
HP 64756/7

70136/70236 Emulator PC Interface

User's Guide



HP Part No. 64756-97012
Printed in U.S.A.
July 1994

Edition 5

Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

© Copyright 1990, 1993, 1994, Hewlett-Packard Company.

This document contains proprietary information, which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this document is subject to change without notice.

HP is a trademark of Hewlett-Packard Company.

IBM and PC AT are a registered trademark of International Business Machines Corporation.

MS-DOS is a trademark of Microsoft Corporation.

Hewlett-Packard Company
P.O. Box 2197
1900 Garden of the Gods Road
Colorado Springs, CO 80901-2197, U.S.A.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013. Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, CA 94304 U.S.A. Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Printing History

New editions are complete revisions of the manual. The date on the title page changes only when a new edition is published.

A software code may be printed before the date; this indicates the version level of the software product at the time the manual was issued. Many product updates and fixes do not require manual changes and, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual revisions.

Edition 1	64756-97001, April 1990
Edition 2	64756-97004, August 1990
Edition 3	64756-97006, October 1990
Edition 4	64756-97009, August 1993
Edition 5	64756-97012, July 1994

Using this Manual

This manual covers the following emulators as used with the PC Interface.

- HP 64756F 70136 emulator
- HP 64757F 70236 emulator
- HP 64757G 70236A emulator

For the most part, the 70136, 70236 and 70236A emulators all operate the same way. Differences between the emulators are described where they exist. All of the 70136, 70236 and 70236A emulators will be referred to as the "70136 emulator" in this manual where they are alike. In the specific instances where 70236 or 70236A emulator differs from the 70136 emulator, it will be referred as the "70236 emulator" or "70236A emulator".

This manual:

- Shows you how to use emulation commands by executing them on a sample program and describing their results.
- Shows you how to use the emulator in-circuit (connected to a target system).
- Shows you how to configure the emulator for your development needs. Topics include: restricting the emulator to real-time execution, selecting a target system clock source, and allowing the target system to insert wait states.

This manual will not:

- Show you how to use every PC Interface command and option. The PC Interface is described in the *HP 64700 Emulator's PC Interface: User's Reference*.

Organization

- Chapter 1** **"Introduction"**-This chapter lists the 70136 emulator features and describes how they can help you in developing new hardware and software.
- Chapter 2** **"Getting Started"**-This chapter shows you how to use emulation commands by executing them on a sample program. This chapter describes the sample program and how to:
- load programs into the emulator
 - map memory
 - display and modify memory
 - display registers
 - step through programs
 - run programs
 - set software breakpoints
 - search memory for data
 - use the analyzer
- Chapter 3** **"In-Circuit Emulation"**-This chapter shows you how to plug the emulator into a target system, and how to use the "in-circuit" emulation features.
- Chapter 4** **"Configuring the Emulator"**-You can configure the emulator to adapt it to your specific development needs. This chapter describes the emulator configuration options and how to save and restore particular configurations.
- Chapter 5** **"Using the Emulator"**-This chapter describes emulation topics that are not covered in the "Getting Started" chapter (for example, coordinated measurements and storing memory).
- Appendix A.** **"File Format Reader"**-This appendix describes how to use the File Format Reader from MS-DOS or PC Interface, load absolute files into the emulator, use global and local symbols with the PC Interface.

Contents

1 Introduction to the 70136 Emulator

Introduction	1-1
Purpose of the Emulator	1-1
Features of the 70136 Emulator	1-3
Supported Microprocessors	1-3
Clock Speeds	1-4
Emulation memory	1-4
Analysis	1-4
Registers	1-5
Single-Step	1-5
Breakpoints	1-5
Reset Support	1-5
Configurable Target System Interface	1-5
Foreground or Background Emulation Monitor	1-6
Real-Time Operation	1-6
Easy Products Upgrades	1-6
Limitations, Restrictions	1-7
DMA Support	1-7
User Interrupts	1-7
Interrupts While Executing Step Command	1-7
Accessing Internal I/O Registers	1-7
PC relative addressing in trace list	1-8
"BRKXA" and "RETXA" Instructions in Stepping	1-8
Stepping at Software Breakpoint	1-8
Evaluation Chip	1-8

2 Getting Started

Introduction	2-1
Before You Begin	2-2
Prerequisites	2-2
The sample program	2-2
Figuer 2-1. Sample Program Listing (Cont'd)	2-4
Assembling and Linking the Sample Program	2-6
Starting Up the 70136 PC Interface	2-7

Selecting PC Interface Commands	2-8
Emulator Status	2-8
Mapping Memory	2-8
Which Memory Locations Should Be Mapped?	2-9
Loading Programs into Memory	2-12
File Format	2-12
Target Memory Type for Memory Load	2-12
Force the Absolute File to Be Read	2-13
File Format Options	2-13
Absolute File Name	2-13
Displaying Symbols	2-14
Displaying Global Symbols	2-15
Loading and Displaying Local Symbols	2-17
Transfer Symbols to the Emulator	2-19
PGR register	2-19
Displaying Memory in Mnemonic Format	2-20
Stepping Through the Program	2-21
Specifying a Step Count	2-23
Modifying Memory	2-24
Running the Program	2-25
Searching Memory for Data	2-25
Breaking into the Monitor	2-26
Using Software Breakpoints	2-26
Defining a Software Breakpoint	2-29
Displaying Software Breakpoints	2-29
Setting a Software Breakpoint	2-30
Clearing a Software Breakpoint	2-30
Using the Analyzer	2-31
Resetting the Analysis Specification	2-31
Specifying a Simple Trigger	2-31
Starting the Trace	2-35
Change the Analyzer Display Format	2-35
Displaying the Trace	2-36
For a Complete Description	2-38
Copying Memory	2-39
Resetting the Emulator	2-39
Exiting the PC Interface	2-40

2-Contents

3	In-Circuit Emulation	
	Introduction	3-1
	Prerequisites	3-1
	Installing the Target System Probe	3-2
	Pin Protector (70236/70236A Emulator Only)	3-3
	Auxiliary Output Lines	3-3
	Installing into a 70136 PLCC Type Socket	3-5
	Installing into a 70136 PGA Type Socket	3-6
	Installing into a 70136 QFP Type Socket	3-7
	Installing into a 70236/236A PGA Type Socket	3-8
	Installing into a 70236/70236A QFP Type Socket	3-8
	In-Circuit Configuration Options	3-10
	Using the Target System Clock Source	3-10
	Allowing the Target System to Insert Wait States	3-10
	Enabling NMI and RESET Input from the Target System	3-10
	Running the Emulator from Target Reset	3-11
	Pin State in Background (70136)	3-12
	Pin State in Background (70236/70236A)	3-13
	Target System Interface (70136)	3-14
	Target System Interface (70236/70236A)	3-17
4	Configuring the 70136 Emulator	
	Introduction	4-1
	Prerequisites	4-2
	Accessing the Emulator Configuration Options	4-2
	Internal Clock	4-4
	Real-Time Mode	4-5
	Break on ROM Writes	4-6
	Software Breakpoints	4-6
	CMB Interaction	4-8
	Target Interrupts	4-9
	Target RESET	4-10
	Lock RDY Signal	4-10
	Read PGR Registers	4-11
	AEX Through to Target	4-12
	uPD72291 FPU	4-12
	20 Bit Address Mode	4-13
	Release bus by HOLD	4-13
	Trace DMA Cycles	4-13
	Trace Refresh Cycles	4-14
	Wait count of DMA	4-14

DMA cycle in Background	4-14
Trace Dummy HALTACK	4-15
Bus Sizing Signal for Emul_mem	4-15
Bus Sizing Signal for Target_mem	4-16
Seg:off Translation Method	4-17
Background Monitor Location	4-18
Monitor Type	4-18
Foreground Monitor Address?	4-20
Storing an Emulator Configuration	4-21
Loading an Emulator Configuration	4-21

5 Using the Emulator

Introduction	5-1
Address Syntax	5-2
Address Expression	5-4
Memory Commands	5-5
Load/Dump Address	5-6
Run Commands	5-7
I/O Command	5-8
Map Command	5-8
Define the data bus size	5-8
Breakpoints Command	5-10
REGISTER NAMES and CLASSES (70136 Emulator)	5-11
BASIC(*) class	5-11
NOCLASS	5-11
PGR class	5-11
REGISTER NAMES and CLASSES (70236/70236A Emulator)	5-12
BASIC(*) class	5-12
NOCLASS	5-12
PGR class	5-12
SIO class	5-13
ICU class	5-14
TCU class	5-14
SCU class	5-15
DMA71 class	5-15
DMA37 class	5-16
Making Coordinated Measurements	5-17
Running the Emulator at /EXECUTE	5-18
Breaking on the Analyzer Trigger	5-18
Storing Memory Contents to an Absolute File	5-20

A File Format Readers

- Introduction A-1
- Using the OMF86, NEC30, NEC33 Reader A-2
 - What the Reader Accomplishes A-2
 - Location of the Reader Program A-4
 - Using the Reader from MS-DOS A-4
 - Using the Reader from the PC Interface A-6
 - If the Reader Won't Run A-8
 - Including Reader in a Make File A-8
- Using the HP 64000 Reader A-9
 - What the Reader Accomplishes A-9
 - Location of the HP 64000 Reader Program A-12
 - Using the Reader from MS-DOS A-12
 - Using the Reader from the PC Interface A-12
 - If the Reader Won't Run A-14
 - Including RHP64000 in a Make File A-14

Illustrations

- Figure 1-1. HP 64756/7 Emulator for uPD70136/70236 1-2
- Figure 2-1. Sample Program Listing 2-3
- Figure 2-2. PC Interface Display 2-7
- Figure 2-3. Sample Program Load Map Listing 2-9
- Figure 2-4. Memory Map Configuration 2-11
- Figure 2-5. Modifying the Trace Specification 2-34
- Figure 2-6. Modifying the Pattern Specification 2-34
- Figure 3-1. Auxiliary Output Lines (70136 Emulator) 3-3
- Figure 3-2. Installing into a 70136 PLCC type socket 3-5
- Figure 3-3. Installing into a 70136 PGA type socket 3-6
- Figure 3-4. Installing into a 70136 QFP type socket 3-7
- Figure 3-5. Installing into a 70236 PGA type socket 3-9
- Figure 4-1. General Emulator Configuration (70136) 4-2
- Figure 4-2. General Emulator Configuration (70236) 4-3

Tables

- Table 5-1. Address expression syntax 5-4
- Table A-1. How to Access Variables A-3
- Table A-2. How to Access Variables A-11



Introduction to the 70136 Emulator

Introduction

The topics in this chapter include:

- Purpose of the emulator
- Features of the emulator
- Limitations and Restrictions of the emulator

Purpose of the Emulator

The 70136 emulator is designed to replace the 70136 microprocessor in your target system to help you debug/integrate target system software and hardware. The emulator performs just like the processor which it replaces, but at the same time, it gives you information about the bus cycle operation of the processor. The emulator gives you control over target system execution and allows you to view or modify the contents of processor registers, target system memory, and I/O resources.



RS-232/RS-422
Connection

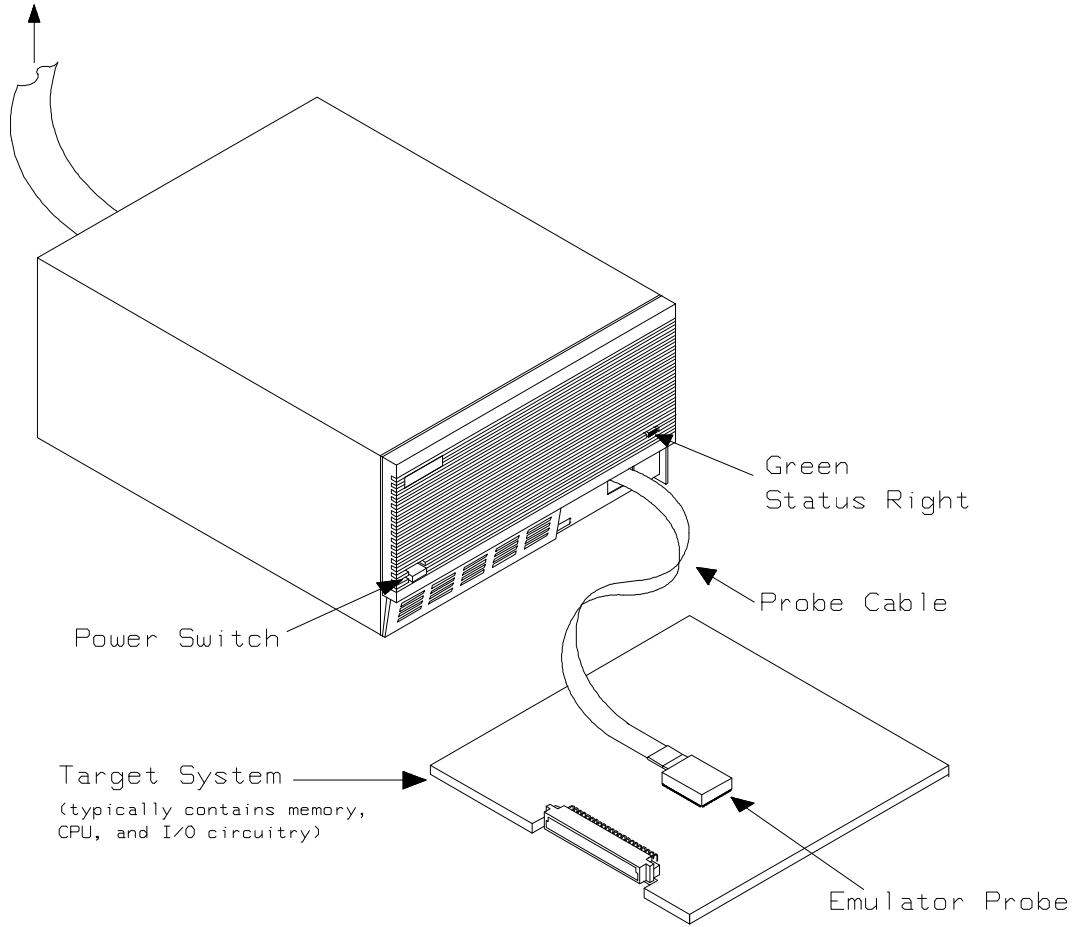


Figure 1-1. HP 64756/7 Emulator for uPD70136/70236

1-2 Introduction

Features of the 70136 Emulator

This section introduces you to the features of the emulator. The chapters which follow show you how to use these features.

Supported Microprocessors

The 70136 emulator probe has a 68-pin PLCC connector. Also provided is the adapter, HP PART No. 64756-61612, that will allow the PLCC probe to connect to the NEC EV-9200G-74 socket which replaces the 74-pin QFP package of 70136 microprocessor.

The HP 64756 emulator supports the following packages of 70136 microprocessor.

- 68-pin PLCC
- 68-pin PGA
(With using PLCC to PGA adapter; refer to the "In-Circuit Emulation Topics" chapter in this manual)
- 74-pin QFP
(With using PLCC to QFP adapter (HP PART No. 64756-61612) and NEC EV-9200G-74 socket)

The 70236 and 70236A emulator probe has an 132-pin PGA connector. Also provided is the NEC EV-9500GD-120 adapter that will allow the PGA probe to connect to the NEC EV-9200GD-120 socket which replaces the 120-pin QFP package of 70236 microprocessor.

The HP 64757 emulator supports the following packages of 70236 or 70236A microprocessor.

- 132-pin PGA
- 120-pin QFP
(With using NEC EV-9500GD-120 adapter and NEC EV-9200GD-120 socket)



Clock Speeds

The 70136 emulator runs with an internal clock speed of 16 MHz (system clock), or with target system clocks from 2-16 MHz.

The 70236 emulator runs with an internal clock speed of 16 MHz (system clock), or with target system clocks from 4-32 MHz.

The 70236A emulator runs with an internal clock speed of 16 MHz (system clock), or with target system clocks from 4-40 MHz.

Emulation memory

The HP 70136 emulator is used with one of the following Emulation Memory Cards.

- HP 64726 128K byte Emulation Memory Card
- HP 64727 512K byte Emulation Memory Card
- HP 64728 1M byte Emulation Memory Card
- HP 64729 2M byte Emulation Memory Card

You can define up to 16 memory ranges (at 256 byte boundaries and at least 256 byte in length). The monitor occupies 4K bytes leaving 124K, 508K, 1020K or 2044K bytes of emulation memory which you may use. You can characterize memory ranges as emulation RAM, emulation ROM, target system RAM, target system ROM, or guarded memory. The emulator generates an error message when accesses are made to guarded memory locations. You can also configure the emulator so that writes to memory defined as ROM cause emulator execution to break out of target program execution.

Analysis

The HP 70136 emulator is used with one of the following analyzers which allows you to trace code execution and processor activity.

- HP 64704 80-channel Emulation Bus Analyzer
- HP 64703 64-channel Emulation Bus Analyzer and 16-channel State Timing Analyzer

The HP 70236/70236A emulator is used with one of the following analyzers which allows you to trace code execution and processor activity.

- HP 64704 80-channel Emulation Bus Analyzer
- HP 64703 64-channel Emulation Bus Analyzer and 16-channel State Timing Analyzer
- HP 64794A/C/D Deep Emulation Bus Analyzer

When you use the HP 70236A emulator over 16MHz, you have to use the HP 64794 Deep Emulation Bus Analyzer.

The Emulation Bus Analyzer monitors the emulation processor using an internal analysis bus. The HP 64703 64-channel Emulation Bus Analyzer and 16-channel State/Timing Analyzer allows you to probe up to 16 different lines in your target system.



Registers You can display or modify the 70136 internal register contents.

Single-Step You can direct the emulation processor to execute a single instruction or a specified number of instructions.

Breakpoints You can set up the emulator/analyzer interaction so that when the analyzer finds a specific state, emulator execution will break to the background monitor.

You can also define software breakpoints in your program. The emulator uses one of 70136 undefined opcode (F1 hex) as software breakpoint interrupt instruction. When you define a software breakpoint, the emulator places the breakpoint interrupt instruction (F1 hex) at the specified address; after the breakpoint interrupt instruction causes emulator execution to break out of your program, the emulator replaces the original opcode.

Reset Support The emulator can be reset from the emulation system under your control, or your target system can reset the emulation processor.

Configurable Target System Interface You can configure the emulator so that it honors target system wait requests when accessing emulation memory. You can configure the emulator so that it presents cycles to, or hides cycles from, the target system when executing in background.



Foreground or Background Emulation Monitor

The emulation monitor is a program that is executed by the emulation processor. It allows the emulation controller to access target system resources. For example, when you display target system memory, it is the monitor program that executes 70136 instructions which read the target memory locations and send their contents to the emulation controller.

The monitor program can execute in *foreground*, the mode in which the emulator operates as would the target processor. The foreground monitor occupies processor address space and executes as if it were part of the target program.

The monitor program can also execute in *background*, the emulator mode in which foreground operation is suspended so that emulation processor can be used to access target system resources. The background monitor does not occupy any processor address space.

Real-Time Operation

Real-time operation signifies continuous execution of your program without interference from the emulator. (Such interference occurs when the emulator temporarily breaks to the monitor so that it can access register contents or target system memory or I/O.)

You can restrict the emulator to real-time execution. When the emulator is executing your program under the real-time restriction, commands which display/modify registers, display/modify target system memory or I/O, or single-step are not allowed.

Easy Products Upgrades

Because the HP 64700 Series development tools (emulator, analyzer, LAN board) contain programmable parts, it is possible to reprogram the firmware and some of the hardware without disassembling the HP 64700A/B Card Cage. This means that you'll be able to update product firmware, if desired, without having to call an HP field representative to your site.

Limitations, Restrictions



DMA Support

In the 70136 Emulator, Direct memory access to the emulation memory by DMA controller is not permitted.

In the 70236 and the 70236A Emulator, Direct memory access to the emulator by external DMA controller is not permitted.

User Interrupts

If you use the background monitor in the 70136 emulator, interrupts are suspended or ignored during background operation. NMI is suspended until the emulator goes into foreground operation. INT interrupt is ignored.

If you use the background monitor in the 70236 and the 70236A emulator, interrupts from target system are suspended during background operation. NMI, and INTP0-INTP7 are suspended until the emulator goes into foreground operation.

Interrupts While Executing Step Command

While executing user program code in stepping in the foreground monitor, interrupts are accepted if they are enabled in the foreground monitor program. When using the foreground monitor you will see the following error message, if the interrupts are acknowledged before stepping user program code.

```
ERROR: Stepping failed
```

Although the error message above appears, the code is executed as you expected to do.

Accessing Internal I/O Registers

When you access internal I/O registers of the emulator, you should use the "display/modify register" command with their register name instead of the "display/modify io_port" command.



PC relative addressing in trace list

When you use the following setting in your program, the branch address forming in PC relative addressing may change to a wrong value only in disassemble list.

- The program is running in the extended address mode.
- The effective address for the PC relative addressing is in the other page.
- The order of the pages is not in sequence in extended address.

