HP 64756/7

70136/70236 Emulator Terminal Interface

User's Guide

HEWLETT PACKARD

HP Part No. 64756-97011
Printed in U.S.A.
July 1994

Edition 4

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.

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

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	64756-97000, April 1990
Edition 2	64756-97003, August 1990
Edition 3	64756-97008, August
Edition 4	**64756-97011, July 1994**

Using this Manual

This manual will show you how to use the following emulators with the firmware resident Terminal 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 will:

- Show you how to use emulation commands by executing them on a sample program and describing their results.
- Show 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.
- Show you how to use the emulator in-circuit (connected to a target system).
- Describe the command syntax which is specific to the 70136 emulator.

This manual will not:

- Describe every available option to the emulation commands; this is done in the *HP 64700 Emulators Terminal Interface: User's Reference*.

Organization

Chapter 1 **Introduction to the 70136 Emulator.** This chapter briefly introduces you to the concept of emulation and lists the basic features of the 70136 emulator.

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, use software breakpoints, search memory for data, and perform coverage tests on emulation memory.

Chapter 3 **Emulation Topics**. This chapter shows you how to: restrict the emulator to real-time execution, use the analyzer trigger to cause breaks, and run the emulator from target system reset.

Chapter 4 **In-Circuit Emulation Topics**. This chapter shows you how to: install the emulator probe into a target system, select a target system clock source, allow the target system to insert wait states, and use the features which allow you to debug target system ROM.

Appendix A **70136 Emulator Specific Command Syntax**. This appendix describes the command syntax which is specific to the 70136 emulator. Included are: emulator configuration items, address syntax, display and access modes.

Appendix B **Using the Optional Foreground Monitor**. This appendix describes how to use the foreground monitor.

Appendix C **70136 Emulator Specific Error Messages.** This appendix describes the error messages which is specific to the 70136 emulator.

Contents

4 In-Circuit Emulation Topics

A 70136 Emulator Specific Command Syntax

Illustrations

Tables

Notes

1

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.

