: (216) 243-7300
A,CH,CM,CS,E,M
Hewlett-Packard Co.
980 Springboro Pike
MIAMISBURG, OH 45343
Tel: (513) 859-8202
A,CH,CM,E*,M
Hewlett-Packard Co.
675 Brooksedge Blvd.
WESTERVILLE, OH 43081
Tel: (614) 436-1041
CH,CM,CS,E*

Oklahoma

Hewlett-Packard Co.
304 N. Meridian, Suite A
P.O. Box 75609
OKLAHOMA CITY, OK 73147
Tel: (405) 946-9499
C,CH,CS,E*,M
Hewlett-Packard Co.
3840 S. 103rd E. Ave., #100
P.O. Box 35747
TULSA, OK 74153
Tel: (918) 665-3300
A**,C,CH,CS,M*,E,P*

Oregon

Hewlett-Packard Co.
9255 S. W. Pioneer Court
P.O. Box 328
WILSONVILLE, OR 97070
Tel: (503) 682-8000
A,CH,CS,E*,M

Pennsylvania

Hewlett-Packard Co.
50 Dorchester Rd.
P.O. Box 6080
HARRISBURG, PA 17111
Tel: (717) 657-5900
C
Hewlett-Packard Co.
111 Zeta Drive
PITTSBURGH, PA 15238
Tel: (412) 782-0400
A,CH,CS,E,M
Hewlett-Packard Co.
2750 Monroe Boulevard
P.O. Box 713
VALLEY FORGE, PA 19482
Tel: (215) 666-9000
A,CH,CM,CS,E,M
Hewlett-Packard Co.
Brookside Park, Suite 122
1 Harbison Way
P.O. Box 21708
COLUMBIA, SC 29221
Tel: (803) 732-0400
A,C,CH,CS,M
Hewlett-Packard Co.
100 Executive Cntr. Dr.
Koger Executive Center
Chesterfield Bldg., Suite 124
GREENVILLE, SC 29615
Tel: (803) 297-4120
C
Hewlett-Packard Co.
One Energy Cntr. #200
Pellissippi Pkwy.
P.O. Box 22490
KNOXVILLE, TN 37933
Tel: (615) 966-4747
A,C,CH,CS,M
Hewlett-Packard Co.
3070 Directors Row
MEMPHIS, TN 38131
Tel: (901) 346-8370
A,C,M
Hewlett-Packard Co.
220 Great Circle Road, Suite 116
NASHVILLE, TN 37228
Tel: (615) 255-1271
C,M,P*
Hewlett-Packard Co.
11002-B Metric Boulevard
AUSTIN, TX 78758
Tel: (512) 835-6771
C,CM,E,P*
Hewlett-Packard Co.
5700 Cromo Dr
P.O. Box 12903
EL PASO, TX 79913
Tel: (915) 833-4400
CH,E*,M**



SALES & SUPPORT OFFICES

Arranged alphabetically by country

UNITED STATES (Cont'd)

Hewlett-Packard Co.
3952 Sand Shell St
FORT WORTH, TX 76137
Tel: (817) 232-9500
A,C,CH,E,M

Hewlett-Packard Co.
10535 Harwin Drive
P.O. Box 42816
HOUSTON, TX 77042
Tel: (713) 776-6400
A,C,CH,CS,E,M,P*

Hewlett-Packard Co.
511 W. John W. Carpenter Fwy.
Royal Tech. Center #100
IRVINE, TX 75062
Tel: (214) 556-1950
C,CH,CS,E

Hewlett-Packard Co.
930 E. Campbell Rd.
P.O. Box 83/1270
RICHARDSON, TX 75083-1270
Tel: (214) 231-6101
A,CH,CM,CS,E,M,P*

Hewlett-Packard Co.
1020 Central Parkway South
P.O. Box 32993
SAN ANTONIO, TX 78232
Tel: (512) 494-9336
A,C,CH,CS,E,M,P*

Utah

Hewlett-Packard Co.
3530 W. 2100 South
P.O. Box 26626
SALT LAKE CITY, UT 84126
Tel: (801) 974-1700
A,CH,CS,E,M