"BRKXA" and "RETXA" Instructions in Stepping

When the "BRKXA" and "RETXA" instructions are executed in stepping, the emulator reads memory for disassembly after stepping. When you execute "BRKXA" instruction in stepping, the normal address where the "BRKXA" instruction is located is extended to read memory for disassemble after stepping. When you execute "RETXA" instruction in stepping, the normal address which is extended to point the "RETXA" instruction is not extended to read memory for disassemble after stepping.

Stepping at Software Breakpoint

When you execute step commands in the foreground monitor, you should not step at the address which the "Software Breakpoint" was set; the stepping will be failed.

```
ERROR: Stepping failed
```

Evaluation Chip

Hewlett-Packard makes no warranty of the problem caused by the 70136/70236/70236A Evaluation chip in the emulator.

Getting Started

Introduction

This chapter leads you through a basic tutorial that shows how to use the 70136 emulator with the PC Interface.

This chapter will:

- Tell you what to do before you use the emulator in the tutorial.
- Describe the sample program used for this chapter's examples.
- Briefly describe how to enter PC Interface commands and how emulator status is displayed.

This chapter will show you how to:

- Start up the PC Interface from the MS-DOS prompt.
- Define (map) emulation and target system memory.
- Load programs into emulation and target system memory.
- Enter emulation commands to view sample program execution.

Before You Begin

Prerequisites

Before beginning the tutorial presented in this chapter, you must have completed the following tasks:

1. Connected the emulator to your computer. The *HP 64700 Series Installation/Service* manual shows you how to do this.
2. Installed the PC Interface software on your computer. Software installation instructions are shipped with the media containing the PC Interface software. The *HP 64700 Emulators PC Interface: User's Reference* manual contains additional information on the installation and setup of the PC Interface.
3. In addition, it is recommended, although not required, that you read and understand the concepts of emulation presented in the *Concepts of Emulation and Analysis* manual. The *Installation/Service* also covers HP 64700 Series system architecture. A brief understanding of these concepts may help avoid questions later.

The sample program

The sample program used in this chapter is listed in figure 2-1. The program emulates a primitive command interpreter.

programs;for getting started

We will show you how to use the emulator to:

- load this program into emulation memory
- execute the program
- monitor the program's operation with the analyzer
- simulate entry of different commands using the **Memory Modify** emulation command.

LOCATION	OBJECT	CODE	LINE	SOURCE	LINE
			1	"70116"	
			2		GLB
			3		Msgs, Init, Cmd_Input, Msg_Dest
			4		DATA
0000			5	Msgs	
0000	436F6D6D61		6	Msg_A	DB
0005	6E64204120				"Command A entered "
000A	656E746572				
000F	656420				
0012	436F6D6D61		7	Msg_B	DB
0017	6E64204220				"Command B entered "
001C	656E746572				
0021	656420				
0024	496E76616C		8	Msg_I	DB
0029	696420436F				"Invalid Command "
002E	6D6D616E64				
0033	202020				
0036			9	End_Msgs	
			10		
			11		PROG
			12		ASSUME DS0:DATA,DS1:COMN
			13	*****	
			14	* The following instructions initialize segment	
			15	* registers and set up the stack pointer.	
			16	*****	
0000	B80000		17	Init	MOV AW,SEG Msg_A
0003	8ED8		18		MOV DS0,AW
0005	B80000		19		MOV AW,SEG Cmd_Input
0008	8EC0		20		MOV DS1,AW
000A	8ED0		21		MOV SS,AW
000C	BC00F9		22		MOV SP,OFFSET Stk
			23	*****	
			24	* Clear previous command	
			25	*****	
000F	26C6060000		26	Rrad_Cmd	MOV Cmd_Input,#0
0014	0090				
			27	*****	
			28	* Read command input byte. If no command has been	
			29	* entered, continue to scan for command input.	
			30	*****	
0016	26A00000		31	Scan	MOV AL,Cmd_Input
001A	3C00		32		CMP AL,#0
001C	74F8		33		BE Scan
			34	*****	
			35	* A command has been entered. Check if it is	
			36	* command A, command B, or invalid.	
			37	*****	
001E	3C41		38	Exe_Cmd	CMP AL,#41H
0020	7407		39		BE Cmd_A
0022	3C42		40		CMP AL,#42H
0024	740C		41		BE Cmd_B
0026	E91200		42		BR Cmd_I
			43	*****	
			44	* Command A is entered. CW = the number of bytes in	
			45	* message A. BP = location of the message. Jump to	

Figure 2-1. Sample Program Listing

```

46 * the routine which writes the message.
47 *****
0029 B91200 48 Cmd_A      MOV     CW,#Msg_B-Msg_A
002C BE0000 49           MOV     IX,OFFSET Msg_A
002F E90F00 50           BR      Write_Msg
51 *****
52 * Command B is entered.
53 *****
0032 B91200 54 Cmd_B      MOV     CW,#Msg_I-Msg_B
0035 BE0012 55           MOV     IX,OFFSET Msg_B
0038 E90600 56           BR      Write_Msg
57 *****
58 * An invalid command is entered.
59 *****
003B B91200 60 Cmd_I      MOV     CW,#End_Msgs-Msg_I
003E BE0024 61           MOV     IX,OFFSET Msg_I
62 *****
63 * Message is written to the destination.
64 *****
0041 BF0001 65 Write_MSG  MOV     IY,OFFSET Msg_Dest
0044 F3A4    66           REP MOVKB
67 *****
68 * The rest of the destination area is filled
69 * with zeros.
70 *****
0046 C60500 71 Fill_Dest  MOV     BYTE PTR [IY],#0
0049 47      72           INC     IY
004A 81FF0021 73           CMP     IY,#Msg_Dest+20H
004E 75F6    74           BNE     Fill_Dest
75 *****
76 * Go back and scan for next command
77 *****
0050 EBBD    78           BR      Read_Cmd
79
80           COMN
81 *****
82 * Command input byte.
83 *****
0000      84 Cmd_Input  DBS     1
85 *****
86 * Destination of the command message.
87 *****
0001      88 Msg_Dest   DDS     3EH
00F9      89 Stk       DWS     1      ; Stack area.
          90           END     Init
          <0000>

```

Figur 2-1. Sample Program Listing (Cont'd)

Data Declarations

The "DATA" section defines the messages used by the program to respond to various command inputs. These messages are labeled **Msg_A**, **Msg_B**, and **Msg_I**.

Initialization

The program instructions from the **Init** label to the **Read_Cmd** label perform initialization. The segment registers are loaded and the stack pointer is set up.

Reading Input

The instruction at the **Read_Cmd** label clears any random data or previous commands from the **Cmd_Input** byte. The **Scan** loop continually reads the **Cmd_Input** byte to look for a command (a value other than 0 hex).



Processing Commands

When a command is entered, the instructions from **Exe_Cmd** to **Cmd_A** decide whether the command was "A", "B", or an invalid command.

If the command input byte is "A" (ASCII 41 hex), execution transfers to the instructions at **Cmd_A**.

If the command input byte is "B" (ASCII 42 hex), execution transfers to the instructions at **Cmd_B**.

If the command input byte is neither "A" nor "B", an invalid command was entered, and execution transfers to the instructions at **Cmd_I**.

The instructions at **Cmd_A**, **Cmd_B**, and **Cmd_I** each load register CW with the displayed message's length and register IX with the message's starting location. Then, execution transfers to **Write_Msg**, which writes the appropriate message to the destination location, **Msg_Dest**.

After the message is written, the instructions at **Fill_Dest** fill the remaining destination locations with zeros. (The entire destination area is 20 hex bytes long.) Then, the program jumps back to read the next command.

The Destination Area

The "COMN" section declares memory storage for the command input byte, the destination area, and the stack area.



Assembling and Linking the Sample Program

The sample program is written for the HP 64853 Cross Assembler/Linker.

Use the following command to assemble and link the sample program.

```
C> asm -oe cmd_rds.s > cmd_rds.o <RETURN>
```

```
C> lnk -o > cmd_rds.m <RETURN>
```

```
object files cmd_rds.R <RETURN>
```

```
library files <RETURN>
```

```
Load addresses: PROG,DATA,COMN 400H,600H,800H <RETURN>
```

```
more files (y or n) N <RETURN>
```

```
absolute file name cmd_rds.X <RETURN>
```

Starting Up the 70136 PC Interface

If you built the emulator device table and set the **HPTABLES** shell environment variable as shown in the *HP 64700 Emulators PC Interface: User's Reference*, you can start up the 70136 PC Interface by entering the following command from the MS-DOS prompt:

```
C> pcv33 <emulname>
```

where <emulname> is **emul_com1** if your emulator is connected to the COM1 port or **emul_com2** if it is connected to the COM2 port. If you edited the \hp64700\tables\64700tab file to change the emulator name, substitute the appropriate name for <emulname> in the above command.

In the command above, **pcv33** is the command to start the PC Interface; "<emulname>" is the logical emulator name given in the emulator device table. (To start the version of the PC Interface that supports external timing analysis, substitute **ptv33** for **pcv33** in this command.) If this command is successful, you will see the display shown in figure 2-2. Otherwise, you will see an error message and return to the MS-DOS prompt.

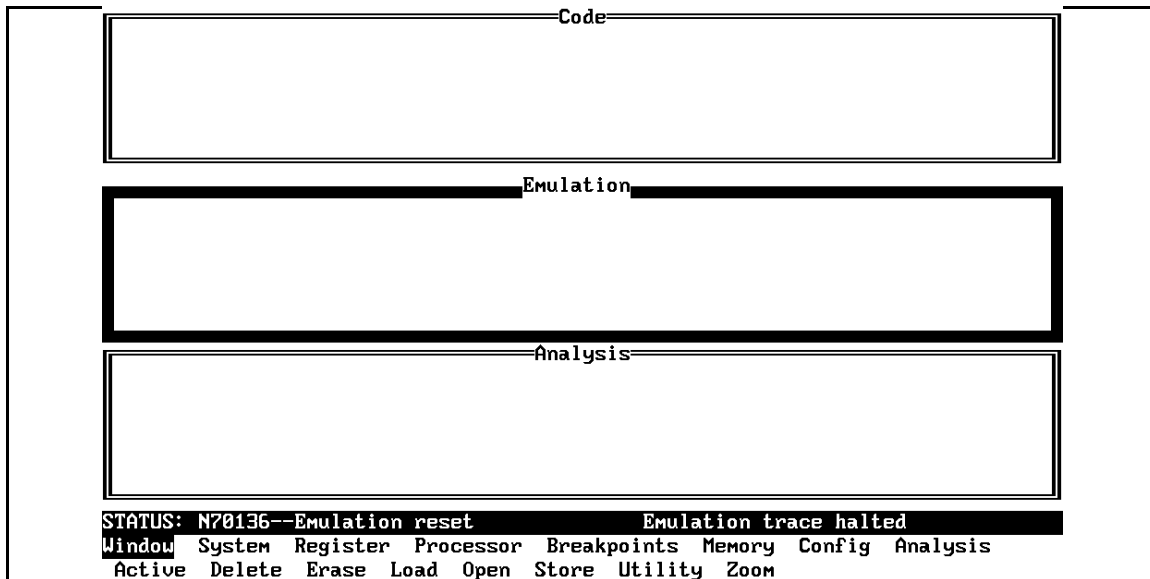


Figure 2-2. PC Interface Display

Selecting PC Interface Commands

This manual will tell you to "select" commands. You can select commands or command options by using the left and right arrow keys to highlight the option. Then press the **Enter** key. Or, you can simply type the first letter of that option. If you select the wrong option, press the **ESC** key to retrace the command tree.

When a command or option is highlighted, the bottom line of the display shows the next level of options or a short message describing the current option.

Emulator Status

The emulator status is shown on the line above the command options. The PC Interface periodically checks the status of the emulator and updates the status line.

Mapping Memory

The 70136 emulator contains high-speed emulation memory (no wait states required) that can be mapped at a resolution of 256 bytes.

Note



When you use the NEC uPD72291 coprocessor on your target system connected to 70136 microprocessor, the uPD72291 can access 70136 emulation memory on coprocessor memory read/write cycles. In this case, you should reset the target system to connect the 70136 emulator to the uPD72291 coprocessor before starting emulation session.

Refer to "In-Circuit Emulation Topics" chapter for more information about accesses to emulation memory.

The memory mapper allows you to characterize memory locations. It allows you specify whether a certain range of memory is present in the target system or whether you will be using emulation memory for that address range. You can also specify whether the target system memory is ROM or RAM, and you can specify that emulation memory be treated as ROM or RAM.

Note



Target system accesses to emulation memory are not allowed.

Target system devices that take control of the bus (for example, DMA controllers) cannot access emulation memory.

Blocks of memory can also be characterized as guarded memory. Guarded memory accesses will generate "break to monitor" requests. Writes to ROM will generate "break to monitor" requests if the "Enable breaks on writes to ROM?" configuration item is enabled (see the "Configuring the Emulator" chapter). The memory mapper allows you to define up to 16 different map terms.

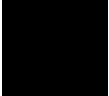
Which Memory Locations Should Be Mapped?

Typically, assemblers generate relocatable files and linkers combine relocatable files to form the absolute file. The linker load map listing will show what locations your program will occupy in memory. For example, the HP 64853 linker load map listing for the sample program is shown in figure 2-3.

HP 64000+ Linker

FILE/PROG NAME	PROGRAM	DATA	COMMON	ABSOLUTE
CMD_RDS.R	00000400	00000600	00000800	
next address	00000452	00000636	000008FB	
XFER address = 00000400 Defined by CMD_RDS.R				
Absolute file name = CMD_RDS.X				
Total number of bytes loaded = 00000183				

Figure 2-3. Sample Program Load Map Listing



From the load map listing, you can see that the sample program occupies locations in three address ranges. The program area, which contains the opcodes and operands which make up the sample program, occupies locations 400 hex through 451 hex. The data area, which contains the ASCII values of the messages the program displays, is occupies locations 600 hex through 635 hex. The destination area, which contains the command input byte and the locations of the message destination and the stack, occupies locations 800 hex through 8FA hex.

Two mapper terms will be specified for the example program. Since the program writes to the destination locations, the mapper block containing the destination locations should not be characterized as ROM memory.

To map memory for the sample program, select:

Config, **M**ap, **M**odify

Using the arrow keys, move the cursor to the "address range" field of term 1. Enter:

0..07ff@e

Notice that "@e" must be added in mapping memory; "@e" is the function code to define as an extended address. Refer to the "Address Expression" section in "Using the Emulator" chapter.

Move the cursor to the "memory type" field of term 1, and press the TAB key to select the **erom** (emulation ROM) type. Move the cursor to the "address range" field of term 2 and enter:

0800..09ff@e

Move the cursor to the "memory type" field of term 2, and press the TAB key to select the **eram** (emulation ROM) type. To save your memory map, use the right arrow key or the **Enter** key to exit the field in the lower right corner. (The **End** key on Vectra keyboards moves the cursor directly to the last field.) The memory configuration display is shown in figure 2-4.

Loading Programs into Memory

If you have already assembled and linked the sample program, you can load the absolute file by selecting:

`Memory, Load`

File Format

Use **Tab** and **Shift-Tab** to select the format of your absolute file. The emulator accepts absolute files in the following formats:

- Intel OMF86 absolute.
- NEC30 absolute.
 - (This absolute file is generated by NEC LK70136 linker for uPD70136.)
- NEC33 absolute.
 - (This absolute file is the extended load module format file which is generated by NEC EL70136 extended mode locator for uPD70136.)
- HP64000 absolute.
- Raw HP64000 absolute.
- Intel hexadecimal.
- Motorola S-records.
- Tektronix hexadecimal.

For this tutorial, choose the HP64000 format.

Target Memory Type for Memory Load

The second field allows you to selectively load the portions of the absolute file which reside in emulation memory, target system memory, both emulation and target system memory.

Since emulation memory is mapped for sample program locations, you can select either "emulation" or "both". Use **Tab** key and **Shift-Tab** to cycle through the choices.

Force the Absolute File to Be Read

This option is only available for the Intel OMF86, NEC30, NEC33, and HP64000 absolute file formats.

It forces the file format reader to regenerate the emulator absolute file (.hpa) and symbol database (.hps) before loading the code. Normally, these files are only regenerated whenever the file you specify (the output of your language tools) is newer than the emulator absolute file and symbol database.

For more information, refer to the File Format Readers appendix.

File Format Options

Some of the formats, such as the Intel OMF86 format, have special options.

Refer to the File Format Readers appendix of this manual for more information.

Absolute File Name

For most formats, you enter the name of your absolute file in the last field. The HP64000 format requires the linker symbol filename instead. Type **cmd_rds.L**, and press **Enter** to start the memory load.

```
Memory Load Configuration

File Format          HP64000
Target memory type for memory load  Both
Force the absolute file to be read  no

Absolute file name
cmd_rds.L

<↑↓> : Interfield movement   Ctrl <←> : Field editing   TAB : Scroll choices

STATUS: N70136--Emulation reset      Emulation trace halted

Enter the name of an HP64000 linker symbol file (ex. test.L).
```

Displaying Symbols

Symbol files are created when you generate absolute files with the HP 64000-PC Cross Assembler/Linkers. When you assemble a source file, an assembler symbol file (with the same base name as the source file and a .a extension) is created. The assembler symbol file contains local symbol information. When you link relocatable assembly modules, a linker symbol file (with the same base name as the absolute file and a .l extension) is created. The linker symbol file contains global symbol information and information about the relocatable assembly modules that combine to form the absolute file.

When you load a file using the HP64000 file format, the file format reader collects global symbol information from the linker symbol file and local symbol information from the assembler symbol files. It uses this information to create a single symbol database with the extension .hps.

If you load a file using the following formats, the file format reader obtains all the global and local symbol information from the absolute file and builds a symbol database with extension .hps.

- Intel OMF86 absolute.
- NEC30 absolute.
- NEC33 absolute.

The following pages show you how to display global and local symbols for the sample program. For more information on symbol display, refer to the *PC Interface Reference*.

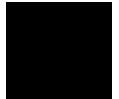
Displaying Global Symbols

When you load a file using the following formats into the emulator, the corresponding symbol database is also loaded.

- Intel OMF86 absolute.
- NEC30 absolute.
- NEC33 absolute.
- HP64000 absolute.

The symbol database also can be loaded with the **System, Symbols, Global, Load** command. Use this command when you load multiple absolute files into the emulator. You can load the various symbol databases corresponding to each absolute file. When you load a symbol database, information from a previous symbol database is lost. That is, only one symbol database can be present at a time.

After a symbol database is loaded, both global and local symbols can be used when entering expressions. You enter global symbols as they appear in the source file or in the global symbols display.



To display global symbols, select:

System **S**ymbols **G**lobal **D**isplay

The symbols window automatically becomes the active window because of this command. You can press <CTRL>**z** to zoom the window. The resulting display follows.

```

                                     Symbols
-----
Modules
-----
CMD_RDS.S

Address      Symbol
-----
00000:00800  Cmd_Input
00000:00400  Init
00000:00801  Msg_Dest
00000:00600  Msgs

STATUS: N70136--Emulation reset           Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Command_file Wait MS-DOS Log Terminal Symbols Exit
```

The global symbols display has two parts. The first part lists all the modules that were linked to produce this object file. These module names are used by you when you want to refer to a local symbol, and are case-sensitive. The second part of the display lists all global symbols in this module. These names can be used in measurement specifications, and are case-sensitive. For example, if you wish to make a measurement using the symbol **Cmd_Input**, you must specify **Cmd_Input**.

The strings **cmd_input** and **CMD_INPUT** are not valid symbol names here.

Loading and Displaying Local Symbols

To display local symbols, select:

System Symbols Local Display

Enter the name of the module you want to display (from the first part of the global symbols list; in this case, **CMD_RDS.S**) and press **Enter**. The resulting display follows.

```

                                     Symbols
-----
Address      Symbol
-----
00000:00429  Cmd_A
00000:00432  Cmd_B
00000:0043B  Cmd_I
00000:00800  Cmd_Input
00000:00636  End_Msgs
00000:0041E  Exe_Cmd
00000:00446  Fill_Dest
00000:00400  Init
00000:00600  Msg_A
00000:00612  Msg_B
00000:00801  Msg_Dest
00000:00624  Msg_I
00000:00600  Msgs
00000:0040F  Read_Cmd
00000:00416  Scan
00000:008F9  Stk
00000:00441  Write_Msg

STATUS: N70136--Emulation reset           Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Command_file Wait MS-DOS Log Terminal Symbols Exit
```

After you display local symbols with the **System Symbols Local Display** command, you can enter local symbols as they appear in the source file or local symbol display. When you display local symbols for a given module, that module becomes the default local symbol module.

If you have not displayed local symbols, you can still enter a local symbol by including the name of the module:

```
module_name:symbol
```

Remember that the only valid module names are those listed in the first part of the global symbols display, and are case-sensitive for compatibility with other systems (such as HP-UX).