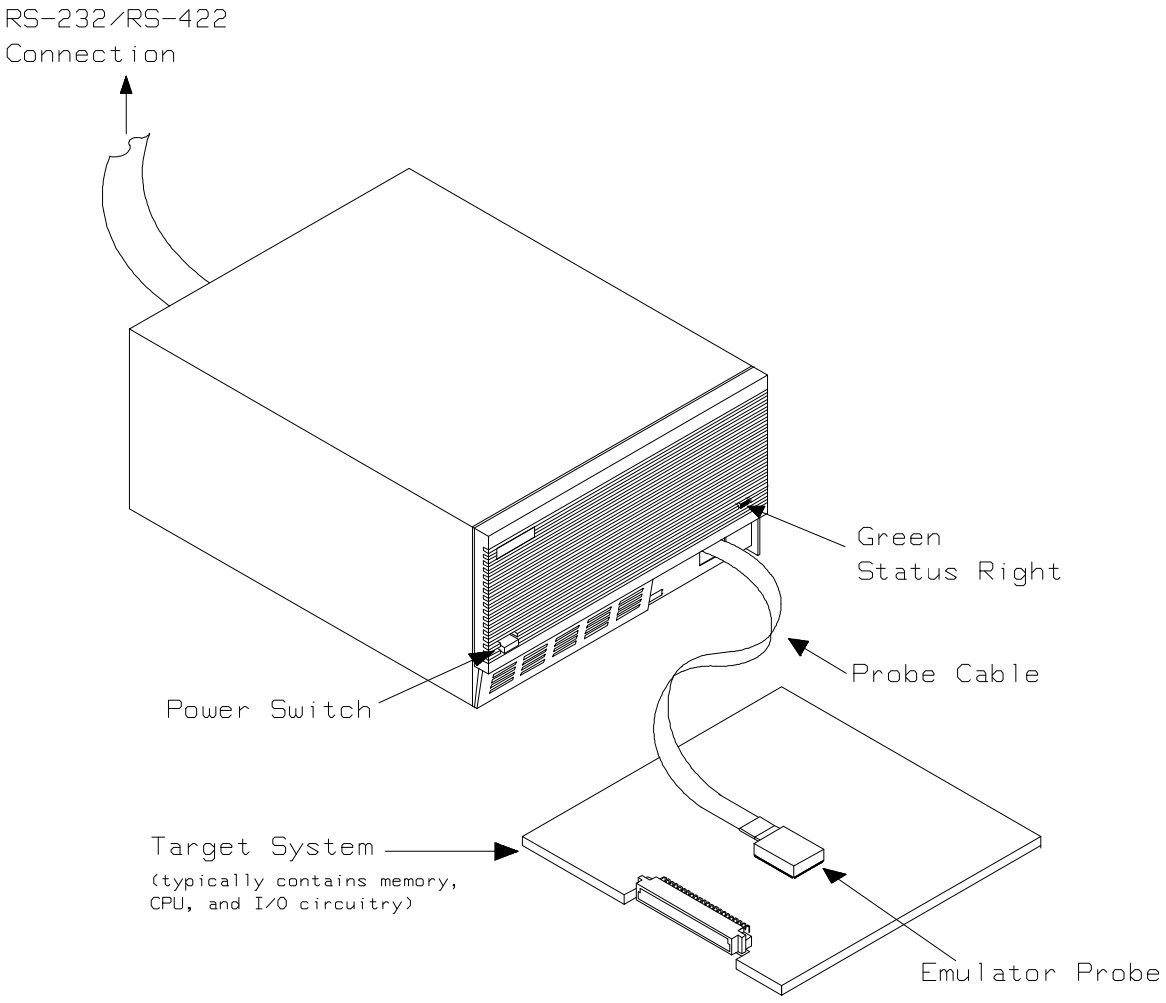


Figure 1-1. HP 64756/7 Emulator for uPD70136/70236

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
- 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.

2

Getting Started

Introduction

This chapter will lead you through a basic, step by step tutorial that shows how to use the HP 64756 emulator for the 70136 microprocessor.

This chapter will:

- Describe the sample program used for this chapter's examples.
- Show you how to use the "help" facility.
- Show you how to use the memory mapper.
- Show you how to enter emulation commands to view execution of the sample program. The commands described in this chapter include:
  - Displaying and modifying memory
  - Stepping
  - Displaying registers
  - Defining macros
  - Searching memory
  - Running
  - Breaking
  - Using software breakpoints
  - Copying memory
  - Testing coverage

Before You Begin

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

1. Completed hardware installation of the HP64700 emulator in the configuration you intend to use for your work:
   - Standalone configuration
   - Transparent configuration
   - Remote configuration
   - Local Area Network configuration

   References: *HP 64700 Series Installation/Service* manual

2. If you are using the Remote configuration, you must have completed installation and configuration of a terminal emulator program which will allow your host to act as a terminal connected to the emulator. In addition, you must start the terminal emulator program before you can work the examples in this chapter.

3. If you have properly completed steps 1 and 2 above, you should be able to hit <RETURN> (or <ENTER> on some keyboards) and get one of the following command prompts on your terminal screen:

```
U>
R>
M>
```

   If you do not see one of these command prompts, retrace your steps through the hardware and software installation procedures outlined in the manuals above, verifying all connections and procedural steps.

   In any case, you **must** have a command prompt on your terminal screen before proceeding with the tutorial.

A Look at the Sample Program

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

```
$MODV33
$OPTIMIZE

NAME    cmd_rds

PUBLIC  Msgs,Init,Cmd_Input,Msg_Dest

COMN    SEGMENT PARA COMMON 'COMN'
;*******************************************************
;  Command input byte.
;*******************************************************
Cmd_Input       DB      ?
;*******************************************************
;  Destination of the command message.
;*******************************************************
Msg_Dest        DB      20H DUP (?)
                EVEN
                DW      6FH DUP (?)    ; Stack area.
Stk             LABEL   WORD
COMN    ENDS

DATA    SEGMENT PARA PUBLIC 'DATA'
Msgs            LABEL   BYTE
Msg_A           DB      "Command A entered "
Msg_B           DB      "Command B entered "
Msg_I           DB      "Invalid Command   "
End_Msgs        LABEL   BYTE
DATA    ENDS

CODE    SEGMENT PARA PUBLIC 'CODE'
                ASSUME  PS:CODE,DS0:DATA,DS1:COMN,SS:COMN
;*******************************************************
;  The following instructions initialize segment
;  regsiters and set up the stack pointer.
;*******************************************************
Init:           MOV     AW,DATA
                MOV     DS0,AW
                MOV     AW,COMN
                MOV     DS1,AW
                MOV     SS,AW
                MOV     SP,OFFSET Stk
;*******************************************************
;  Clear previous command
;*******************************************************
Read_Cmd:       MOV     Cmd_Input,0
                NOP
;*******************************************************
;  Read command input byte.  If no command has been
;  entered, continue to scan for command input.
;*******************************************************
Scan:           MOV     AL,Cmd_Input
                CMP     AL,0
                BE      Scan
;*******************************************************
;  A command has been entered.  Check if it is
```