Virginia

Hewlett-Packard Co.
4305 Cox Road
GLEN ALLEN, VA 23060
P.O. Box 9669
RICHMOND, VA 23228
Tel: (804) 747-7750
A,C,CH,CS,E,M,P*

Washington

Hewlett-Packard Co.
15815 S.E. 37th Street
BELLEVUE, WA 98006
Tel: (206) 643-4000
A,CH,CM,CS,E,M

Hewlett-Packard Co.
708 North Argonne Road
P.O. Box 3808
SPOKANE, WA 99220-3808
Tel: (509) 922-7000
CH,CS

West Virginia

Hewlett-Packard Co.
4604 MacCorkle Ave.
CHARLESTON, WV 25304
Tel: (304) 925-0492
A,M

Wisconsin

Hewlett-Packard Co.
275 N. Corporate Dr.
BROOKFIELD, WI 53005
Tel: (414) 784-8800
A,CH,CS,E*,M

URUGUAY

Pablo Ferrando S.A.C. e l.
Avenida Italia 2877
Casilla de Correo 370
MONTEVIDEO
Tel: 80-2586
Telex: Public Booth 901
A,CM,E,M

Mini Computadores, Ltda.
Avda. del Libertador Brig
Gral Lavalleja 2071
Local 007

MONTEVIDEO

Tel: 29-55-22
Telex: 901 P BOOTH UY
P

Olympia de Uruguay S.A.
Maquinas de Oficina
Avda. del Libertador 1997
Casilla de Correos 6644

MONTEVIDEO

Tel: 91-1809, 98.-3807
Telex: 6342 OROU UY
P

VENEZUELA

Hewlett-Packard de Venezuela C.A.
3RA Transversal Los Ruices Norte
Edificio Segre 1, 2 & 3
Apartado 50933
CARACAS 1071

Tel: 239-4133
Telex: 251046 HEWPACK
A,CH,CS,E,M,P

Hewlett-Packard de Venezuela C.A.
Residencias Tia Betty Local 1
Avenida 3 y con calle 75
MARACAIBO, Estado Zulia
Apartado 2646
Tel: (061) 75801-75805-75806-80304
Telex: 62464 HPMAR
C,E*

Hewlett-Packard de Venezuela C.A.
Urb. Lomas de Este
Torre Trebol — Piso 11
VALENCIA, Estado Carabobo
Apartado 3347
Tel: (041) 222992/223024
CH,CS,P

Albis Venezolana S.R.L.
Av. Las Marias, Ota. Alix,
El Pedregal
Apartado 81025
CARACAS 1080A
Tel: 747984, 742146
Telex: 24009 ALBIS VC
A

Tecnologica Medica del Caribe, C.A.
Multicentro Empresarial del Este
Ave. Libertador
Edif. Libertador
Nucleo "C" - Oficina 51-52
CARACAS
Tel: 339867/333780
M

CIZUCA

Cientifica Zulia C.A.
Calle 70, Los Olivos
No. 66-86

Apartado 1843

MARACAIBO

Tel: 54-64-37, 54-63-85, 54-64-94
Telex: 62144
A

YUGOSLAVIA

Do Hermes
General Zdanova 4
Telex: YU-61000 BEOGRAD
A,CH,E,P

Hermes

Titova 50

Telex: YU-61000 LJUBLJANA
CH,CS,E,M,P

Elektrotehna

Titova 51

Telex: YU-61000 LJUBLJANA
CM

ZAMBIA

R.J. Tilbury (Zambia) Ltd.
P.O. Box 32792

LUSAKA

Tel: 215590
Telex: 40128
E

ZIMBABWE

Field Technical Sales
45 Kelvin Road, North
P.B. 3458

SALISBURY

Tel: 705 231
Telex: 4-122 RH
E,P

August 1984

HP distributors are printed in italics.

64243-90901, OCTOBER 1985
E1085

Replaces: 64243-90901, August 1985
E0885



PRINTED IN U.S.A.