When you include the name of a source file with a local symbol, that module becomes the default local symbol module, as with the **System Symbols Local Display** command.

Local symbols must be from assembly modules that form the absolute whose symbol database is currently loaded. Otherwise, no symbols will be found (even if the named assembler symbol file exists and contains information).

One thing to note: It is possible for a symbol to be local in one module and global in another, which may result in some confusion. For example, suppose symbol XYZ is a global in module A and a local in module B and that these modules link to form the absolute file. After you load the absolute file (and the corresponding symbol database), entering XYZ in an expression refers to the symbol from module A. Then, if you display local symbols from module B, entering XYZ in an expression refers to the symbol from module B, **not the global symbol**. Now, if you again want to enter XYZ to refer to the global symbol from module A, you must display the local symbols from module A (since the global symbol is also local to that module). Loading local symbols from a third module, if it was linked with modules A and B and did not contain an XYZ local symbol, would also cause XYZ to refer to the global symbol from module A.

Transfer Symbols to the Emulator

You can use the emulator's symbol-handling capability to improve measurement displays. You do this by transferring the symbol database information to the emulator. To transfer the global symbol information to the emulator, use the command:

```
System Symbols Global Transfer
```

Transfer the local symbol information for all modules by entering:

```
System Symbols Local Transfer All
```

You can find more information on emulator symbol handling commands in the *Emulator PC Interface Reference*.

PGR register

You can configure the 70136 emulator to break to the monitor to read the current value of page registers when the emulation system needs to convert normal address to extended address.

However, the normal address mode is only used in this sample program.

You should change the configuration not to break to the monitor to read page registers in the general emulator configuration. Select:

```
Config, General
```

Use the arrow keys to move the cursor to the "Read PGR register" field, answer [n], press **End** to move to the lower right corner, and press **Enter** to exit the general emulator configuration.

See the "Configuring the Emulator" chapter for a complete description of the emulator configuration.

Displaying Memory in Mnemonic Format

Once you have loaded a program into the emulator, you can verify that the program has indeed been loaded by displaying memory in mnemonic format. To do this, select:

Memory, Display, Mnemonic

Enter the address range "400H..429H". You could also specify this address range using symbols.

For example,
"Init..Cmd_A" or "Init..Init+29H".

The Emulation window remains active. You can press <CTRL>z to zoom the memory window. The resulting display follows.

If you want to see the rest of the sample program memory locations, you can select "Memory, Display, Mnemonic" command and enter the range from 42AH to 451H.

```
Emulation
Address      Symbol      Mnemonic
-----
00000:00400  Init        MOU AW, #0000
00000:00403  -           MOU DS0, AW | MOU AW, #0000
00000:00408  -           MOU DS1, AW | MOU SS, AW | MOU SP, #08f
00000:0040f  _RDS.S:Read_Cmd  MOU DS1:0800, #00
00000:00415  -           NOP
00000:00416  CMD_RDS.S:Scan  MOU AL, DS1:0800
00000:0041a  -           CMP AL, #00
00000:0041c  -           BE/Z CMD_RDS.S:Scan
00000:0041e  D_RDS.S:Exe_Cmd  CMP AL, #41
00000:00420  -           BE/Z CMD_RDS.S:Cmd_A
00000:00422  -           CMP AL, #42
00000:00424  -           BE/Z CMD_RDS.S:Cmd_B
00000:00426  -           BR NEAR PTR CMD_RDS.S:Cmd_I
00000:00429  CMD_RDS.S:Cmd_A  MOU CW, #0012

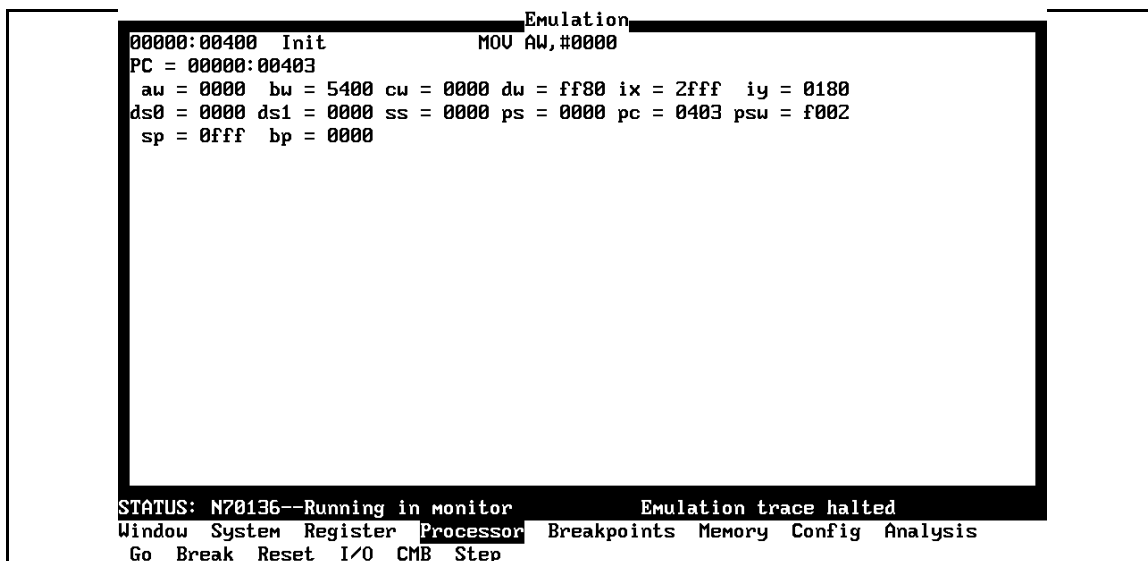
STATUS: N70136--Emulation reset           Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Display Modify Load Store Copy Find Report
```

Stepping Through the Program

The emulator allows you to execute one instruction or a number of instructions with step command. To begin stepping through the sample program, select:

`Processor, Step, Address`

Enter a step count of 1, enter the symbol **Init** (defined as a global in the source file), and press **Enter** to step from program's first address, 400H. The Emulation window remains active. Press <CTRL>**z** to view a full screen of information. The executed instruction, the program counter address (PS:PC), and the resulting register contents are displayed as shown in the following.



```
Emulation
00000:00400  Init          MOU AW,#0000
PC = 00000:00403
aw = 0000  bw = 5400  cw = 0000  dw = ff80  ix = 2fff  iy = 0180
ds0 = 0000  ds1 = 0000  ss = 0000  ps = 0000  pc = 0403  psw = f002
sp = 0fff  bp = 0000

STATUS: N70136--Running in monitor      Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Go Break Reset I/O CMB Step
```

Note



You cannot display registers if the processor is reset. Use the "Processor **Break**" command to cause the emulator to start executing in the monitor.

You can display registers while the emulator is executing a user program (if execution is not restricted to real-time); emulator execution will temporarily break to the monitor.

Note



There are a few cases in which the emulator can not step. Step command is not accepted between each of the following instructions and the next instruction.

- 1) Manipulation instructions for sreg :
MOV sreg,reg16; MOV sreg,mem16; POP sreg.
 - 2) Prefix instructions: PS:, SS:, DS0:, DS1:, REPC, REPNC, REP, REPE, REPZ, REPNE, REPNZ, BUSLOCK.
 - 3) EI, RETI, DI.
-

Note



You cannot use over 100000 hex address in "Processor Step" command.

To continue stepping through the program, you can select:

`Processor, Step, Pc`

After selecting this command, you can change the previous step count. If you wish to step the same number of times, just press **Enter** to start the step.

To save time when single-stepping, you can use the function key macro <F1>, which executes the command:

`Processor Step Pc 1`

For more information, see the *Emulator PC Interface Reference* chapter on Function Key Macros.

To repeat the previous command, you can press <CTRL>**r**.

Specifying a Step Count

If you want to step several times from the current program counter, select:

Processor, Step, Pc

The previous step count is displayed in the "number of instructions" field. You can enter a number from 1 through 99 to specify the number of times to step. Type 5 into the field, and press **Enter**. The resulting display follows.

When you specify step counts greater than 1, only the last instruction and the register contents after that instruction are displayed.

```
Emulation
00000:00403 -          MOU DS0,AW ! MOU AW,#0000
PC = 00000:00408
aw = 0000  bw = 5400  cw = 0000  dw = ff80  ix = 2fff  iy = 0180
ds0 = 0000  ds1 = 0000  ss = 0000  ps = 0000  pc = 0408  psw = f002
sp = 0fff  bp = 0000

00000:00408 -          MOU DS1,AW ! MOU SS,AW ! MOU SP,#08f
00000:0040f _RDS.S:Read_Cmd MOU DS1:0800,#00
00000:00415 -          NOP
00000:00416 CMD_RDS.S:Scan MOU AL,DS1:0800
00000:0041a -          CMP AL,#00
PC = 00000:0041c
aw = 0000  bw = 5400  cw = 0000  dw = ff80  ix = 2fff  iy = 0180
ds0 = 0000  ds1 = 0000  ss = 0000  ps = 0000  pc = 041c  psw = f046
sp = 08f9  bp = 0000

STATUS: N70136--Running in monitor          Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Go Break Reset I/O CMB Step
```

Modifying Memory

The preceding step commands show the sample program is executing in the **Scan** loop, where it continually reads the command input byte to look for a command.

To simulate the entry of a sample program command, you can modify the command input byte by selecting:

Memory, Modify, Byte

Now enter the address of the memory location to be modified, an equal sign, and new value of that location, for example, **Cmd_Input="A"**. (The **Cmd_Input** label was defined as a global symbol in the source file.)

To verify that "A" was indeed written to **Cmd_Input** (800 hex), select:

Memory, Display, Byte

Type the address 800H or the symbol **Cmd_Input**, and press **Enter**. The resulting display is shown below.

The screenshot displays a debugger interface with three main sections: Symbols, Emulation, and Analysis. The Symbols section shows a list of symbols: Cmd_A at address 00000:00429, Cmd_B at 00000:00432, and Cmd_I at 00000:0043B. The Emulation section shows a memory location at address 00000:00800 containing the hex value 41, which corresponds to the ASCII character 'A'. The Analysis section is currently empty. At the bottom, a status bar indicates 'STATUS: N70136--Running in monitor Emulation trace halted' and provides navigation options: Window System Register Processor Breakpoints Memory Config Analysis, Go Break Reset I/O CMB Step.

Symbols	
Address	Symbol
00000:00429	Cmd_A
00000:00432	Cmd_B
00000:0043B	Cmd_I

Emulation		
Address	Data (hex)	Ascii
00000:00800	41	A

Analysis	

STATUS: N70136--Running in monitor Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Go Break Reset I/O CMB Step

You can continue to step through the program as shown earlier in this chapter to view the instructions which are executed when an "A" (41 hex) command is entered.

Running the Program

To start the sample program, select:

```
Processor, Go, Pc
```

The status line will show that the emulator is "Running user program".

Note



You can not use over 100000 hex address in "Processor Go" command.

Searching Memory for Data

You can search the message destination locations to verify that the sample program writes the appropriate messages for the allowed commands. The command "A" (41 hex) was entered above, so the "Command A entered " message should have been written to the **Msg_Dest** locations. Because you must search for hexadecimal values, you will want to search for a sequence of characters which uniquely identify the message, for example, " A " or 20 hex, 41 hex, and 20 hex. To search the destination memory location for this sequence of characters, select:

```
Memory, Find
```

Enter the range of the memory locations to be searched, "800H..820H", and enter the data " A " or 20H, 41H, and 20H. The resulting information in the Emulation window shows you that the message write occurred correctly. The message is:

```
Pattern match at address: 0000808@p
```

To verify that the sample program works for the other allowed commands, you can modify the command input byte to "B" and search for " B " (20 hex, 42 hex, and 20 hex), or you can modify the command input byte to "C" and search for "d C" (64 hex, 20 hex, and 43 hex).

Breaking into the Monitor

To break emulator execution from the sample program to the monitor program, select:

`Processor, Break`

The status line shows that the emulator is "Running in monitor".

While the break will occur as soon as possible, the actual stopping point may be many cycles after the break request. This depends on the type of instruction being executed, and whether the processor is in a hold state.

Using Software Breakpoints

Software breakpoints are provided with one of 70136 undefined opcode (F1 hex) as breakpoint interrupt instruction.

When you define or enable a software breakpoint, the emulator will replace the opcode at the software breakpoint address with the breakpoint interrupt instruction.

Caution



Software breakpoints should not be set, cleared, enabled, or disabled while the emulator is running user code. If any of these commands are entered while the emulator is running user code, and the emulator is executing code in the area where the breakpoint is being modified, program execution may be unreliable.

Caution

When you use extended address mode, care should be taken for software breakpoints. If you change the relation between the physical address and the extended address after you set a software breakpoint (ex. change address mode or change the contents of the page register), emulation system may not recognize the software breakpoint.

In this case, the breakpoint interrupt instruction (F1 hex) is left in memory and the software break will not occur at the specified address. When you set a software breakpoint with using symbols, you also should not change the relation between the physical address and the extended address after setting a software breakpoint.



Note

You must only set software breakpoints at memory locations which contain instruction opcodes (not operands or data). If a software breakpoint is set at a memory location which is not an instruction opcode, the software breakpoint instruction will never be executed and the break will never occur.

Note

NMI will be ignored, when software breakpoint and NMI occur at the same time.

Note

Because software breakpoints are implemented by replacing opcodes with the breakpoint interrupt instruction, you cannot define software breakpoints in target ROM. You can use the Terminal Interface **cim** command to copy target ROM into emulation memory (see the *Terminal Interface Reference* manual for information on the **cim** command).

Note

Do not set, clear, enable or disable software breakpoints while the emulator is running user code. If you enter any of these commands while the emulator is executing user code in the area of the breakpoint you are modifying, program execution may be unreliable.

Note

Software breakpoint will be ignored, when software breakpoint and other emulation break (for example, break command, trigger command, etc.) occur at the same time. Refer to *PC Interface: User's Reference* manual.

When software breakpoints are enabled and emulator detects the breakpoint interrupt instruction (F1 hex), it generates a break to background request which as with the "processor break" command. Since the system controller knows the locations of defined software breakpoints, it can determine whether the breakpoint interrupt instruction (F1 hex) is a software breakpoint or opcode in your target program.

If it is a software breakpoint, execution breaks to the monitor, and the breakpoint interrupt instruction is replaced by the original opcode. A subsequent run or step command will execute from this address.

If it is an opcode of your target program, execution still breaks to the monitor, and an "Undefined software breakpoint" status message is displayed.

When software breakpoints are disabled, the emulator replaces the breakpoint interrupt instruction with the original opcode. Up to 32 software breakpoints may be defined.

Defining a Software Breakpoint

To define a breakpoint at the address of the `Cmd_I` label of the sample program (43B hex), select:

```
Breakpoints, Add
```

Enter the local symbol "Cmd_I". After the breakpoint is added, the Emulation window becomes active and shows that the breakpoint is set.

You can add multiple breakpoints in a single command by separating them with a semicolon. For example, you could type "`2010h;2018h;2052h`" to set three breakpoints.

Run the program by selecting:

```
Processor, Go, Pc
```

The status line shows that the emulator is running the user program. Modify the command input byte to an invalid command by selecting:

```
Memory, Modify, Byte
```

Enter an invalid command, such as "`Cmd_Input=75h`". The following messages result:

```
ALERT: Software breakpoint: 00000:0043b  
STATUS: Running in monitor
```

To continue program execution, select:

```
Processor, Go, Pc
```

Displaying Software Breakpoints

To view the status of the breakpoint, select:

```
Breakpoints, Display
```

The display shows that the breakpoint was cleared.

```

Emulation
-----
Status  Address
-----
Clear   00000:0043b

STATUS: N70136--Running in monitor      Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Display Add Remove Set Clear

```

Setting a Software Breakpoint

A breakpoint is disabled when it is hit. To re-enable the software breakpoint, you can select:

`Breakpoints, Set, Single`

The address of the breakpoint you just added is still in the address field. To set this breakpoint again, press **Enter**.

As with the "Breakpoints Add" command, the Emulation window becomes active and shows that the breakpoint is set.

Clearing a Software Breakpoint

If you wish to clear a software breakpoint that does not get hit during program execution, you can select:

`Breakpoints, Clear, Single`

The address of the breakpoint set in the previous section is still in the address field. To clear this breakpoint, press **Enter**.

Using the Analyzer

The analyzer collects data at each pulse of a clock signal, and saves the data (a trace state) if it meets a "storage qualification" condition.

Note



Emulators which have the optional external analyzer will display the "**Internal/External**" option after commands in the following examples. Select **Internal** to execute the example commands.

Resetting the Analysis Specification

To be sure that the analyzer is in its default or power-up state, select:

`Analysis, Trace, Reset`

Specifying a Simple Trigger

Suppose you wish to trace the states of the sample program which follow the read of a "B" (42 hex) command from the command input byte. To do this, you must modify the default analysis specification by selecting:

`Analysis, Trace, Modify`

The emulation analysis specification is shown. Use the right arrow key to move the "Trigger on" field. Type "a" and press **Enter**.

You'll enter the pattern expression menu. Press the up arrow key until the **addr** field directly opposite the pattern **a=** is highlighted. Type the address of the command input byte, using either the global symbol **Cmd_Input** or address 800H, and press **Enter**.

The Data field is now highlighted. Type 0XX42 and press **Enter**. "42" is the hexadecimal value of the B command and the "X"s specify "don't care" values. When 42H is read from the command input byte (800H), a lower byte read is performed because the address is even. If the address is odd, you must specify the data to 42XX.

Now the Status field is highlighted. Use the TAB key to view the status qualifier choices.

Qualifier	Status Bits	Description
bs16	0xx xx1x xxxx xxxxB	Bus size 16
bs8	0xx xx0x xxxx xxxxB	Bus size 16
coproc	0x0 xxxx x101 x0xxB	Co-processor access
cprd	0x0 xxxx x101 x01xB	Co-processor read
cpwr	0x0 xxxx x101 x00xB	Co-processor write
exec	0x0 xxxx x0xx xxxxB	Executed code
extaddr	0xx 1xxx xxxx xxxxB	Extended address mode
extmemrd	0x0 1xxx 1110 x11xB	memory read in extended address mode
extmemwr	0x0 1xxx 1110 x10xB	memory write in extended address mode
fetch	0x0 xxxx 1100 x11xB	Program fetch
grdacc	0xx xxxx 0xxx x1xxB	Guarded access
haltack	0x0 xxxx x111 x00xB	Halt acknowledge
holdack	0x1 xxxx xxxx xxxxB	Hold acknowledge
intack	0x0 xxxx x100 x01xB	Interrupt acknowledge
io	0x0 xxxx x110 x0xxB	I/O access
ioread	0x0 xxxx x110 x01xB	I/O read
iowrite	0x0 xxxx x110 x00xB	I/O write
memory	0x0 xxxx 1110 x1xxB	memory access
memforcp	0x0 xxxx 1101 x1xxB	memory access for cp
memread	0x0 xxxx 1110 x11xB	memory read
memrdcp	0x0 xxxx 1101 x11xB	memory read for cp
memwrite	0x0 xxxx 1110 x10xB	memory write
memwrpc	0x0 xxxx 1101 x10xB	memory write for cp
mon	0xx x0xx xxxx xxxxB	monitor cycle
nmladdr	0xx 0xxx xxxx xxxxB	normal address mode
nmlmemrd	0x0 0xxx 1110 x11xB	memory read in normal address mode
nmlmemwr	0x0 0xxx 1110 x10xB	memory write in normal address mode
read	0x0 xxxx x1xx xx1xB	read cycle
write	0x0 xxxx x1xx xx0xB	write cycle
wrrom	0xx xxx0 xxxx xx0xB	write to ROM

70136 Analysis Status Qualifiers

This trace command example uses the status qualifier "read". The following analysis status qualifiers also can be used with the 70136 emulator.