Figure 2-1. Sample Program Listing

```
;  command A, command B, or invalid.
;*******************************************************
;
Exe_Cmd:        CMP     AL,41H
                BE      Cmd_A
                CMP     AL,42H
                BE      Cmd_B
                BR      Cmd_I
;*******************************************************
;
;  Command A is entered.  CW = the number of bytes in
;  message A.  BP = location of the message.  Jump to
;  the routine which writes the message.
;*******************************************************
;
Cmd_A:          MOV     CW,Msg_B-Msg_A
                MOV     IX,OFFSET Msg_A
                BR      Write_Msg
;*******************************************************
;
;  Command B is entered.
;*******************************************************
;
Cmd_B:          MOV     CW,Msg_I-Msg_B
                MOV     IX,OFFSET Msg_B
                BR      Write_Msg
;*******************************************************
;
;  An invalid command is entered.
;*******************************************************
;
Cmd_I:          MOV     CW,End_Msgs-Msg_I
                MOV     IX,OFFSET Msg_I
;*******************************************************
;
;  Message is written to the destination.
;*******************************************************
;
Write_Msg:      MOV     IY,OFFSET Msg_Dest
                REP MOVBK       Msg_Dest,Msgs
;*******************************************************
;
;  The rest of the destination area is filled
;  with zeros.
;*******************************************************
;
Fill_Dest:      XOR     AL,AL
                MOV     CW,OFFSET Msg_Dest+20H
                SUB     CW,IY
                REP STM Msg_Dest
;*******************************************************
;
;  Go back and scan for next command
;*******************************************************
;
                BR      Read_Cmd
CODE    ENDS
                END     Init
```

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

Data Declarations

The area at DATA segment 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 see if a command is entered (a value other than 0H).

Processing Commands

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

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

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

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

The instructions at **Cmd_A**, **Cmd_B**, and **Cmd_I** each load register CW with the length of the message to be displayed and register IX with the starting location of the appropriate message. 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 20H bytes long.) Then, the program jumps back to read the next command.

The Destination Area

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

The program emulates a primitive command interpreter.

Using the "help" Facility

The HP 64700 Series emulator's Terminal Interface provides an excellent help facility to provide you with quick information on the various commands and their options. From any system prompt, you can enter "**help**" or "**?**" as shown below.

```
R>help

help  - display help information

  help <group>          - print help for desired group
  help -s <group>       - print short help for desired group
  help <command>        - print help for desired command
  help                  - print this help screen

--- VALID <group> NAMES ---
  gram      - system grammar
  proc      - processor specific grammar

  sys       - system commands
  emul      - emulation commands
  trc       - analyzer trace commands
  xtrc      - external trace analysis commands
  *         - all command groups
```

Commands are grouped into various classes. To see the commands grouped into a particular class, you can use the help command with that group. Viewing the group help information in short form will cause the commands or the grammar to be listed without any description.

For example, if you want to get some information for group gram, enter "**help gram**". Following help information should be displayed.

```
R>help gram
```

```
 gram - system grammar
------------------------------------------------------------------

--- SPECIAL CHARACTERS ---
  #  - comment delimiter    ;  - command separator    Ctl C - abort signal
  {} - command grouping     "" - ascii string         ''    - ascii string
  Ctl R - command recall    Ctl B - recall backwards

--- EXPRESSION EVALUATOR ---
  number bases:  t-ten   y-binary   q-octal   o-octal   h-hex
  repetition and time counts default to decimal - all else default to hex
  operators:     ()  ~  *  /  %  +  -  <  <  >  >  &  ^  |  &&

--- PARAMETER SUBSTITUTION ---
  &token& - pseudo-parameter included in macro definition
          - cannot contain any white space between & pairs
          - performs positional substitution when macro is invoked
  Example
     Macro definition:  mac getfile={load -hbs"transfer -t &file&"}
     Macro invocation:  getfile MYFILE.o
```

Help information exists for each command. Additionally, there is help information for each of the emulator configuration items.

Becoming Familiar with the System Prompts

A number of prompts are used by the HP 64700 Series emulators. Each of them has a different meaning, and contains information about the status of the emulator before and after the commands execute. These prompts may seem cryptic at first, but there are two ways you can find out what a certain prompt means if you are not familiar with it.

Using "help proc" to View Prompt Description

The first way you can find information on the various system prompts is to look at the **proc** help text.

```
R>help proc
```

```
--- Address format -----
 Memory address -- 32 bit (seg:off) logical or 20 bit physical (@p) or
     24 bit extended (@e) address
 IO address -- 16 bit address

 --- Emulation Prompt Status Characters ---
   U - running user code      M - running in monitor
   c - slow clock             w - waiting for target ready line
   R - emulation reset        r - target reset
   h - halted                 g - bus grant
   b - slow bus cycle         W - awaiting CMB ready
   T - awaiting target reset  ? - unknown state

 --- Analyzer STATUS Field Equates ---
   memrd   - memory read      memwr    - memory write
   iord    - I/O read         iowr     - I/O write
   fetch   - fetch cycle      exec     - first instruction byte
   mem     - memory access    io       - I/O access
   read    - read cycle       write    - write cycle
   intack  - interrupt ack    haltack  - halt ack
   cprd    - coproc read      cpwr     - coproc write
   memrdcp - mem rd for cp    memwrcp  - mem wr for cp
   coproc  - coproc cycle     memforcp - memory cycle for cp
   bs8     - bus size 8       bs16     - bus size 16
   nmladdr - normal address   extaddr  - extended address
   grd     - guarded access   wrrom    - write to ROM
   holdack - hold ack         mon      - monitor cycle
```

Using the Emulation Status Command (es) for Description of Current Prompt

When using the emulator, you will notice that the prompt changes after entering certain commands. If you are not familiar with a new prompt and would like information about that prompt only, enter the **es** (emulation status) command for more information about the status of the emulator.

```
U>es
N70136--Running user program
```

Initializing the Emulator

If you plan to follow this tutorial by entering commands on your emulator as shown in this chapter, verify that no one else is using the emulator. To initialize the emulator, enter the following command:

```
R>init
# Limited initialization completed
```

The **init** command with no options causes a limited initialization, also known as a warm start initialization. Warm start initialization does not affect system configuration. However, the **init** command will reset emulator and analyzer configurations. The **init** command:

- Resets the memory map.
- Resets the emulator configuration items.
- Resets the break conditions.
- Clears software breakpoints.

The **init** command does not:

- Clear any macros.
- Clear any emulation memory locations; mapper terms are deleted, but if you respecify the mapper terms, you will find that the emulation memory contents are the same.

Other Types of Initialization

There are two options to the **init** command which specify other types of initializations. The **-p** option specifies a powerup initialization, also known as a cold start initialization. The cold start initialization sequence includes the emulator, analyzer, system controller, and communications port initialization; additionally, performance verification tests are run.

The **-c** option also specifies a cold start initialization, except that performance verification tests are not run.

Mapping Memory

Depending on the memory board, emulation memory consists of 128K, 512K, 1M or 2M bytes, mappable in 256 byte blocks. The monitor occupies 4K bytes, leaving 124K , 508K, 1020K or 2044K bytes of emulation memory which you may use. The emulation memory system does not introduce wait states.

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 **rom** break condition is enabled. Memory is mapped with the **map** command. To view the memory mapping options, enter:

```
M>help map
```

```
map - display or modify the processor memory map

  map                              - display the current map structure
  map <addr>..<addr> <type> <attrib> - define address range as memory type
  map other <type> <attrib>        - define all other ranges as memory type
  map -d <term#>                   - delete specified map term
  map -d *                         - delete all map terms

--- VALID <type> OPTIONS ---
  eram  -  emulation ram
  erom  -  emulation rom
  tram  -  target ram
  trom  -  target rom
  grd   -  guarded memory

--- VALID emulation memory <attrib> OPTIONS ---
  <none> - bus size is 16
  8      - bus size is 8
  16     - bus size is 16
```

Enter the **map** command with no options to view the default map structure.

```
M>map
```

```
# remaining number of terms   : 16
# remaining emulation memory  : 1f000h bytes
map  other tram
```