Qualifier	Status Bits	Description
bs16	1xxx xx1x xxxx xxxxB	bus size 16
bs8	1xxx xx0x xxxx xxxxB	bus size 8
coproc	1xx0 xxxx x101 00xxB	co-processor access
cprd	1xx0 xxxx x101 001xB	co-processor read
cpwr	1xx0 xxxx x101 000xB	co-processor write
dma_mem	1xx0 xxxx x110 11xxB	DMA cycle
dma_memrd	1xx0 xxxx x110 111xB	DMA read cycle
dma_memwr	1xx0 xxxx x110 110xB	DMA write cycle
dma_cscd	0xxx xxxx xxxx xxxxB	DMA cascade
ext_io	1xx0 xxxx x110 00xxB	external I/O access
ext_iord	1xx0 xxxx x110 001xB	external I/O read
ext_iowr	1xx0 xxxx x110 000xB	external I/O write
exec	1xx0 xxxx x0xx xxxxB	executed code
extaddr	1xxx 1xxx xxxx xxxxB	extended address mode
extmemrd	1xx0 1xxx 1110 011xB	memory read in extended address mode
extmemwr	1xx0 1xxx 1110 010xB	memory write in extended address mode
fetch	1xx0 xxxx 1100 011xB	program fetch
grdacc	1xxx xxxx 0xxx xxxxB	guarded access
haltack	1xx0 xxxx x111 000xB	halt acknowledge
holdack	1xx1 xxxx xxxx xxxxB	hold acknowledge
intacki	1xx0 xxxx x100 101xB	interrupt acknowledge (ICU)
intacks	1xx0 xxxx x100 001xB	interrupt acknowledge (SLAVE)
int_io	1xx0 xxxx x110 10xxB	internal I/O access
int_iord	1xx0 xxxx x110 101xB	internal I/O read
int_iowr	1xx0 xxxx x110 100xB	internal I/O write
memory	1xx0 xxxx 1110 01xxB	memory access
memforcp	1xx0 xxxx 1101 01xxB	memory access for cp
memread	1xx0 xxxx 1110 011xB	memory read
memrdcp	1xx0 xxxx 1101 011xB	memory read for cp
memwrite	1xx0 xxxx 1110 010xB	memory write
memwrpc	1xx0 xxxx 1101 010xB	memory write for cp
mon	1xxx x0xx xxxx xxxxB	monitor cycle
nmladdr	1xxx 0xxx xxxx xxxxB	normal address mode
nmlmemrd	1xx0 0xxx 1110 011xB	memory read in normal address mode
nmlmemwr	1xx0 0xxx 1110 010xB	memory write in normal address mode
read	1xx0 xxxx x1xx xx1xB	read cycle
refresh	1xx0 xxxx x100 111xB	refresh cycle
write	1xx0 xxxx x1xx xx0xB	write cycle
wrrom	1xxx xxx0 x1xx xx0xB	write to ROM

70236 Analysis Status Qualifiers

This trace command example uses the status qualifier "read". The following analysis status qualifiers also can be used with the 70236 emulator.

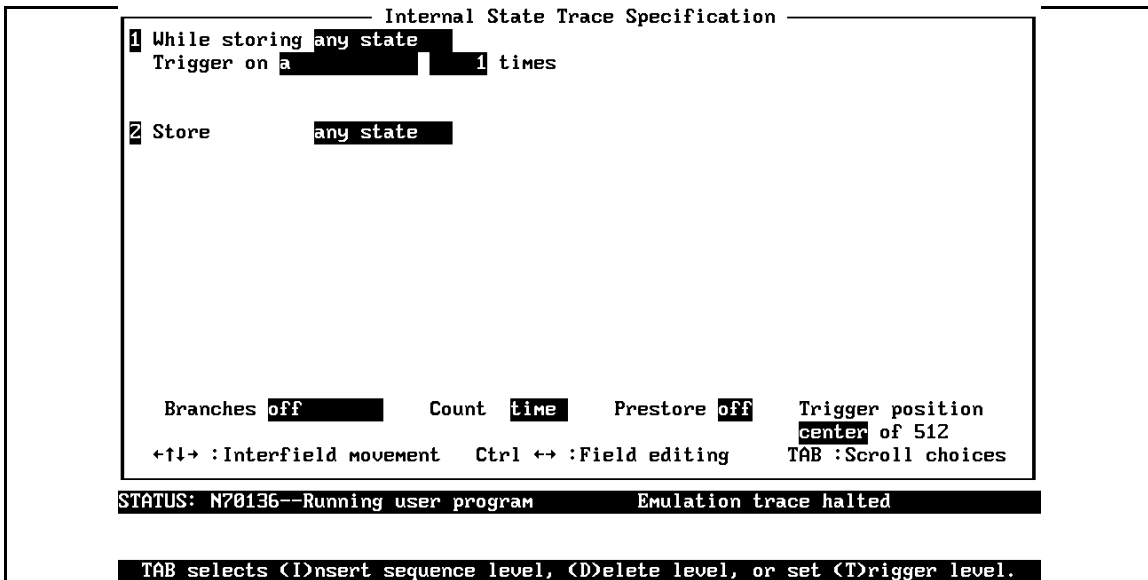


Figure 2-5. Modifying the Trace Specification

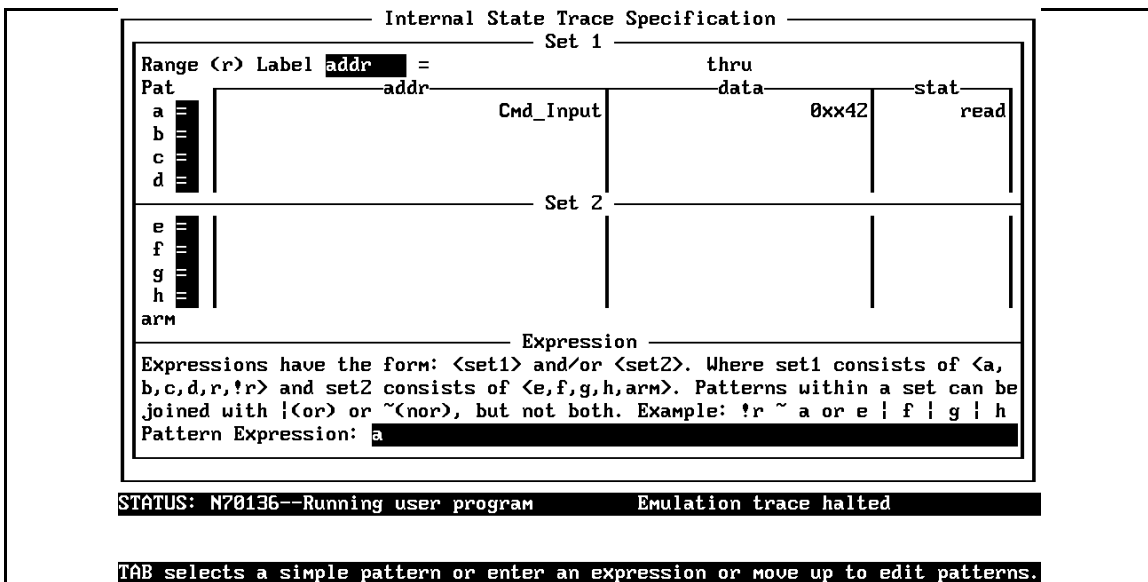


Figure 2-6. Modifying the Pattern Specification

Note



You can combine qualifiers to form more specific qualifiers. For example, the expression **memory&&read** matches only memory reads. See the *Emulator PC Interface Reference* for more information.

Select the **read** status and press **Enter**.

The resulting analysis specification is shown in figure 2-5. To save the new specification, use **End Enter** to exit the field in the lower right corner. You'll return to the trace specification. Press **End** to move the "trigger position" field. Use the TAB key until it says **center**, then press **Enter** to exit the trace specification.

Starting the Trace

To start the trace, select:

Analysis, Begin

A message on the status line will show you that the trace is running. You do not expect the trigger to be found, because no commands have been entered. Modify the command input byte to "B" by selecting:

Memory, Modify, Byte

Enter **Cmd_Input="B"**. The status line now shows that the trace is complete. (If you have problems, you may be running in monitor. Select **Processor Go Pc** to return to the user program.)

Change the Analyzer Display Format

To change the analyzer display format, enter the command:

Analysis Format

Use the down arrow key to move to the field labeled **addr**. And, use the right arrow key to move the field labeled **Width** above. The default width of the address column is six characters. A width of 17 characters is often wide enough to accommodate most symbol names. Type **17** to change the width of the address column, and press **End**, then **Enter**.

Displaying the Trace

To display the trace, select:

Analysis, Display

You are now given two fields in which to specify the states to display. Use the **End** key to move the cursor to the "Ending state to display" field. Type 60 into the press **Enter**. The resulting trace is similar to trace shown in the following display (use <CTRL>z to zoom the trace window). You may need to press the **Home** key to get to the top of the trace.

```

Analysis
-----
Line  addr,H  70236 mnemonic,H          count,R  seq
-----
-7  00041c   f874  prefetch          N  0.440 uS  .
-6  00041c  BE/Z   000416          N  0.160 uS  .
-5  00041e   413c  prefetch          N  0.400 uS  .
-4  000416   a026  prefetch          N  0.560 uS  .
-3  000418   0800  prefetch          N  0.560 uS  .
-2  000416  MOV   AL,DS1:0800    N  0.240 uS  .
-1  00041a   003c  prefetch          N  0.320 uS  .
 0  000800   xx42  memory read       N  0.560 uS  +
 1  00041a  CMP   AL,#00         N  0.120 uS  .
 2  00041c   f874  prefetch          N  0.440 uS  .
 3  00041c  BE/Z   000416          N  0.200 uS  .
 4  00041e   413c  prefetch          N  0.360 uS  .
 5  00041e  CMP   AL,#41        N  0.200 uS  .
 6  000420   0774  prefetch          N  0.360 uS  .
 7  000420  BE/Z   000429          N  0.200 uS  .
 8  000422   423c  prefetch          N  0.360 uS  .


STATUS: N70236--Running user program      Emulation trace complete
Window System Register Processor Breakpoints Memory Config Analysis
Begin Halt CMB Format Trace Display
    
```


Note




The character displayed in the right side of disassemble list specifies the following information.

Character	Information
N	Normal address mode
E	Extended address mode
M	Monitor cycle (background)

Note  Running in the user program, symbols can not be displayed in the trace list.

Note  When you use the following setting in your program, the branch address forming in PC relative addressing may change to a wrong value in disassemble trace list.

- The program is running in the extended address mode.
- The effective address for the PC relative addressing is in the other page.
- The order of the pages is not in sequence in extended address.

Note  If you choose to dump a complete trace into the trace buffer, it will take a few minutes to display the trace.

Line 0 in the above trace list shows the analyzer trigger state. The trigger state is always on line 0. The other states show the exit from the **Scan** loop and the **Exe_Cmd** instructions.

Processor, Break
Analysis, Display

Line	addr,H	70236 mnemonic,H	count,R	seq
9	000422	CMP AL, #42	0.200 uS	.
10	000424	0c74 prefetch	N 0.360 uS	.
11	000424	BE/Z CMD_RDS.S:Cmd_B	0.200 uS	.
12	000426	12e9 prefetch	N 0.360 uS	.
13	:Cmd_B	12b9 prefetch	N 0.560 uS	.
14	000434	be00 prefetch	N 0.600 uS	.
15	:Cmd_B	MOV CW, #0012	0.160 uS	.
16	000436	0612 prefetch	N 0.400 uS	.
17	000435	MOV IX, #0612	0.240 uS	.
18	000438	06e9 prefetch	N 0.320 uS	.
19	00043a	b900 prefetch	N 0.560 uS	.
20	000438	BR NEAR PTR CMD_RDS.S:Write_Msg	0.160 uS	.
21	00043c	0012 prefetch	N 0.400 uS	.
22	te_Msg	bfxx prefetch	N 0.560 uS	.
23	000442	0801 prefetch	N 0.560 uS	.
24	te_Msg	MOV IX, #0801	0.240 uS	.

STATUS: N70236--Running in monitor Emulation trace complete

Window System Register Processor Breakpoints Memory Config Analysis

Begin Halt CMB Format Trace Display

The resulting display shows the **Cmd_B** instructions, the branch to **Write_Msg**, and the beginning of the instructions that move the Entered B command message to the destination locations.

For a Complete Description

For a complete description of using the HP 64700 Series analyzer with the PC Interface, refer to the *Analyzer PC Interface User's Guide*.

Copying Memory

You can copy the contents of one range of memory to another. This is a useful feature to test things like the relocatability of programs. To test if the sample program is relocatable within the same segment, copy the program to an unused, but mapped, area of emulation memory. For example, select:

`Memory, Copy`

Enter 400H through 452H as the source memory range to be copied, and enter 500H as the destination address.

To verify that the program is relocatable, run it from its new address by selecting:

`Processor, Go, Address`

Enter 500H. The status line shows that the emulator is "Running user program". You may wish to trace program execution or enter valid and invalid commands and search the message destination area (shown earlier in this chapter) to verify that the program works correctly at its new address.

Resetting the Emulator

To reset the emulator, select:

`Processor, Reset, Hold`

The emulator is reset (suspended) until you enter a "**P**rocessor **B**reak", "**P**rocessor **G**o", or "**P**rocessor **S**tep" command. A CMB execute signal also will run the emulator if reset.

You also can specify that the emulator begin executing in the monitor after reset instead of remaining in the suspended state.

To do this, select:

`Processor, Reset, Monitor`

Exiting the PC Interface

There are three different ways to exit the PC Interface. You can exit the PC Interface using the "locked" option which restores the current configuration next time you start the PC Interface. You can select this option as follows.

```
System, Exit, Locked
```

Another way to execute the PC Interface is with the "unlocked" option, which presents the default configuration the next time you start the PC Interface. You can select this option with the following command.

```
System, Exit, Unlocked
```

Or, you can exit the PC Interface without saving the current configuration using the command:

```
System Exit No_Save
```

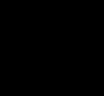
See the *Emulator PC Interface Reference* for a complete description of the system exit options and their effect on the emulator configuration.

In-Circuit Emulation

Introduction

The emulator is *in-circuit* when it is plugged into the target system. This chapter covers topics which relate to in-circuit emulation.

This chapter will:

- Describe the issues concerning the installation of the emulator probe into target systems.
 - Show you how to install the emulator probe.
 - Show you how to use features related to in-circuit emulation.
- 

Prerequisites

Before performing the tasks described in this chapter, you should be familiar with how the emulator operates in general. Refer to the *Concepts of Emulation and Analysis* manual and the "Getting Started" chapter of this manual.

Installing the Target System Probe

The 70136 emulator probe has a 68-pin PLCC connector; the 70236 and 70236A emulator probe has a 132-pin PGA connector.

Caution



OBSERVE THESE PRECAUTIONS TO AVOID EMULATOR CIRCUIT DAMAGE. Take the following precautions while using the 70136 emulator.

Power Down Target System. Turn off power to the user target system and to the 70136 emulator 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 70136 emulator contains devices which are susceptible to damage by static discharge. Therefore, take precautions before handling the user plug to avoid emulator damage.

Protect Target System CMOS Components. If your target system includes any CMOS components, turn on the target system first, then turn on the 70136 emulator. When powering down, turn off the emulator first, then turn off power to the target system.

Pin Protector (70236/70236A Emulator Only)

The target system probe has a pin protector that prevents damage to the probe when inserting and removing the probe from the target system microprocessor socket. **Do not** use the probe without a pin protector installed. If the target system probe is installed on a densely populated circuit board, there may not be enough room for the plastic shoulders of the probe socket. If this occurs, another pin protector may be stacked onto the existing pin protector.

Auxiliary Output Lines

Two auxiliary output lines, "TARGET BUFFER DISABLE" and "SYSTEM RESET", are provided with the 70136 emulator. The "TARGET BUFFER DISABLE" output line is also provided with the 70236 and 70236A emulator.

Caution



DAMAGE TO THE EMULATOR PROBE WILL RESULT IF THE AUXILIARY OUTPUT LINES ARE INCORRECTLY INSTALLED.

When installing the auxiliary output lines into the end of the emulator probe cable, make sure that the ground pins on the auxiliary output lines (labeled with white dots) are matched with the ground receptacles in the end of the emulator probe cable.

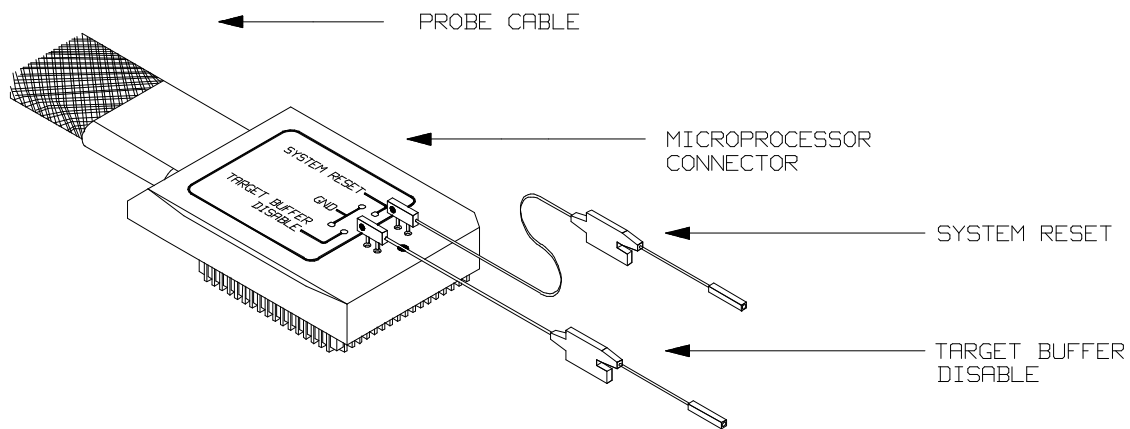
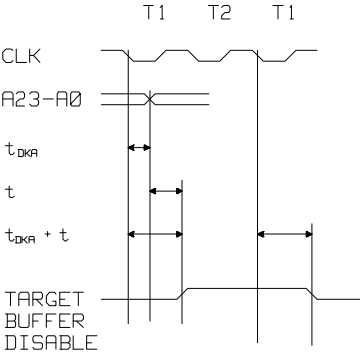


Figure 3-1. Auxiliary Output Lines (70136 Emulator)

TARGET BUFFER DISABLE ---This active-high output is used when the co-processor memory accesses to emulation memory will be operated. This output is used to tristate (in other words, select the high Z output) any target system devices on the 70136 data bus. Target system devices should be tristated because co-processor memory reads from emulation memory will cause data to be output on the user probe.

This "TARGET BUFFER DISABLE" output will be driven with the following timing in the co-processor memory access cycle.



The time 't' is
 26 nsec MIN.
 56.8 nsec MAX.
 (70136 Emulator)
 26 nsec MIN.
 61.8 nsec MAX.
 (70236 Emulator)

SYSTEM RESET (70136 only) ---This active-high, CMOS output should be used to synchronously reset the emulator and the target system.

Installing into a 70136 PLCC Type Socket

To connect the microprocessor connector to the target system, proceed with the following instructions.

- Remove the 70136 microprocessor (PLCC type) from the target system socket. Note the location of pin 1 on the microprocessor and on the target system socket.
- Store the microprocessor in a protected environment (such as antistatic form).
- Install the microprocessor connector into the target system microprocessor socket.

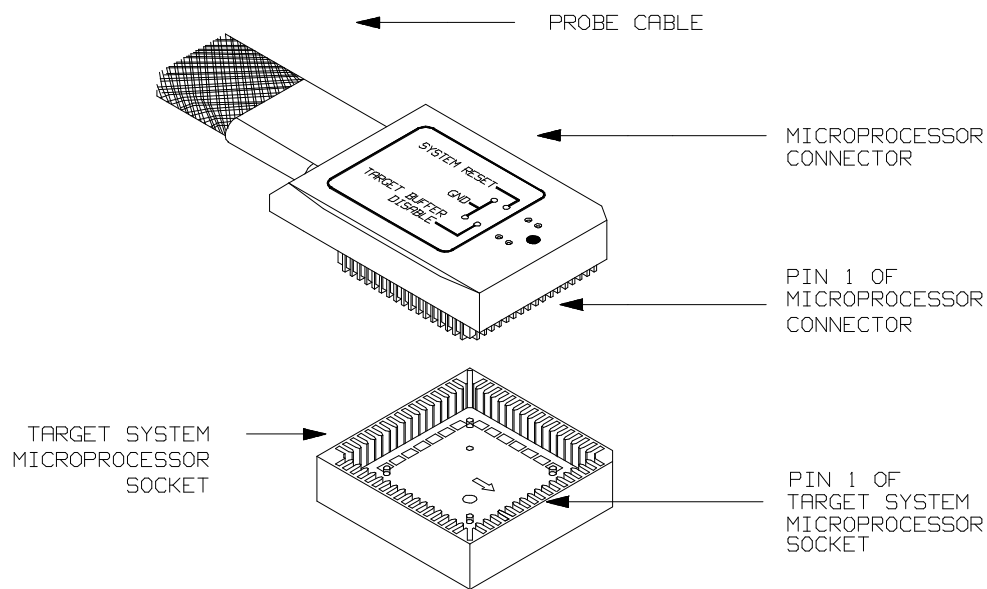


Figure 3-2. Installing into a 70136 PLCC type socket

Installing into a 70136 PGA Type Socket

The 70136 emulator is provided with an AMP 821574-1 socket and a pin protector in order to plug into the target system socket of an PGA type. You may use this AMP socket with the pin protector to connect the microprocessor connector to the target system.

To connect the microprocessor connector to the target system, proceed with the following instructions.

- Remove the 70136 microprocessor (PGA type) from the target system socket. Note the location of pin A1 on the microprocessor and on the target system socket.
- Store the microprocessor in a protected environment (such as antistatic form).
- Place the microprocessor connector with an AMP socket and a pin protector (see figure 3-3), attached to the end of the probe cable, into the target system microprocessor socket.

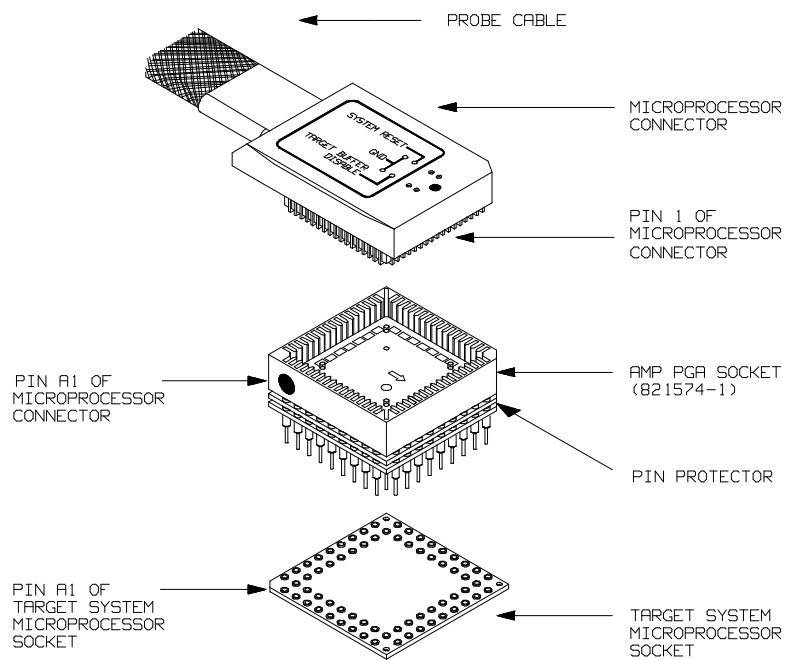


Figure 3-3. Installing into a 70136 PGA type socket

Installing into a 70136 QFP Type Socket

To connect the 70136 emulator microprocessor connector to the NEC EV-9200G-74 socket on the target system, you should use the adapter, HP PART NO. 64756-61612, that will allow the PLCC microprocessor connector to connect to the QFP socket.

To connect the microprocessor connector to the target system, proceed with the following instructions.

- Note the location of pin 1 on the NEC EV-9200G-74 socket on the target system.
- Place the microprocessor connector with the adapter (see figure 3-4), attached to the end of the probe cable, into the target system microprocessor socket.

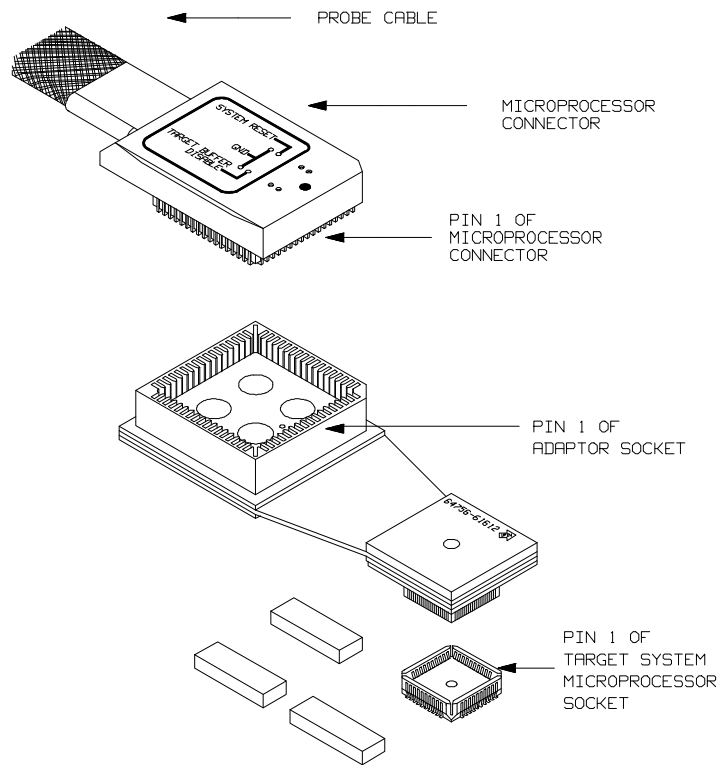


Figure 3-4. Installing into a 70136 QFP type socket

Installing into a 70236/236A PGA Type Socket

To connect the microprocessor connector to the target system, proceed with the following instructions.

- Remove the 70236 or 70236A microprocessor (PGA type) from the target system socket. Note the location of pin A1 on the microprocessor and on the target system socket.
- Store the microprocessor in a protected environment (such as antistatic form).
- Install the microprocessor connector into the target system microprocessor socket with a pin protector (see figure 3-5).

Caution



DO NOT use the microprocessor connector without using a pin protector. The pin protector is provided to prevent damage to the microprocessor connector when connecting and removing the microprocessor connector from the target system PGA socket.

Installing into a 70236/70236A QFP Type Socket

To connect the 70236 or 70236A emulator microprocessor connector to the NEC EV-9200GD-120 socket on the target system, you should use the NEC EV-9500GD-120 adapter that will allow the PGA microprocessor connector to connect to the QFP socket.

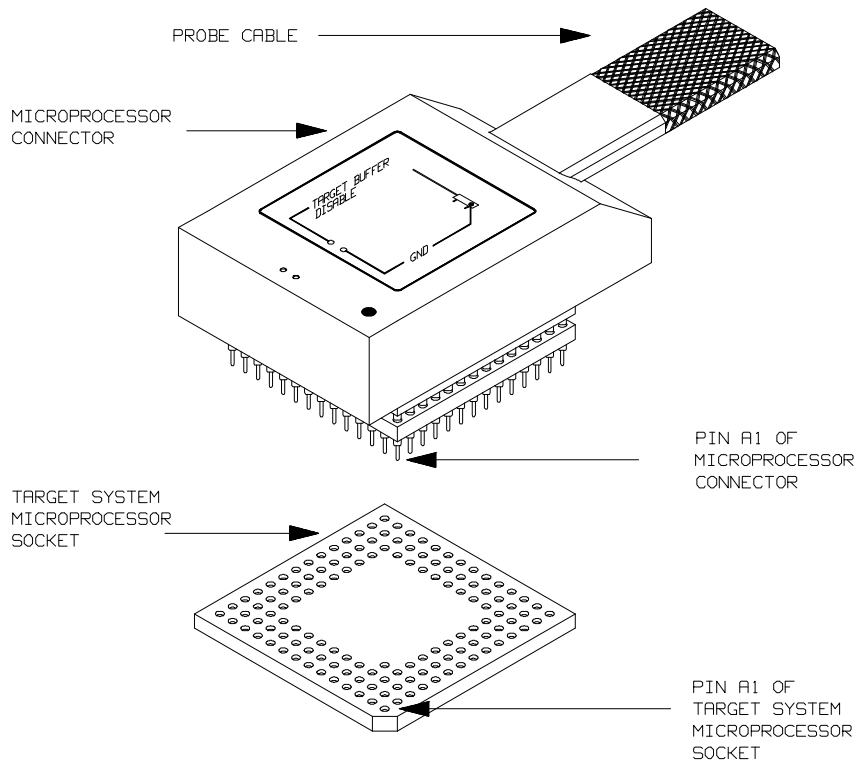


Figure 3-5. Installing into a 70236 PGA type socket

In-Circuit Configuration Options

The 70136 emulator provide configuration options for the following in-circuit emulation issues. Refer to the chapter on "Configuring the Emulator" for more information on these configuration options.

Using the Target System Clock Source

The default 70136, 70236 and 70236A emulator configuration selects the internal 16 MHz (system clock speed) clock as the emulator clock source.

You should configure the emulator to select an external target system clock source for the "in-circuit" emulation.

Allowing the Target System to Insert Wait States

High-speed emulation memory provides no-wait-state operation. However, the emulator may optionally respond to the target system ready line while emulation memory is being accessed.

Note



When you use the NEC uPD72291 coprocessor on your target system connected to 70136 microprocessor, the uPD72291 can access 70136 emulation memory on coprocessor memory read/write cycles.

In this case, you should reset the target system to connect the 70136 emulator to the uPD72291 coprocessor before starting emulation session.

Enabling NMI and RESET Input from the Target System

You can configure whether the emulator should accept or ignore the NMI and RESET signals from the target system.

Running the Emulator from Target Reset

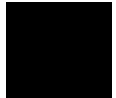
You can specify that the emulator begins executing from target system reset. When the target system RESET line becomes active and then inactive, the emulator will start reset sequence (operation) as actual microprocessor.

At First, you must specify the emulator responds to RESET signal by the target system (see the "Enable RESET Input From Target?" configuration in Chapter 4 of this manual).

To specify a run from target system reset, select:

```
Processor Go Reset <RESET>
```

The status now shows that the emulator is "Awaiting target reset". After the target system is reset, the status line message will change to show the appropriate emulator status.



Pin State in Background (70136)

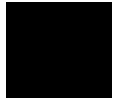
While the emulator is running in the background monitor, probe pins are in the following state.

Address Bus	Same as foreground
Data Bus	Always high impedance except accessing target. When accessing target by background monitor, same as foreground.
$\overline{R/W}, \overline{M/IO}$ BUSST0	Always high level except accessing target. When accessing target by background monitor, same as foreground.
BUSST1	Always low level except accessing target. When accessing target by background monitor, same as foreground.
Other	Same as foreground

Pin State in Background (70236/70236A)

While the emulator is running in the background monitor, probe pins are in the following state.

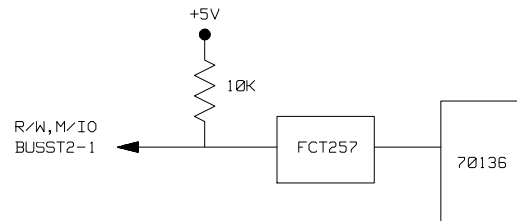
Address Bus	Same as foreground
Data Bus	Always high impedance except accessing target. When accessing target by background monitor, same as foreground.
$\overline{R/W}, \overline{M/IO}, \overline{IORD}, \overline{IOWR}, \overline{MWR}$	Always high level except accessing target. When accessing target by background monitor, same as foreground.
\overline{MRD}	Same as foreground except for emulation memory write. When accessing emulation memory write, low.
Other	Same as foreground



Target System Interface (70136)

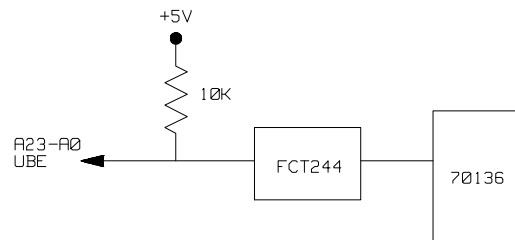
$\overline{R/W}$ $\overline{M/IO}$
BUSST2-1

These signals are connected to 70136 through FCT257 and 10K ohm pull-up register.



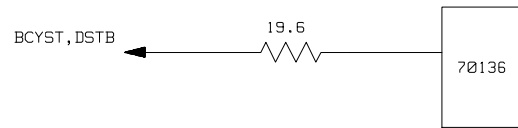
$\overline{A23-A0}$
 \overline{UBE}

These signals are connected to 70136 through FCT244 and 10K ohm pull-up register.



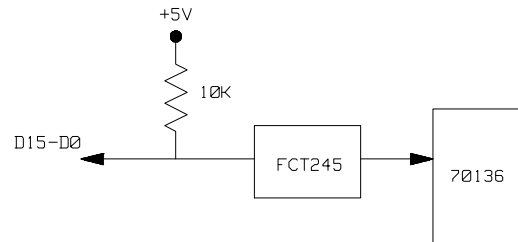
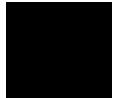
BCYST DSTB

These signals are connected to 70136 through 19.6 ohm.



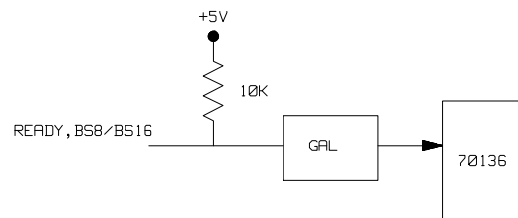
D15-D0

These signals are connected to 70136 through FCT245 and 10K ohm pull-up register.



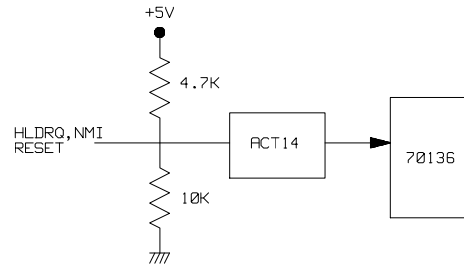
READY
BS8/BS16

These signals are connected to 70136 through GAL and 10K ohm pull-up register.



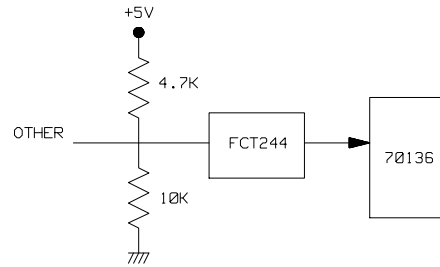
HLDRO
NMI
RESET

These signals are connected to 70136 through ACT14 and 4.7K ohm pull-up and 10K ohm pull-down registers.



OTHER

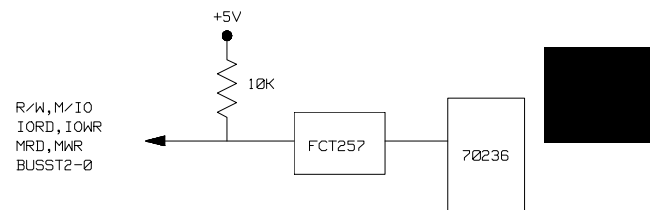
These signals are connected to 70136 through FCT244 and 4.7K ohm pull-up and 10K ohm pull-down registers.



Target System Interface (70236/70236A)

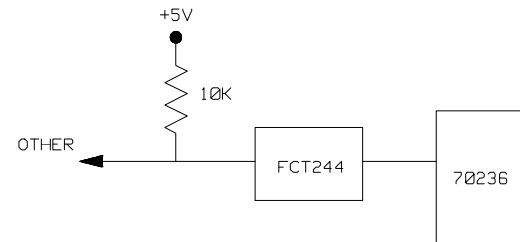
$\overline{R/W}$ $\overline{M/IO}$
 \overline{IORD} \overline{IOWR}
 \overline{MRD} \overline{MWR}
BUSST2-0

These signals are connected to 70236/70236A through FCT257 and 10K ohm pull-up register.



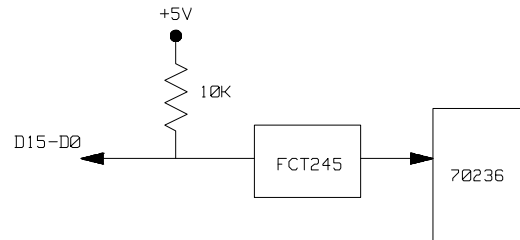
OTHER(INPUT)

These signals are connected to 70236/70236A through FCT244 and 10K ohm pull-up register.



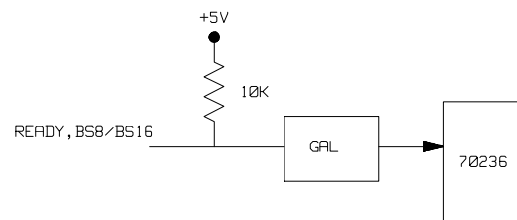
D15-D0

These signals are connected to 70236/70236A through FCT245 and 10K ohm pull-up register.



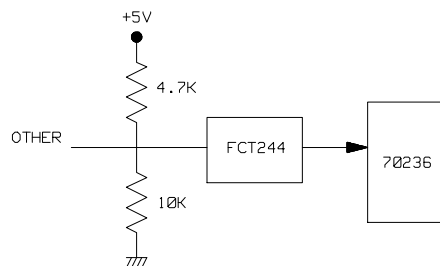
READY
BS8/BS16

These signals are connected to 70236/70236A through GAL and 10K ohm pull-up register.



OTHER(OUTPUT)

These signals are connected to 70236/70236A through FCT244 and 4.7K ohm pull-up and 10K ohm pull-down registers.



Configuring the 70136 Emulator

Introduction

Your 70136 emulator can help you in all stages of target system development. For instance, you can run the emulator out-of-circuit when developing target system software and in-circuit when integrating software with hardware. You can use the emulator's internal clock or your target system clock. Emulation memory can be used with your target system memory, and it can be mapped as RAM or ROM. You can execute your target programs in real-time or allow emulator execution to be diverted into the monitor when commands request access of target system resources (target system memory, register contents, etc.)

The emulator is a versatile instrument and may be configured to suit your needs at any stage of the development process. This chapter describes the emulator configuration options.

This chapter will:

- Show you how to access the emulator configuration options.
- Describe the emulator configuration options.
- Show you how to save a particular emulator configuration, and load it again at a later time.

Prerequisites

Before performing the tasks described in this chapter, you should be familiar with how the emulator operates in general. Refer to the *HP 64700 Emulators: Concepts of Emulation and Analysis* manual and the "Getting Started" chapter of this manual.

Accessing the Emulator Configuration Options

Select:
Config, General

When you position the cursor to a configuration item, a brief description of the item appears at the bottom of the display.

```
General Emulation Configuration
Internal clock      [y] Real-time mode      [n] Break on ROM writes [y]
Software brkpoints [n] CMB interaction    [n] Target interrupts  [y]
Target RESET      [y] Lock RDY signal    [n] Read PGR registers [y]
AEX through to target [y] uPD72291 FPU      [y] 20 bit address mode [n]
Bus sizing signal for emul_mem ->emul Bus sizing signal for target_mem->tgt
Seg:off translation method minseg Background monitor location 00ff000
Monitor type      background

<↑↓> : Interfield movement  Ctrl ↔ : Field editing  TAB : Scroll choices

STATUS: N70136--Emulation reset           Emulation trace halted
When 'yes' is selected, the address in the absolute file is assumed as 20 bit
physical. Otherwise, it is considered as 24 bit extended.
```

Figure 4-1. General Emulator Configuration (70136)

4-2 Configuring the 70136 Emulator


```

General Emulation Configuration
Internal clock      [y] Real-time mode      [n] Break on ROM writes [y]
Software brkpoints [n] CMB interaction    [n] Target interrupts  [y]
Target RESET      [y] Lock RDY signal    [n] Read PGR registers [y]
AEX through to target [y] uPD72291 FPU      [y] Z0 bit address mode [n]
Release bus by HOLD [n] Trace DMA cycles    [y] Trace refresh cycles [y]
Wait count of DMA  0 DMA cycle in backgrnd [y] Trace dummy HALTACK [y]
Bus sizing signal for emul_mem ->emul Bus sizing signal for target_mem->tgt
Seg:off translation method minseg Background monitor location 00ff000
Monitor type      background
+↑↓: Interfield movement Ctrl ↔ : Field editing TAB : Scroll choices

STATUS: N70236--Emulation reset Emulation trace halted
When 'yes' is selected, the emulator uses the internal clock in emulation pod.
Otherwise, the clock input from the target system clocks the emulator.

```

Figure 4-2. General Emulator Configuration (70236)

Note



You can use the System Terminal window to modify the emulator configuration. If you do this, some PC Interface features may no longer work properly. We recommend that you modify the emulator configuration using only the PC Interface.

Internal Clock

This configuration item allows you to select whether the emulator will be clocked by the internal clock source or by a target system clock source.

Yes Selects the internal clock oscillator as the emulator clock source.

The internal clock speed of the 70136, 70236 and 70236A are 16 MHz (system clock). This is the default.

No An external target system clock is the emulator clock source.

In the 70136 emulator, external clock sources must be within the range of 2-16 MHz.

In the 70236 emulator, external clock sources must be within the range of 4-32 MHz.

In the 70236A emulator, external clock sources must be within the range of 4-40 MHz.



Note



When the 70136 emulator is plugged into the target system, you should use the external target system clock source to synchronize the emulator with the target system.

Note



Changing the clock source drives the emulator into the reset state.

Real-Time Mode

The "Real-Time mode" question lets you configure the emulator to refuse commands that cause an emulator break to monitor during user program runs.

- No** All commands, whether or not they require a break to the emulation monitor, are accepted by the emulator.
- Yes** When runs are restricted to real-time and the emulator is running the user program, all commands that cause a break (except "**P**rocessor **R**eset", "**P**rocessor **B**reak", "**P**rocessor **G**o", and "**P**rocessor **S**tep") are refused. For example, the following commands are not allowed when runs are restricted to real-time:
- Display/modify registers.
 - Display/modify target system memory.
 - Display/modify I/O.

Caution



Restrict emulator to real-time runs with certain target systems. If your target system circuitry depends on constant program execution, you should restrict the emulator to real-time runs. This helps avoid target system damage. Remember that you still can execute the "**P**rocessor **R**eset", "**P**rocessor **B**reak", and "**P**rocessor **S**tep" commands. You should use caution when executing these commands.