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 memory locations your program will occupy in memory. A linker load map listing for the sample program is shown below.

```
Command line: ldv33  -h -L -o cmd_rds.XCommand line: ldv33  -c cmd_rds.k -h -L -o
cmd_rds.X

SEG /CODE=400h
SEG /DATA=600h
SEG /COMN=800h
LOAD cmd_rds.o
END

OUTPUT MODULE NAME:   cmd_rds
OUTPUT MODULE FORMAT: HP64000 absolute


MODULE SUMMARY
--------------

MODULE    SEGMENT   CLASS     HP SECTION  START  END

cmd_rds   /users/jlomktg/miizuka/emul/V53A/termdemo/cmd_rds.o
          CODE      CODE      PROG        00400  0044D
          COMN      COMN      DATA        00800  008FF
          DATA      DATA      COMMON      00600  00635

SEGMENT SUMMARY
---------------

SEGMENT   CLASS     GROUP     START  END    LENGTH  ALIGNMENT     COMBINE

CODE      CODE                00400  0044D  0004E   Paragraph     Public
DATA      DATA                00600  00635  00036   Paragraph     Public
COMN      COMN                00800  008FF  00100   Paragraph     Common
??SEG                         00000  00000  00000   Paragraph     Public
??DATA1   ??INIT              00000  00002  00003   Byte          Common
```

From the load map listing, you can see that the sample program occupies three address range. The program area, which contains the opcodes and operands which make up the sample program, occupies locations 400 through 44D hex. The data area, which contains the ASCII values of the messages the program transfers, is occupies locations 600 through 635 hex. The destination area, which contains the command input byte and the locations of the message destination, occupies locations 800 through 8FF hex.

Since the program writes to the destination locations, the mapper block of destination area should not be characterized as ROM memory. Enter the following command to map memory for the sample program, and display the memory map.

```
R>map 0..7ff erom
R>map 800..9ff eram
R>map
```

```
# remaining number of terms   : 14
# remaining emulation memory  : 1e600h bytes
map  0000000@e..00007ff@e  erom      # term 1
map  0008000@e..00009ff@e  eram      # term 2
map  other tram
```

When mapping memory for your target system programs, you may wish to characterize emulation memory locations containing programs and constants (locations which should not be written to) as ROM. This will prevent programs and constants from being written over accidentally, and will cause breaks when instructions or commands attempt to do so (if the **rom** break condition is enabled).

Getting the Sample Program into Emulation Memory

This section assumes you are using the emulator in one of three configurations:

1. Connected only to a terminal, which is called the *standalone* configuration. In the standalone configuration, you must modify memory to load the sample program.

2. Connected between a terminal and a host computer, which is called the *transparent* configuration. In the transparent configuration, you can load the sample program by downloading from the "other" port.

3. Connected to a host computer and accessed via a terminal emulation program (for example, the terminal window of the PC Interface). Configurations in which the emulator is connected to, and accessed from, a host computer are called *remote* configurations. In the remote configuration, you can load the sample program by downloading from the same port.

Standalone Configuration

If you are operating the emulator in the standalone configuration, the only way to get the sample program into emulation memory is by modifying emulation memory locations with the **m** (memory display/modification) command.

You can enter the sample program into memory with the **m** command as shown below.

```
R>m -db 400=0b8,60,0,8e,0d8,0b8,80,0,8e,0c0,8e,0d0,0bc,0,1
R>m -db 40f=26,0c6,6,0,0,0,90,26,0a0,0,0,3c,0,74,0f8
R>m -db 41e=3c,41,74,7,3c,42,74,0a,0eb,10,0b9
R>m -db 429=12,0,0be,0,0,0eb,0e,0b9,12,0,0be,12,0,0be,06,0b9,12,0
R>m -db 43b=0be,24,0,0bf,1,0,0f3,0a4,32,0c0,0b9
R>m -db 446=21,0,2b,0cf,0f3,0aa,0eb,0c1
R>m -db 600="Command A entered Command B entered Invalid command   "
```

After entering the opcodes and operands, you would typically display memory in mnemonic format to verify that the values entered are correct (see the example below). If any errors exist, you can modify individual locations. Also, you can use the **cp** (copy memory) command if, for example, a byte has been left out, but the locations which follow are correct.

Note 

Be careful about using this method to enter programs from the listings of relocatable source files. If source files appear in relocatable sections, the address values of references to locations in other relocatable sections are not resolved until link-time. The correct values of these address operands will not appear in the assembler listing.

Transparent Configuration

If your emulator is connected between a terminal and a host computer, you can download programs into memory using the **load** command with the **-o** (from other port) option. The **load** command will accept absolute files in the following formats:

- HP absolute.
- Intel hexadecimal.
- Tektronix hexadecimal.
- Motorola S-records.

The examples which follow will show you the methods used to download HP absolute files and the other types of absolute files.