Note



When program execution should take place in real-time and the emulator should break to the monitor to read page registers (refer to "Read PGR registers" section in this chapter), the following commands are not allowed with using physical or <segment>:<offset> address expression.

- Display/modify emulation memory.
-

Break on ROM Writes

This question allows you to specify that the emulator break to the monitor upon attempts to write to memory space mapped as ROM. The emulator will prevent the processor from writing to memory mapped as emulation ROM. It cannot prevent writes to target system RAM locations mapped as ROM, though the write to ROM break is enabled.

- Yes** Causes the emulator to break into the emulation monitor whenever the user program attempts to write to a memory region mapped as ROM.
- No** The emulator will not break to the monitor upon a write to ROM.

Note



The **wrom** analysis specification status option allows you to use write to ROM cycles as trigger and storage qualifiers.

Software Breakpoints

This question allows you to enable or disable the software breakpoints feature.

When you define (add) a breakpoint, software breakpoints are automatically enabled.

- No** The software breakpoints feature is disabled. This is the default emulator configuration, so you must change this item before you can use software breakpoints.
- Yes** Allows you to use the software breakpoints feature. The emulator detects the breakpoint interrupt instruction (F1 hex), it generates a break to background request which as with the "processor break" command.

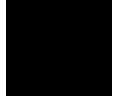
When you define or enable a software breakpoint to a specified address, the emulator will replace the opcode with one of 70136 undefined opcode (F1 hex) as breakpoint interrupt instruction. When the emulator detects the breakpoint interrupt instruction (F1 hex), user program breaks to the monitor, and the original opcode will be replaced at the software breakpoint address. A subsequent run or step command will execute from this address.

Since the system controller knows the locations of defined software breakpoints, it can determine whether the breakpoint interrupt instruction (F1 hex) is a software breakpoint or opcode in your target program.

If it is a software breakpoint, execution breaks to the monitor, and the breakpoint interrupt instruction is replaced by the original opcode. A subsequent run or step command will execute from this address.

If it is an opcode of your target program, execution still breaks to the monitor, and an "Undefined software breakpoint" status message is displayed.

Refer to the "Getting Started" for information on using software breakpoints.



CMB Interaction

Coordinated measurements are measurements made synchronously in multiple emulators or analyzers. Coordinated measurements can be made between HP 64700 Series emulators that communicate over the Coordinated Measurement Bus (CMB).

Multiple emulator start/stop is one type of coordinated measurement. The CMB signals READY and /EXECUTE are used to perform multiple emulator start/stop.

This configuration item allows you to enable/disable interaction over the READY and /EXECUTE signals. (The third CMB signal, TRIGGER, is unaffected by this configuration item.)

No The emulator ignores the /EXECUTE and READY lines, and the READY line is not driven.

Yes Multiple emulator start/stop is enabled. If you enter the

`Processor, CMB, Go, ...`

command, the emulator will start executing code when a pulse on the /EXECUTE line is received. The READY line is driven false while the emulator is running in the monitor. It goes true whenever execution switches to the user program.

Note



CMB interaction also will be enabled when you enter the

`Processor, CMB, Execute`

command.

Target Interrupts

This configuration option specifies whether or not the emulation processor accepts to NMI signal generated by the target system.

Yes The emulator accepts NMI signal generated by the target system. When the NMI signal is accepted, the emulator calls the NMI procedure as actual microprocessor.

No The emulator ignores NMI signal from target system completely.

Note



You should not use "**P**rocessor **S**tep" command if target system can generates NMI.

When the emulator accepts NMI input in stepping, the following error message will be shown.

```
ERROR : Stepping failed
```

In this case, you should configure that the emulator ignores NMI input from the target system with this configuration.

Target RESET

The 70136 emulator can respond or ignore target system reset while running in user program or waiting for target system reset (refer to "Processor Go Reset" command in "In-circuit Emulation" chapter). While running in background monitor, the 70136 emulator ignores target system reset completely independent on this setting.

- | | |
|------------|---|
| Yes | Specify that, this is a default configuration, make the emulator to respond to reset from target system. In this configuration, emulator will accept reset and execute from reset vector (0FFFF0 hex) as same manner as actual microprocessor after reset is inactivated. |
| No | If disabled, the emulator completely ignores the reset signal from target system. This is true if the emulator is in foreground (executing user program). |



Lock RDY Signal

High-speed emulation memory provides no-wait-state operation. However, the emulator may optionally respond to the target system ready lines while emulation memory is being accessed.

- | | |
|------------|---|
| No | When the ready relationship is not locked to the target system, emulation memory accesses ignore ready signals from the target system (no wait states are inserted). |
| Yes | When the ready relationship is locked to the target system, emulation memory accesses honor ready signals from the target system (wait states are inserted if requested). |

Read PGR Registers

This configuration item allows you to specify whether the emulator should break to the monitor to read page registers or whether the emulator should use the copy of page registers when the emulation system will convert logical address to extended address in the following commands.

- Display/modify memory with entering physical, <SEGMENT>:<OFFSET>, or no function code address expression.
- Modify software breakpoints.

Yes Specifies that the emulator should break to the monitor to get the current value of page registers on accesses to emulation/target memory.

No Specifies that the emulator should use the copy of page registers which is renewed at breaking to the monitor or changing the value of page registers with using the following PC Interface command.

Register Modify <pgr 1 .. pgr 64>

You should select this configuration when you only use the normal address mode in your program or the value of page registers is not changed after initializing while executing your program.

AEX Through to Target

This configuration option allows you to select the **AEX** (Address Extension) signal level which is driven to the target system while in the background monitor cycles.

- Yes** Specifies that the emulator will drive the **AEX** signal with the level dependent on the address mode in background monitor cycles. When you use the extended address in an emulation command, the **AEX** signal will be driven to high level in background monitor.
- No** Specifies that the emulator will hold the **AEX** signal with the level dependent on the last foreground address mode just before entering background monitor. When the program is running on normal address mode, the emulator will hold the **AEX** signal level low in background monitor.

uPD72291 FPU

This configuration option allows you to select the assembler mnemonics for FPU (Floating Point Unit) to display memory.

- Yes** Specifies that mnemonics for NEC uPD72291 floating point processor will be used to display memory.
- No** Specifies that mnemonics for Intel 80287 numeric processor extension will be used to display memory.

20 Bit Address Mode

This configuration option allows you to specify the load address of an absolute file in "Memory Lord" command.

- No** Specify that the emulator will interpret address in absolute file as 24-bit extended address.
- Yes** Specifies that the emulator will interpret address in absolute file as 20-bit physical address.

Release bus by HOLD

(70236/70236A Emulator only) This configuration allows you to specify whether or not the emulator accepts HLDRQ (Hold Request) signal generated by the target system in background.

- No** The emulator ignores HLDRQ signal from target system completely in background.
- Yes** The emulator accepts HLDRQ signal. When the HLDRQ is accepted, the emulator will respond as actual microprocessor.

Trace DMA Cycles

(70236/70236A Emulator only) This question allows you to specify whether or not the analyzer trace the emulation processor's internal DMA cycles.

- Yes** Specifies that the analyzer will trace the internal DMA cycles.
- No** Specifies that the analyzer will not trace the internal DMA cycles.

Trace Refresh Cycles

(70236/70236A Emulator only) This question allows you to specify whether or not the analyzer trace the emulation processor's refresh cycles.

- | | |
|------------|--|
| Yes | Specifies that the analyzer will trace the refresh cycles. |
| No | Specifies that the analyzer will not trace the refresh cycles. |

Wait count of DMA

(70236/70236A Emulator only) When you want to trace internal DMA cycles correctly with using the emulator, you must set the number of wait count for internal DMA cycles.

The number is the same as the value of DMAW (Wait for the DMA cycle) of the WCY4 (programmable wait, cycle 4) register (I/O address FFF6 hex). See the "Trace DMA Cycles" in this chapter.

DMA cycle in Background

(70236/70236A Emulator only) This configuration allows you to specify whether or not the emulation processor's internal DMA is allowed while in background.

- | | |
|------------|--|
| Yes | The internal DMA is allowed while in background. |
| No | The internal DMA is not allowed while in background. |

Trace Dummy HALTACK

(70236 Emulator only) This question allows you to specify whether or not the analyzer trace the emulation processor's dummy HALT acknowledge cycles.

Whenever breaks occur during the emulation processor is HALTed, the HALT acknowledge cycle will be occurred one more time. This configuration specifies that the analyzer trace or not this HALT acknowledge cycles.

- | | |
|------------|---|
| No | Specifies that the analyzer will not trace the dummy HALT acknowledge cycles. |
| Yes | Specifies that the analyzer will trace the dummy HALT acknowledge cycles. |

Bus Sizing Signal for Emul_mem

- | | |
|-------------|---|
| emul | Specifies that the bus size of emulation memory is selected from the setting of the map configuration. Refer to the "Mapping Memory" command description in "Using the Emulator" chapter. |
| tgt | Specifies that the <u>bus</u> size of emulation memory is defined from the BS8/BS16 input of the target system. |

Bus Sizing Signal for Target_mem

tgt	Specifies that the <u>bus</u> size of target memory is defined from the BS8/BS16 input of the target system.
emul	Specifies that the bus size of target memory is selected from the setting of the map configuration. Refer to the "Mapping Memory" command description in "Using the Emulator" chapter.

Note



The data bus size of I/O accesses is only defined from the BS8/BS16 input of the target system.

Seg:off Translation Method

The run and step commands allow you to enter addresses in either logical form (segment:offset, e.g., 0F000H:0000H) or physical form (e.g., 0F000H). When a physical address (non-segmented) is entered with either a run or step command, the emulator must convert it to a logical (segment:offset) address.

minseg Specifies that the physical run address is converted such that the low 16 bits of the address become the offset value. The physical address is right-shifted 4 bits and ANDed with 0F000H to yield the segment value.

```
logical_addr = ((phys_addr >> 4) & 0xf000):(phys_addr & 0xffff)
```

maxseg Specifies that the low 4 bits of the physical address become the offset. The physical address is right-shifted 4 bits to yield the segment value.

```
logical_addr = (phys_addr >> 4):(phys_addr & 0xf)
```

curseg Specifies that the value entered with either a run or step command (0 thru 0ffff hex) becomes the offset. In this selecting, the current segment value is not changed.

```
logical_addr = (current segment):(entered value)
```

If you use logical addresses other than the three methods which follow, you must enter run and step addresses in logical form.

Background Monitor Location

You can relocate the monitor from the default monitor location to any 4K byte boundary. When entering monitor block addresses, you must only specify addresses on 4K byte boundaries; otherwise, an invalid syntax message is displayed. The location of background monitors may be important because background cycles of the 70136 emulator can be visible to the target system.

In default, the monitor is located on 0FF000 hex through 0FFFFFF hex.

Note



If your target system have some circuitry which monitors bus activities to detect illegal access to resources, You may need to relocate monitor address.

Monitor Type

This configuration option allows you to select and use a foreground emulation monitor program. The default monitor is background monitor.

Note



Halt instructions will cause "processor halted" emulation status.

In this status, the emulator cannot break to the monitor.

In this case, you should enter the "reset" command to reset the emulator first.

background

Specify monitor type as background monitor. When you select background monitor, you can specify the background monitor location.

Note



While running in background monitor, the 70136 emulator ignores target system reset.

foreground

Specify monitor type as foreground monitor. When you select foreground monitor, you must specify correct foreground monitor start address with next configuration question (foreground monitor address). After you completed the configuration setting, you need to load foreground monitor program to the emulator with "Memory, Load" feature. The foreground monitor program must already assembled and linked with appropriate location specification. Refer to the *HP 64756 70136 Emulator Terminal Interface User's Guide* for more information.

Note



You must **not** use the foreground monitor if you wish to perform coordinated measurements.



Note



If you select a foreground monitor, a 4 kilobyte block is automatically mapped at the address specified by the next question.

Foreground Monitor Address?

The location of the foreground monitor is important because it will occupy part of the processor address space. Foreground monitor location must not overlap the location of target system programs. The default foreground monitor location is "0F0000H".

When entering monitor block addresses, you must only specify addresses on 4K byte boundaries; otherwise, an invalid syntax message is displayed.

Note



Relocating the monitor causes all memory mapper terms to be removed.

Note



You should not load the foreground monitor provided with the 70136 emulator at the base address 0 or 0ff000 hex; the 70136 microprocessor's vector table is located.

And, You can not load the foreground monitor at the base address over 100000 hex.

Storing an Emulator Configuration

The PC Interface lets you store a particular emulator configuration so that it may be re-loaded later. The following information is saved in the emulator configuration.

- Emulator configuration items.
- Key macro specifications.
- Memory map.
- Break conditions.
- Trigger configuration.
- Window specifications.

To store the current emulator configuration, select:

`Config, Store`

Enter the name of a file in which to save the emulator configuration.

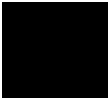
Loading an Emulator Configuration

If you want to reload a previously stored emulator configuration, select:

`Config, Load`

Enter the configuration file name and press **Enter**. The emulator will be reconfigured with the values specified in the configuration file.

Notes



Using the Emulator

Introduction

In the "Getting Started" chapter, you learned how to use the basic features of the 70136 emulator. This chapter describes the more in-depth features of the emulator.

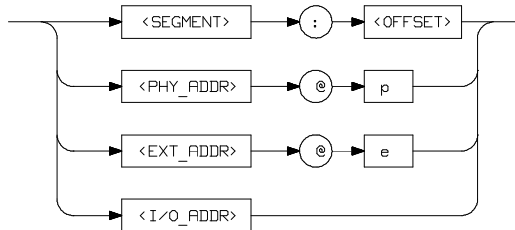
This chapter shows you how to:

- Address syntax in emulation commands.
- Address expression in emulation commands.
- Register names and classes.
- Make coordinated measurements.
- Store the contents of memory into absolute files.



Address Syntax

Syntax



The address used in emulation commands may be specified as a segment:offset address, physical address, or as an extended address (though a physical address in run commands (see table 5-1) is converted to a <segment>:<offset> address and an extended address in memory commands (see table 5-1) is converted to a value of the page register and a <segment>:<offset> address by the emulation system).

The physical and extended address specifications are of the following form. "@e" and "@p" are the function codes to define as an extended or a physical address.

Extended address EXT_ADDR@e

Physical address PHY_ADDR@p

Expressions are defined in the *HP 64700 Emulators Terminal Interface: User's Reference* manual.

Parameters

- <SEGMENT>** This expression (0-0FFFF hex) is the segment portion of the logical address. The value specified is placed in the 70136 PS register.
- <OFFSET>** This expression (0-0FFFF hex) is the offset portion of the logical address. The value specified is placed in the 70136 PC register.
- <PHY_ADDR>** This expression (0-0FFFFFF hex) with "@p" function code is a physical address in the 70136 address range. In run commands (see table 5-1), the emulation system converts this physical address to a <segment>:<offset> address as specified by the "Seg:off translation method" configuration option in "Configuring the 70136 Emulator" chapter.
- <EXT_ADDR>** This expression (0-0FFFFFFF hex) with "@e" function code is a extended address in the 70136 address range. In memory commands (see table 5-1), the emulation system converts this extended address to a value of the page register and a <segment>:<offset> address to access the memory.
- <I/O_ADDR>** This expression (0-0FFFF hex) with no function code is a 70136 I/O address. This expression should be used in I/O command (see table 5-1).

Address Expression

Table 5-1 is the address expression matrix used in emulation commands.

Table 5-1. Address expression syntax

Command group		<EXT_ADDR> (@e)	<PHY_ADDR> (@p)	<SEGMENT>:<OFFSET>	No function code
Memory commands		OK	OK(*1)	OK(*1)	same as <PHY_ADDR>
Run commands (*2)	maxseg	ERROR	OK	OK	same as <PHY_ADDR>
	minseg	ERROR	OK	OK	same as <PHY_ADDR>
	curseg	ERROR	ERROR	OK	<OFFSET> (0-0FFFFH)
I/O command		ERROR	ERROR	ERROR	OK (0-0FFFFH)
Map command		OK	ERROR	ERROR	ERROR
Breakpoints command		OK	OK(*1)	OK(*1)	same as <PHY_ADDR>

*1 : Emulator breaks to the monitor on accesses to emulation memory (refer to the "Read PGR registers" in "Configuring the 70136 Emulator" chapter.)

*2 : Refer to "Seg:off Translation Method" in "Configuring the 70136 Emulator" chapter.

Memory Commands

The following commands are included in memory commands.

Memory, Display
Memory, Modify
Memory, Store
Memory, Copy
Memory, Find

You can use the following address expression in memory commands (refer to "Read PGR registers" configuration item in "Configuring the 70136 Emulator" chapter).

	<EXT_ADDR>	<PHY_ADDR>	<SEGMENT>: <OFFSET>
Read PGR : YES	OK	OK*	OK*
Read PGR : NO	OK	OK	OK

(* - Emulator breaks into the monitor on accesses to emulation memory)

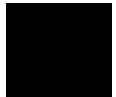
When you set the emulator to read PGR on address translation, the emulator should break to the monitor to get the current value of page register on address translation.

When you set the emulator not to read PGR on address translation, the emulator should use the copy of page registers which is renewed at breaking to the monitor or changing the value of page registers. In this case, the emulator does not break to the monitor.

Note



You may answer "NO" to "Read PGR registers" configuration item when you only use normal address mode in your program or the value of page registers is not changed after initializing while executing your program.



Note



When program execution should take place in real-time (refer to "Configuring the 70136 Emulator" chapter) and the emulator should break to the monitor to read page registers, the commands showing above which need physical to extended address conversion are not allowed in running user program. If you entered, the following error message will be shown:

```
ERROR:  Restricted to real time runs
```

Load/Dump Address

When you download programs into memory using "Memory Load" command, the emulator will interpret an address in the absolute file owing to the following configuration setting (refer to "20 bit address mode" configuration item in "Configuring the 70136 Emulator" chapter).

configuration	address mode
20 bit address mode : no	extended address
20 bit address mode : yes	physical address

When you dump memory to a host file using "Memory Store" command, the address information saved to host file is defined from the address expression used in the "Memory Load" command.

Address expression (in dump command)	address information (to a host file)
<EXT_ADDR> ("@e")	extended address
<PHY_ADDR> ("@p")	physical address
<SEGMENT>:<OFFSET>	physical address
No function code	physical address

Note



When you download the host file made by "Memory Store" command before, you should set the same "Address mode for file loading" configuration item that you enter the "Memory Store" command. Otherwise, the memory image is not the same as when you enter the "Memory Store" command to make the host file.

Note



When you download the host file with physical address information made by "Memory Store" command, you should set up the same value to page registers (PGR 1 - PGR 64) that you enter the "Memory Store" command. Otherwise, the memory image is not same as when you enter the "Memory Store" command to make the host file.

Run Commands

The following commands are included in run commands.

```
Processor, Go, Address
Processor, CMB, Go, Address
Processor, Step, Address
```

You can use the following address expression in run commands.

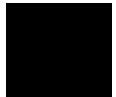
(refer to "Seg:off translation method" configuration item in "Configuring the 70136 Emulator" chapter)

	<EXT_ADDR>	<PHY_ADDR>	<SEGMENT>:<OFFSET>	No function code
Seg:off maxseg	ERROR	OK	OK	SAME AS <PHY_ADDR>
Seg:off minseg	ERROR	OK	OK	SAME AS <PHY_ADDR>
Seg:off curseg	ERROR	ERROR	OK	<OFFSET>

You should not use the extended address expression in run commands.

If you use extended address expression, the following error messages will be shown.

```
ERROR: Extended address can not be used
```



I/O Command

The following command is included in I/O command.

```
Processor, I/O
```

You can only use the I/O address expression ; this expression (0-0ffff hex) with no function code defines a 70136 I/O address.

Note



You should not access 70136 page registers (PGR 1 - PGR 64) with using "Processor, I/O" command. You should use "Register" command to access page registers.

Map Command

The following command is included in map command.

```
Config, Map, Modify
```

You can only use the extended address expression ; this expression (0-0ffffff hex and with "@e" function code) defines a 70136 extended address.

Define the data bus size

The data bus size for memory accesses can be defined in map command. For example, enter the following command to map memory (The extended address expression should be used in map command).

```
Config, Map, Modify
```

Using the arrow keys, move the cursor to the "Address range" field of term 1. Enter:

```
0..7ff@e
```

Move the cursor to the "Memory type" field of term 1, and press the TAB key to select the **erom** (emulation ROM) type. Move the cursor to the "Bus size" field of term 1, and enter "16" to map this emulation ROM with 16-bit data bus. Move the cursor to the "address range" field of term 2 and enter:

```
800..9ff@e
```

Move the cursor to the "memory type" field of term 2, and press the TAB key to select the **eram** (emulation ROM) type. Move the cursor to the "Bus size" field of term 2, and enter "8" to map this emulation RAM with 8-bit data bus.

To save your memory map, use the right arrow key or the **Enter** key to exit the field in the lower right corner. (The **End** key on Vectra keyboards moves the cursor directly to the last field.) The memory configuration display is shown in below.

```

Memory Map Configuration
Unmapped memory type eram
Term      Address Range      Memory Type  Bus Size
1 0..7ff0e      erom        16
2 800..9fff0e  eram        8
3 Empty        grd         16
4 Empty        grd         16
5 Empty        grd         16
6 Empty        grd         16
7 Empty        grd         16
8 Empty        grd         16
9 Empty        grd         16
10 Empty       grd         16
11 Empty       grd         16
12 Empty       grd         16
13 Empty       grd         16
14 Empty       grd         16
15 Empty       grd         16
16 Empty       grd         16
←↑→ : Interfield movement  Ctrl ↔ : Field editing  TAB : Scroll choices
STATUS: N70236--Emulation reset      Emulation trace halted
Use the TAB and Shift-TAB keys to pick bus size for mapped range.
  
```

The other memory ranges are mapped as target RAM with 16-bit data bus (if the data bus size is not specified in **map** command, the address ranges will be mapped with 16-bit data bus by default).

Note



The data bus size for memory accesses also can be defined from the BS8/BS16 input of the target system.

Refer to "Bus size signal for emulation/target memory" configuration items in "Configuring the 70136 Emulator" chapter.

Note 

The data bus size of I/O accesses (external I/O only) is defined from the BS8/BS16 input of the target system.

Breakpoints Command

The following commands are included in breakpoints command.

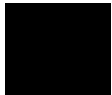
Breakpoints

You can use the following address expression in breakpoints command (refer to "Read PGR registers" configuration item in "Configuring the 70136 Emulator" chapter).

	<EXT_ADDR>	<PHY_ADDR>	<SEGMENT> : <OFFSET>
Read PGR : YES	OK	OK*	OK*
Read PGR : NO	OK	OK	OK

(* - Emulator breaks into the monitor on accesses to emulation memory)

When you set the emulator to read PGR on address translation, the emulator should break to the monitor to get the current value of page register to convert logical address to extended address using in emulation system.



When you set the emulator not to read PGR on address translation, the emulator should use the copy of page registers which is renewed at breaking to the monitor or changing the value of page registers. In this case, the emulator does not break to the monitor.

Note 

You may answer "NO" to "Read PGR registers" configuration item when you only use normal address mode in your program or the value of page registers is not changed after initializing while executing your program.

REGISTER NAMES and CLASSES (70136 Emulator)

The following register names and classes are used with the "Register Display/Modify" commands in 70136 emulator.

BASIC(*) class

Register name	Description
aw, bw cw, dw bp, ix, iy ds0, ds1, ss sp, pc, ps, psw	BASIC registers.

NOCLASS

Register name	Description
al, ah, bl, bh cl, ch, dl, dh	

PGR class (page registers)

Register name	Description
pgr1	PGR 1 register
pgr2	PGR 2 register
:	:
:	:
pgr63	PGR 63 register
pgr64	PGR 64 register
xam	XAM register (Read only)

REGISTER NAMES and CLASSES (70236/70236A Emulator)

The following register names and classes are used with the "Register Display/Modify" commands in 70236 emulator.

BASIC(*) class

Register name	Description
aw, bw cw, dw bp, ix, iy ds0, ds1, ss sp, pc, ps, psw	BASIC registers.

NOCLASS

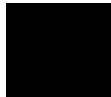
Register name	Description
al, ah, bl, bh cl, ch, dl, dh	

PGR class (Page registers)

Register name	Description
pgr1	PGR 1 register
pgr2	PGR 2 register
:	:
:	:
pgr63	PGR 63 register
pgr64	PGR 64 register
xam	XAM register (Read only)

SIO class (System I/O registers)

Register name	Description
bsel	Bank selection register
badr	Bank address register
brc	Baud rate counter
wmb0	Programmable wait, memory boundary 0 register
wcy1	Programmable wait, cycle 1 register
wcy0	Programmable wait, cycle 0 register
wac	Programmable wait, memory address control register
tcks	Timer clock selection register
sbc	Stand-by control register
refc	Refresh control register
wmb1	Programmable wait, memory boundary 1 register
wcy2	Programmable wait, cycle 2 register
wcy3	Programmable wait, cycle 3 register
wcy4	Programmable wait, cycle 4 register
sula	SCU low address register
tula	TCU low address register
iula	ICU low address register
dula	DMAU low address register
opha	On-chip peripheral high address register
opsel	On-chip peripheral selection register
sctl	System control register



ICU class (Interrupt Control Unit registers)

Register name	Description
imkw	Interrupt mask word register
irq	Interrupt request register (Read only)
iis	Interrupt in-service register (Read only)
ipol	Interrupt polling register (Read only)
ipfw	Interrupt priority and finish word register (Write only)
imdw	Interrupt mode word register (Write only)
iiw1	Interrupt initialize word 1 register (Write only)
iiw2	Interrupt initialize word 2 register (Write only)
iiw3	Interrupt initialize word 3 register (Write only)
iiw4	Interrupt initialize word 4 register (Write only)

Caution



When **ipol** register is displayed, interrupts are suspended until the FI command is published.

TCU class (Timer Control Unit registers)

Register name	Description
tct0	Timer/counter 0 register
tst0	Timer status 0 register (Read only)
tct1	Timer/counter 1 register
tst1	Timer status 1 register (Read only)
tct2	Timer/counter 2 register
tst2	Timer status 2 register (Read only)
tmd	Timer/counter mode register (Write only)

SCU class (Serial Control Unit registers)

Register name	Description	
srb	Serial receive data buffer	(Read only)
sst	Serial status register	(Read only)
stb	Serial transmit data buffer	(Write only)
scm	Serial command register	(Write only)
smd	Serial mode register	(Write only)
simk	Serial interrupt mask register	(Write only)

DMA71 class (DMA Control Unit registers (for uPD71071 mode))

Register name	Description	
dicm	DMA initialize register	(Write only)
dch	DMA channel register	
dbc/dcc0	DMA base/current count register channel 0	
dbc/dcc1	DMA base/current count register channel 1	
dbc/dcc2	DMA base/current count register channel 2	
dbc/dcc3	DMA base/current count register channel 3	
dba/dca0	DMA base/current address register channel 0	
dba/dca1	DMA base/current address register channel 1	
dba/dca2	DMA base/current address register channel 2	
dba/dca3	DMA base/current address register channel 3	
dmd0	DMA mode control register channel 0	
dmd1	DMA mode control register channel 1	
dmd2	DMA mode control register channel 2	
dmd3	DMA mode control register channel 3	
ddc	DMA device control register	
dst	DMA status register	(Read only)
dmk	DMA mask register	

DMA37 class (DMA Control Unit register (for uPD71037 mode))

Register name	Description
cmd	DMA read status/write command register
bank0	DMA bank register channel 0
bank1	DMA bank register channel 1
bank2	DMA bank register channel 2
bank3	DMA bank register channel 3
adr0	DMA current address register channel 0
adr1	DMA current address register channel 1
adr2	DMA current address register channel 2
adr3	DMA current address register channel 3
cnt0	DMA current count register channel 0
cnt1	DMA current count register channel 1
cnt2	DMA current count register channel 2
cnt3	DMA current count register channel 3
sfrq	Software DMA write request register (Write only)
smsk	DMA write single mask register (Write only)
mode	DMA write mode register
clbp	DMA clear byte pointer F/F (Write only)
init	DMA initialize register (Write only)
cmsk	DMA clear mask register (Write only)
amsk	DMA write all mask register bit (Write only)

Making Coordinated Measurements

Coordinated measurements are measurements synchronously made in multiple emulators or analyzers. Coordinated measurements can be made between HP 64700 Series emulators, which communicate over the Coordinated Measurement Bus (CMB). Coordinated measurements can also be made between an emulator and another instrument connected to the BNC connector.

This chapter will describe coordinated measurements made from the PC Interface which involve the emulator. These types of coordinated measurements are:

- Running the emulator on reception of the CMB /EXECUTE signal.
- Using the analyzer trigger to break emulator execution into the monitor.

Three signal lines on the CMB are active and serve the following functions:

/TRIGGER	Active low. The analyzer trigger line on the CMB and on the BNC serve the same logical purpose. They provide a means for the analyzer to drive its trigger signal out of the system, or for external trigger signals to arm the analyzer or break the emulator into its monitor.
READY	Active high. This line is for synchronized, multi-emulator start and stop. When you enable CMB run control interaction, all emulators must break to background on receipt of a false READY signal and will not return to foreground until this line is true.
/EXECUTE	Active low. This line serves as a global interrupt signal. On receipt of an enabled /EXECUTE signal, each emulator is to interrupt whatever it is doing and execute a previously defined process, such as run the emulator or start a trace measurement.

Running the Emulator at /EXECUTE

Before you can specify that the emulator run on receipt of the /EXECUTE signal, you must enable CMB interaction. To do this, select:

Config, General

Use the arrow keys to move the cursor to the "CMB Interaction? [n]" question, and type "y". Use the **Enter** key to exit out of the lower right-hand field in the configuration display.

To begin executing a program on receipt of the /EXECUTE signal, select:

Processor, CMB, Go

Now you may select either the current program counter ("Pc", in other words, the current PS:PC), or a specific address.

The command you enter is saved, and is executed when the /EXECUTE signal becomes active. Also, you will see the message "ALERT: CMB execute; run started".

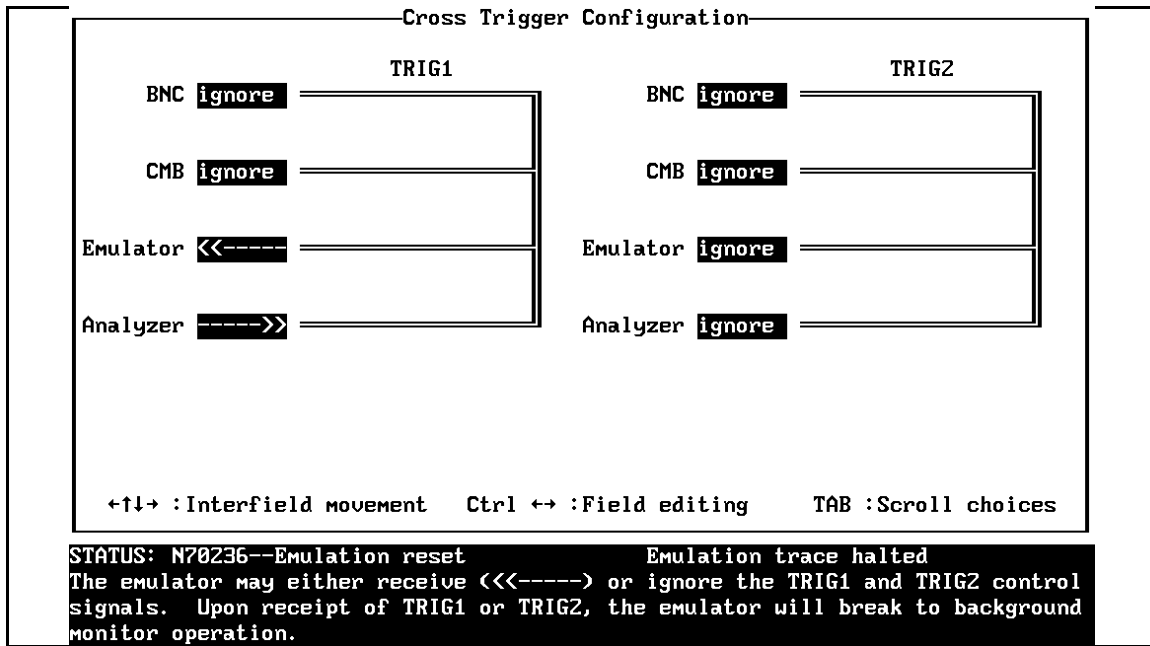
Breaking on the Analyzer Trigger

To break emulator execution into the monitor when the analyzer trigger condition occurs, you modify the trigger configuration. To access the trigger configuration, select:

Config, Trigger

The trigger configuration display contains two diagrams, one for each internal TRIG1 and TRIG2 signal.

To use the internal TRIG1 signal to connect the analyzer trigger to the emulator break line, move the cursor to the highlighted "Analyzer" field in the TRIG1 portion of the display. Use the **TAB** key to select the "----->>" arrow pointing from the analyzer to TRIG1. Next, move the cursor to the highlighted "Emulator" field and use the **TAB** key to select the arrow pointing toward the emulator (<<-----). This specifies that emulator execution will break into the monitor when the TRIG1 signal is driven. The trigger configuration display appears as follows:



Storing Memory Contents to an Absolute File

The "Getting Started" chapter shows you how to load absolute files into emulation or target system memory. You can also store emulation or target system memory to an absolute file with the following command.

`Memory, Store`

When you store memory using "Memory, Store" command, the address information saved to an absolute file is defined from the address expression used in the "Memory Store" command. refer to "Address Expression in Emulation Commands" section in this chapter.

Note



The first character of the absolute file name must be a letter. You can name the absolute file with a total of 8 alphanumeric characters. You also can include an extension of up to 3 alphanumeric characters. If the file is stored in HP 64000 format, its extension must be ".X".

Caution



The "Memory Store" command writes over an existing file if it has the same name that is specified with the command. You may wish to verify beforehand that the specified filename does not already exist.

File Format Readers

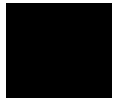
Introduction

The 70136 PC Interface is provided with the following "reader".

- Intel Object Module Format (OMF86) Reader
 - (This Reader is for the Intel OMF86 absolute file)
- NEC30 Reader
 - (This Reader is for the load module format file which is generated by NEC LK70136 linker for uPD70136)
- NEC33 Reader
 - (This Reader is for the extended load module format file which is generated by NEC EL70136 extended mode locator for uPD70136)
- HP64000 Reader

The Reader converts the file(s) into two files that are usable with the HP 64756 emulator. This means that you can use available language tools to create absolute files, then load those files into the emulator using the 70136 PC Interface.

The Reader can operate from within the PC Interface or as a separate process. When operating the Reader, it may be necessary to execute it as a separate process if there is not enough memory on your personal computer to operate the PC Interface and Reader simultaneously. You can also operate the reader as part of a "make file".



Using the OMF86, NEC30, NEC33 Reader

What the Reader Accomplishes

The Reader accepts as input an absolute file in the form "<file>.<ext>", and creates two new files that are used by the PC Interface: an "absolute" file, and an ASCII symbol file.

The Absolute File

During execution of the Reader, an absolute file (<file>.HPA) is created. This absolute file is a binary memory image which is optimized for efficient downloading into the emulator.

The ASCII Symbol File

The ASCII symbol file (<file>.HPS) produced by the Reader contains global symbols, module names, local symbols, and, when using applicable development tools such as a "C" Compiler, program line number. Local symbols evaluate to a fixed (static, not stack relative) address.

Note



You must use the required options for your specific language tools to include symbolic ("debug") information in the absolute file. The Reader will only convert symbol information that is present in the input absolute file.

The symbol file contains symbol and address information in the following form:

```

module_name1
module_name2
...
module_nameN
global_symbol1 0100:1234
global_symbol2 0100:5678
...
global_symbolN 0100:ABCD
|module_name|# 1234 0200:0872
|module_name|local_symbol1 0200:0653
|module_name|local_symbol2 0200:0872
...
|module name|local_symbolN 0200:0986

```

The space preceding module names is required. A single tab separates symbol and address.

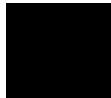
Each of the symbols is sorted alphabetically in the order: module names, global symbols, and local symbols.

The local symbols are scoped. This means that to access a variable named "count" in a function named "foo" in a source file module named "main.c", you would enter "main.c:foo.count". See table A-1.

Table A-1. How to Access Variables

Module Name	Function Name	Variable Name	You Enter:
MAIN.C	FOO	COUNT	MAIN.C:FOO.COUNT
MAIN.C	BAR	COUNT	MAIN.C:BAR.COUNT
MAIN.C	line number	23	MAIN.C: line 23

Line numbers will appear similar to a local symbol except that "local_symbolX" will be replaced by "#NNNNN" where NNNNN is a five digit decimal number. Line numbers should appear in ascending order.



Note



When the line number symbol is displayed in the emulator, it appears in brackets. Therefore, the symbol "modname:# 345" will be displayed as "modname:[345]" in mnemonic memory and trace list displays.

Line number symbols are accessed by entering the following on one line in the order shown:

module name
colon (:)
space
the word "line"
space
the decimal line number

For example:

```
MAIN.C: line 23
```

Location of the Reader Program

The Reader is located in the directory named `\hp64700\bin` by default, along with the PC Interface. This directory must be in the environment variable PATH for the Reader and PC Interface to operate properly. This is usually defined in the "`\autoexec.bat`" file. **The following examples assume that you have "`\hp64700\bin`" include in your PATH variable. If not, you must supply the directory name when executing the Reader program.**

Using the Reader from MS-DOS

The command names for the Reader are shown below.

Intel OMF86 Reader	RDOMF86.EXE
NEC30 Reader	RDNEC30.EXE
NEC33 Reader	RDNEC33.EXE

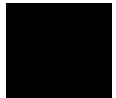
You can execute the Reader from the command line with the following command syntax:

```
C:\HP64700\BIN\<READER> [-q] [-u] [-m]  
<filename> <RETURN>
```

- <READER>** is the name of the command name for the Reader
- [-q]** Specifies the "quiet" mode. This option suppress the display of messages.
- [-u]** Specifies that the first leading underscore ("_") of a symbol is not removed.
- [-m]** (RDOMF86.EXE only) Specifies that the OMF86 Reader removes duplicate module names generated by some construction tools. Some tools enclose all of the functions and variables in a module within a block (or function) whose name is the same as that of the module (or source file). When this option is used, the Intel OMF86 Reader will ignore the first enclosing block in a module if its name matches the module name.
- <filename>** Specifies the name of the file containing the absolute program. You can include an extension in the file name.

The following commands will create the files "TESTPROG.HPA" and "TESTPROG.HPS".

```
ENTER: RDOMF86 TESTPROG.ABS  
ENTER: RDNEC30 TESTPROG.LNK  
ENTER: RDNEC33 TESTPROG.EXL
```



Using the Reader from the PC Interface

The 70136 PC Interface has a file format option under the "Memory Load" command.

After you select OMF86 as the file format, the Intel OMF86 Reader will operate on the file you specify. After the Reader completes successfully, the 70136 PC Interface will load the absolute and symbol files produced by the Reader.

To use the Reader from the PC Interface, follow these steps:

1. Start up the 70136 PC Interface.
2. Select "Memory, Load". The memory load menu will appear.
3. Specify the file format as "OMF86". This will appear as the default file format.
4. Specify the memory to be loaded (emulation, target, or both).
5. Specify to force the file format reader to regenerate the emulator absolute file (.HPA) and symbol database (.HPS) before loading the code. Normally, these files are only regenerated whenever the file you specify (the output of your language tools) is newer than the emulator absolute file and symbol database.
6. Specify that the OMF86 Reader removes duplicate module names generated by some construction tools. Some tools enclose all of the functions and variables in a module within a block (or function) whose name is the same as that of the module (or source file). When this option is used, the Intel OMF86 Reader will ignore the first enclosing block in a module if its name matches the module name.
7. Specify that the first leading underscore ("_") of a symbol is not removed.

8. Specify a file in Intel OMF86 format ("TESTFILE.OMF", for example). **The file extension can be something other than ".OMF", but ".HPA" or ".HPS" cannot be used.**

Using the Intel OMF86 file that you specify (TESTFILE.OMF, for example), the PC Interface performs the following:

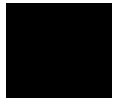
- It checks to see if two files with the same base name and extensions .HPS and .HPA already exist (for example, TESTFILE.HPS and TESTFILE.HPA).
- If TESTFILE.HPS and TESTFILE.HPA don't exist, the Intel OMF86 Reader produces them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.
- If TESTFILE.HPS and TESTFILE.HPA already exist but the create dates and times are earlier than the Intel OMF86 file creation date/time, the Intel OMF86 Reader recreates them. The new absolute file, TESTFILE.HPA, is then loaded into emulator.
- If TESTFILE.HPS and TESTFILE.HPA already exist but the dates and times are later than the creation date/time for the Intel OMF86 file, the current absolute file, TESTFILE.HPA, is then loaded into the emulator.

Note



Date/time checking only done within the PC Interface. When you run the Reader at the MS-DOS command line prompt, the Reader will always update the absolute and symbol files.

When the Reader operates on a file, a status message will be displayed indicating that it is reading an absolute file. When the Reader completes its processing, another message will be displayed indicating the absolute file is being loaded.



Note

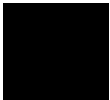
When you use NEC33 Reader and load an absolute file, you should configure that the emulator interprets address in the absolute file as extended address. Refer to "20 Bit Address Mode" section in "Configuring the 70136 Emulator" chapter.

If the Reader Won't Run

If your program is very large, the PC Interface may run out of memory while attempting to create the database file. If this occurs, exit the PC Interface and execute the Reader program at the MS-DOS command prompt.

Including Reader in a Make File

You may want to incorporate the "RDOMF86", "RDNEC30", or "RDNEC33" process as the last step in your "make" file, or as a step in your construction process, so as to eliminate the possibility of having to exit the PC Interface due to space limitations describe above. If the file with "-.HPA" and "-.HPS" extensions are not current, loading an absolute file will automatically create them.



Using the HP 64000 Reader

An HP 64000 “reader” is provided with the PC Interface. The HP 64000 Reader converts the files into two files that are usable with your emulator. This means that you can use available language tools to create HP 64000 absolute files, then load those files into the emulator using the PC Interface.

The HP 64000 Reader can operate from within the PC Interface or as a separate process. When operating the HP 64000 Reader, it may be necessary to execute it as a separate process if there is not enough memory on your personal computer to operate the PC Interface and HP 64000 Reader simultaneously. You can also operate the reader as part of a “make file.”

What the Reader Accomplishes

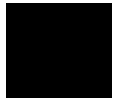
Using the HP 64000 files (<file.X>, <file.L>, <scr1.A>, <scr2.A>, ...) the HP 64000 Reader will produce two new files, an “absolute” file and an ASCII symbol file, that will be used by the PC Interface. These new files are named: “<file>.hpa” and “<file>.hps.”

The Absolute File

During execution of the HP 64000 Reader, an absolute file (<file>.hpa) is created. This absolute file is a binary memory image which is optimized for efficient downloading into the emulator.

The ASCII Symbol File

The ASCII symbol file (<file>.hps) produced by the HP 64000 Reader contains global symbols, module names, local symbols, and, when using applicable development tools such as a “C” compiler, program line numbers. Local symbols evaluate to a fixed (static, not stack relative) address.



Note



You must use the required options for your specific language tools to include symbolic (“debug”) information in the HP 64000 symbol files. The HP 64000 Reader will only convert symbol information present in the HP 64000 symbol files (<file.L>, <src1.A>, <src2.A>, ...).

The symbol file contains symbol and address information in the following form:

```
module_name1
module_name2
...
module_nameN
global_symbol1 0100:1234
global_symbol2 0100:5678
...
global_symbolN 0100:ABCD
|module_name1|# 1234 0200:0872
|module_name1|local_symbol1 0200:0653
|module_name1|local_symbol2 0200:0872
...
|module_name1|local_symbolN 0200:0986
```

Each of the symbols is sorted alphabetically in the order: module names, global symbols, and local symbols.

Line numbers will appear similar to a local symbol except that “local_symbolX” will be replaced by “#NNNNN” where NNNNN is a five digit decimal line number. The addresses associated with global and local symbols are specific to the processor for which the HP 64000 files were generated.

Note

If your emulator can store symbols internally, symbols will appear in disassembly. When the line number symbol is displayed in the emulator, it appears in brackets. Therefore, the symbol “MODNAME: line 345” will be displayed as “MODNAME:[345]” in mnemonic memory and trace list displays.

The space preceding module names is required. Although formatted for readability here, a single tab separates symbol and address.

The local symbols are scoped. This means that to access a variable named “count” in a source file module named “main.c,” you would enter “MAIN.C:COUNT” as shown below.

Table A-2. How to Access Variables

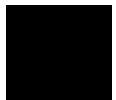
Module Name	Variable Name	You Enter:
MAIN.C	COUNT	MAIN.C:COUNT
MAIN.C	line number 23	MAIN.C: line 23

You access line number symbols by entering the following on one line in the order shown:

- module name
- colon (:)
- space
- the word “line”
- space
- the decimal line number

For example:

MAIN.C: line 23



Location of the HP 64000 Reader Program

The HP 64000 Reader is located in the directory named \hp64700\bin by default, along with the PC Interface. This directory must be in the environment variable PATH for the HP 64000 Reader and PC Interface to operate properly. The PATH is usually defined in the “\autoexec.bat” file.

The following examples assume that you have “\hp64000\bin” included in your PATH variable. If not, you must supply the directory name when executing the Reader program.

Using the Reader from MS-DOS

The command name for the HP 64000 Reader is **RHP64000.EXE**. To execute the Reader from the command line, for example, enter:

```
RHP64000 [-q] <filename>
```

[-q] This option specifies the “quiet” mode, and suppresses the display of messages.

<filename> This represents the name of the HP 64000 linker symbol file (file.L) for the absolute file to be loaded.

The following command will create the files “TESTPROG.HPA” and “TESTPROG.HPS”

```
RHP64000 TESTPROG.L
```

Using the Reader from the PC Interface

The PC Interface has a file format option under the “Memory Load” command. After you select HP64000 as the file format, the HP 64000 Reader will operate on the file you specify. After this completes successfully, the PC Interface will accept the absolute and symbol files produced by the Reader.

To use the Reader from the PC Interface:

1. Start up the PC Interface.
2. Select “Memory Load.” The memory load menu will appear.
3. Specify the file format as “HP64000.” This will appear as the default file format.
4. Specify the name of an HP 64000 linker symbol file (TESTFILE.L for example).

Using the HP 64000 file that you specify (TESTFILE.L, for example), the PC Interface performs the following:

- It checks to see if two files with the same base name and extensions .HPS and .HPA already exist (for example, TESTFILE.HPS and TESTFILE.HPA).
- If TESTFILE.HPS and TESTFILE.HPA don't exist, the HP 64000 Reader produces them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.
- If TESTFILE.HPS and TESTFILE.HPA already exist but the create dates and times are earlier than the HP 64000 linker symbol file creation date/time, the HP 64000 Reader recreates them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.
- If TESTFILE.HPS and TESTFILE.HPA already exist but the dates and times are later than the creation date and time for the HP 64000 linker symbol file, the HP 64000 Reader will not recreate TESTFILE.HPA. The current absolute file, TESTFILE.HPA, is then loaded into the emulator.

Note



Date/time checking is only done within the PC Interface. When running the HP 64000 Reader at the MS-DOS command line prompt, the HP 64000 Reader will always update the absolute and symbol files.

When the HP 64000 Reader operates on a file, a status message will be displayed indicating that it is reading an HP 64000 file. When the HP 64000 Reader completes its processing, another message will be displayed indicating the absolute file is being loaded.

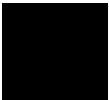
The PC Interface executes the Reader with the “-q” (quiet) option by default.

**If the Reader Won't
Run**

If your program is very large, the PC Interface may run out of memory while attempting to create the database file. If this occurs, you will need to exit the PC Interface and execute the program at the MS-DOS command prompt to create the files that are downloaded to the emulator.

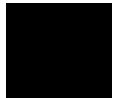
**Including RHP64000
in a Make File**

You may wish to incorporate the "RHP64000" process as the last step in your "make file," as a step in your construction process, to eliminate the possibility of having to exit the PC Interface due to space limitations describe above. If the files with ".HPA" and ".HPS" extensions are not current, loading an HP 64000 file will automatically create them.



Index

- A**
 - absolute files
 - <file>.hpa created by HP 64000 Reader, **A-9**
 - Intel OMF86, **A-1**
 - loading, **2-12**
 - NEC30, **A-1**
 - NEC33, **A-1**
 - storing, **5-20**
 - address expression
 - breakpoints command, **5-10**
 - emulation commands, **5-4**
 - I/O command, **5-8**
 - load/dump commands, **5-6**
 - map command, **5-8**
 - memory commands, **5-5**
 - run Commands, **5-7**
 - step command, **2-25**
 - Address expression syntax, **5-4**
 - Address Mode, emulator configuration, **4-13**
 - Address Syntax, **5-2**
 - analysis begin, **2-35**
 - analysis display, **2-36**
 - analysis specification
 - resetting the, **2-31**
 - trigger condition, **2-31**
 - analyzer
 - features of, **1-4**
 - analyzer, using the, **2-31**
 - apapter
 - PGA to QFP package of the uPD70236 and uPD70236, **1-3**
 - PLCC to QFP package of the uPD70136, **1-3**
 - ASCII symbol file (<file>.hps), **A-9**
 - assemblers, **2-9**
 - assembling and linking the getting started sample program, **2-6**
- B**
 - background, **1-6**
 - background monitor, **4-18**



- locating the, **4-18**
- pin state, **3-12/3-13**
- BNC connector, **5-17**
- break command, **2-26**
- breaking on analyzer trigger, **5-18**
- breakpoint interrupt instruction
 - software breakpoints (70136), **2-26**
- Breakpoints command
 - address expression, **5-10**
- breaks
 - software breakpoints, **2-26**
- breaks on ROM writes, **4-6**
- BRKXA and RETXA instructions, **1-8**
- BS8/BS16 input
 - emulation memory, **4-15**
 - I/O accesses, **4-16, 5-10**
 - memory accesses, **5-9**
 - target memory, **4-16**
- Bus size
 - map command, **5-8**

C caution statements

- change page registers after software breakpoints defined, **2-27**
- real-time dependent target system circuitry, **4-5**
- software breakpoint cmds. while running user code, **2-26**

cautions

- filenames in the memory store command, **5-20**
- installing the target system probe, **3-2**

characterization of memory, **2-9**

cim, Terminal Interface command, **2-27**

clock source

- external, **3-10, 4-4**
- in-circuit, **4-4**
- internal, **3-10, 4-4**

CMB (coordinated measurement bus), **5-17**

CMB interaction, **4-8**

CMB signals, **5-17**

commands (PC Interface), selecting, **2-8**

configuration (emulator), **4-1**

- loading, **4-21**
- storing, **4-21**

configuration options

- 20 bit address mode, **4-13**
- DMA cycle in background (70236 only), **4-14**
- FPU, **4-12**
- wait count of DMA (70236/70236A only), **4-14**
- accessing, **4-2**
- AEX signal, **4-12**
- background monitor location, **4-18**
- breaks on ROM writes, **4-6**
- bus sizing signal for emulation memory, **4-15**
- bus sizing signal for target memory, **4-16**
- CMB interaction, **4-8**
- emulator clock source, **4-4**
- foreground monitor address, **4-20**
- in-circuit, **3-10**
- lock READY signal, **4-10**
- monitor type, **4-18**
- read PGR registers, **4-11**
- real-time mode, **4-5**
- release bus by HOLD (70236/70236A only), **4-13**
- segment:offset translation method, **4-17**
- software breakpoints, **4-6**
- target interrupts, **4-9**
- target reset, **4-10**
- trace DMA cycles (70236/70236A only), **4-13**
- trace dummy HALT acknowledge (70236/70236A only), **4-15**
- trace refresh cycles (70236/70236A only), **4-14**
- configuration(hardware), installing the emulator, **2-2**
- coordinated measurements
 - break on analyzer trigger, **5-18**
 - definition, **5-17**
 - multiple emulator start/stop, **4-8**
 - run at /EXECUTE, **5-18**
- coprocessor
 - access emulation memory, **2-8, 3-10**
- copy memory command, **2-39**
- count, step command, **2-23**
- cur segment method, **4-17**
- D** device table, emulator, **2-7**
- displaying the trace, **2-36**
- DMA
 - external, **2-9**



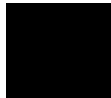
- DMA (70136), **1-7**
- DMA cycle
 - background (70236/70236A only), **4-14**
- dump command
 - interpret address, **5-6**
- E** emulation analyzer, **1-4**
- Emulation commands
 - address expression, **5-4**
- emulation memory
 - access by uPD72291 coprocessor, **2-8, 3-10**
 - note on target accesses, **2-9**
 - size of, **2-8**
- emulation monitor
 - foreground or background, **1-6**
- emulation RAM and ROM, **2-9**
- emulator
 - device table, **2-7**
 - feature list, **1-3**
 - purpose of, **1-1**
 - reset, **2-39**
 - running from target reset, **3-10/3-11**
 - status, **2-8**
 - supported microprocessor package, **1-3**
- emulator configuration
 - configuration options, **4-1**
 - loading, **4-21**
 - storing, **4-21**
- Emulator features
 - emulation memory, **1-4**
- emulator probe
 - installing, **3-2**
- eram, memory characterization, **2-9**
- erom, memory characterization, **2-9**
- Evaluation Chip, **1-8**
- EXECUTE
 - CMB signal, **5-17**
 - run at, **5-18**
- executing programs, **2-25**
- exiting the PC Interface, **2-40**
- external clock source, **4-4**

- F**
 - file formats
 - HP64000, **A-12**
 - file formats, absolute, **2-12**
 - find data in memory, **2-25**
 - foreground, **1-6**
 - foreground monitor, **4-18**
 - locating the, **4-20**
 - FPU, emulator configuration, **4-12**

- G**
 - getting started, **2-1**
 - prerequisites, **2-2**
 - global symbols, **2-15, 2-21**
 - grd, memory characterization, **2-9**
 - guarded memory accesses, **2-9**

- H**
 - halt instructions, **4-18**
 - hardware installation, **2-2**
 - HP 64000 Reader, **A-9**
 - using with PC Interface, **A-12**
 - HP 64000 Reader command (RHP64000.EXE), **A-12**
 - HP64000 file format, **A-12**
 - HPTABLES environment variable, **2-7**

- I**
 - I/O command
 - address expression, **5-8**
 - IEEE-695 reader, **A-2**
 - in-circuit configuration options, **3-10**
 - in-circuit emulation, **3-1**
 - installation
 - hardware, **2-2**
 - software, **2-2**
 - internal clock source, **4-4**
 - internal I/O register access, **1-7**
 - internal I/O registers
 - display, **1-7**
 - modify, **1-7**
 - interrupt
 - from target system, **3-10**
 - from target system (70136), **1-7**
 - from target system (70236), **1-7**
 - while stepping, **1-7**
 - interrupts
 - from target system, **4-9**



- L**
 - line numbers, **2-37**
 - linkers, **2-9**
 - linking the getting started sample program, **2-6**
 - load command
 - interpret address, **5-6**
 - load map, **2-9**
 - loading absolute files, **2-12**
 - local symbols, **2-17, 2-29, A-3, A-11**
 - locating the foreground monitor, **4-20**
 - location address
 - foreground monitor, **4-20**
 - locked, PC Interface exit option, **2-40**
 - logical run address, conversion from physical address, **4-17**
- M**
 - make file, **A-9**
 - Map command
 - command expression, **5-8**
 - data bus size, **5-8**
 - address expression, **5-8**
 - mapping memory, **2-8**
 - max segment method, **4-17**
 - memory
 - copy range, **2-39**
 - displaying in mnemonic format, **2-20**
 - modifying, **2-24**
 - reassignment of emulation memory blocks, **2-11**
 - searching for data, **2-25**
 - Memory commands
 - address expression, **5-5**
 - memory memory
 - access by coprocessor, **2-9**
 - memory, mapping, **2-8**
 - method, cur segment, **4-17**
 - method, max segment, **4-17**
 - method, min segment, **4-17**
 - microprocessor package, **1-3**
 - microprocessor socket
 - for QFP package of uPD70136, **1-3**
 - for QFP package of uPD70236 and uPD70236, **1-3**
 - min segment method, **4-17**
 - monitor
 - background, **4-18**

- foreground, **4-18**
- monitor block, **4-20**
- monitor program memory, size of, **2-8**

N

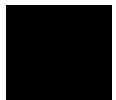
- NEC30
 - reader, **A-1**
- NEC33
 - reader, **A-1**
- NMI signal
 - from target system, **4-9**
- note
 - PC relative addressing in disassemble list, **2-37**
 - step address not allowed over 1M hex, **2-22**
- note statements
 - reassignment of emul. mem. blocks by mapper, **2-11**
- notes
 - absolute file names for stored memory, **5-20**
 - break to read page registers, **5-5, 5-10**
 - changing internal clock forces reset, **4-4**
 - CMB interaction enabled on execute command, **4-8**
 - coordinated measurements require background. monitor, **4-19**
 - date checking only in PC Interface, **A-13**
 - displaying complete traces, **2-37**
 - line number symbols in memory and trace listings, **A-4**
 - mapper terms deleted when monitor is relocated, **4-20**
 - mapping foreground monitor automatically, **4-19**
 - Reader only checks date/time within the PC Interface, **A-7**
 - register command, **2-21**
 - software breakpoints not allowed in target ROM, **2-27**
 - software breakpoints only at opcode addresses, **2-27**
 - step not accepted, **2-22**
 - symbolic information is required in absolute file, **A-2**
 - target accesses to emulation memory, **2-9**
 - use required options to include symbols, **A-10**
 - using NEC33 Reader, **A-8**
 - using terminal window to modify configuration, **4-3**
 - write to ROM analyzer status, **4-6**

O

- OMF86
 - reader, **A-1**

P

- PC Interface
 - exiting the, **2-40**



- HP 64000 Reader, **A-12**
- selecting commands, **2-8**
- starting the, **2-7**
- physical run address, conversion to logical run address, **4-17**
- pin protector
 - target system probe (70236/70236A only), **3-3**
- prerequisites for getting started, **2-2**

Q

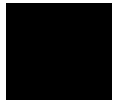
- qualifiers, analyzer status (70136 emulator), **2-32**
- qualifiers, analyzer status (70236 emulator), **2-33**

R

- RAM, mapping emulation or target, **2-9**
- RDY signal from target, **4-10**
- READY, CMB signal, **5-17**
- real-time mode, **4-5**
- register command, **2-21**
- register commands, **1-5**
- registers
 - classes (70136 emulator), **5-11**
 - classes (70236 emulator), **5-12**
 - names (70136 emulator), **5-11**
 - names (70236 emulator), **5-12**
- release bus by HOLD (70236/70236A only), **4-13**
- relocatable files, **2-9**
- RESET
 - from target system, **4-10**
- reset (emulator), **2-39**
 - running from target reset, **3-11**
- RESET signal, **3-10**
- resetting the analyzer specifications, **2-31**
- respond to target HLDRQ during background operation (70236/70236A only), **4-13**
- restrict to real-time runs
 - permissible commands, **4-5**
 - target system dependency, **4-5**
- ROM
 - writes to, **2-9**
- ROM, mapping emulation or target, **2-9**
- run address, conversion from physical address, **4-17**
- run at /EXECUTE, **5-18**
- Run commands
 - address expression, **5-7**

run from target reset, **3-10/3-11, 4-10**
running programs, **2-25**

- S** sample , **2-2**
- sample program
 - assembling, **2-6**
 - linking, **2-6**
- searching for data in memory, **2-25**
- selecting PC Interface commands, **2-8**
- simple trigger, specifying, **2-31**
- single step, **2-21**
- software breakpoint
 - 70136 breakpoint interrupt instruction, **4-7**
 - stepping, **1-8**
- software breakpoints, **2-26, 4-6**
 - and NMI, **2-27**
 - clearing, **2-30**
 - defining (adding), **2-29**
 - displaying, **2-29**
 - ignored, **2-28**
 - setting, **2-30**
- software installation, **2-2**
- specifications
 - analysis specification, **2-31**
- starting the trace, **2-35**
- status (analyzer) qualifiers, 70136 emulator, **2-32**
- status (analyzer) qualifiers, 70236 emulator, **2-33**
- status line, **2-8**
- step, **2-21**
- step count, **2-23**
- Stepping
 - at software breakpoint, **1-8**
 - BRKXA and RETXA instructions, **1-8**
- stepping failed, **1-7/1-8**
- symbols, **2-14**
 - .HPS file format, **A-10**
 - global, **2-21**
 - local, **2-29, A-9**
- system command
 - exit, **2-40**



- T**
 - target reset
 - running from, **3-11**
 - target reset, running from, **3-10**
 - target system
 - dependency on executing code, **4-5**
 - interface, **3-14, 3-17**
 - target system probe
 - cautions for installation, **3-2**
 - pin protector (70236/70236A only), **3-3**
 - target system RAM and ROM, **2-9**
 - trace
 - description of listing, **2-37**
 - displaying the, **2-36**
 - starting the, **2-35**
 - Trace list
 - extended address mode, **2-36**
 - normal address mode, **2-36**
 - PC relative addressing in disassemble list, **1-8**
 - trace signals, **2-31**
 - tracing dummy HALT acknowledge (70236/70236A only), **4-15**
 - tracing internal DMA cycles (70236/70236A only), **4-13**
 - tracing refresh cycles (70236/70236A only), **4-14**
 - tram, memory characterization, **2-9**
 - TRIG1, TRIG2 internal signals, **5-18**
 - trigger
 - breaking into monitor on, **5-18**
 - specifying a simple, **2-31**
 - trigger condition, **2-31**
 - TRIGGER, CMB signal, **5-17**
 - trom, memory characterization, **2-9**
- U**
 - undefined software breakpoint, **2-28, 4-7**
 - unlocked, PC Interface exit option, **2-40**
 - using the HP 64000 file reader, **A-9**
- W**
 - wait count of DMA (70236/70236A only), **4-14**
 - wait states, allowing the target system to insert, **4-10**
- Z**
 - zoom, window, **2-16, 2-20**