HP Absolutes

Downloading HP format absolute files requires the **transfer** protocol. The example below assumes that the **transfer** utility has been installed on the host computer (HP 64884 for HP 9000 Series 500, or HP 64885 for HP 9000 Series 300).

Note 

Notice that the transfer command on the host computer is terminated with the <ESCAPE>**g** characters; by default, these are the characters which temporarily suspend the transparent mode to allow the emulator to receive data or commands.

```
R>load -hbo <RETURN> <RETURN>
$ transfer -rtb cmd_rds.X <ESCAPE>g
####
R>
```

Other Supported Absolute Files

The example which follows shows how to download Intel hexadecimal files, but the same method (and a different **load** option) can be used to load Tektronix hexadecimal and Motorola S-record files as well.

```
        R>load -io <RETURN> <RETURN>
        $ cat ihexfile <ESCAPE>g
#####
Data records = 00003  Checksum error = 00000
R>
```

Remote Configuration

If the emulator is connected to a host computer, and you are accessing the emulator from the host computer via a terminal emulation program, you can also download files with the **load** command. However, in the remote configuration, files are loaded from the same port that commands are entered from. For example, if you wish to download a Tektronix hexadecimal file from a Vectra personal computer, you would enter the following commands.

```
R>load -t <RETURN>
```

After you have entered the **load** command, exit from the terminal emulation program to the MS-DOS operating system. Then, copy your hexadecimal file to the port connected to the emulator, for example:

```
C:\copy thexfile com1: <RETURN>
```

Now you can return to the terminal emulation program and verify that the file was loaded.

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.

```
R>m -dm 400..44d

000400@p  -             MOV AW,#0060
000403@p  -             MOV DS0,AW | MOV AW,#0080
000408@p  -             MOV DS1,AW | MOV SS,AW | MOV SP,#010
00040f@p  -             MOV DS1:0000,#00
000415@p  -             NOP
000416@p  -             MOV AL,DS1:0000
00041a@p  -             CMP AL,#00
00041c@p  -             BE/Z 000416
00041e@p  -             CMP AL,#41
000420@p  -             BE/Z 000428
000422@p  -             CMP AL,#42
000424@p  -             BE/Z 000430
000426@p  -             BR SHORT 000438
000428@p  -             MOV CW,#0012
00042b@p  -             MOV IX,#0000
00042e@p  -             BR SHORT 00043e
000430@p  -             MOV CW,#0012
000433@p  -             MOV IX,#0012
000436@p  -             BR SHORT 00043e
000438@p  -             MOV CW,#0012
00043b@p  -             MOV IX,#0024
00043e@p  -             MOV IY,#0001
000441@p  -             REP/E/Z MOVBKB
000443@p  -             XOR AL,AL
000445@p  -             MOV CW,#0021
000448@p  -             SUB CW,IY
00044a@p  -             REP/E/Z STMB
00044c@p  -             BR SHORT 00040f
```

If you display memory in mnemonic format and do not recognize the instructions listed or see some illegal instructions or opcodes, go back and make sure the memory locations you are trying to display have been mapped. If the memory map is not the problem, recheck the linker load map listing to verify that the absolute addresses of the program agree with the locations you are trying to display.

Stepping Through the Program

The emulator allows you to execute one instruction or a number of instructions with the **s** (step) command. Enter the **help s** to view the options available with the step command.

```
R>help s
```

```
s - step emulation processor

  s                     - step one from current PC
  s <count>             - step <count> from current PC
  s <count> $           - step <count> from current PC
  s <count> <addr>      - step <count> from <addr>
  s -q <count> <addr>   - step <count> from <addr>, quiet mode
  s -w <count> <addr>   - step <count> from <addr>, whisper mode

--- NOTES ---
  STEPCOUNT MUST BE SPECIFIED IF ADDRESS IS SPECIFIED!
  If <addr> is not specified, default is to step from current PC.
  A <count> of 0 implies step forever.
```

A step count of 0 will cause the stepping to continue "forever" (until some break condition, such as "write to ROM", is encountered, or until you enter <CTRL>**c**). The following command will step from the first address of the sample program.

```
R>s 1 0:400
```

```
00000:00400  -           MOV AW,#0060
PC = 00000:00403
```

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.

Displaying Registers

The step command shown above executed a MOV AW,#0060H instruction. Enter the following command to view the contents of the registers.

```
M>reg *

 reg ps=0000 pc=0403 psw=f002 aw=0060 bw=4ec0 cw=0000 dw=ff80 sp=0fff bp=1566
reg ix=fffd iy=0180 ds0=0000 ds1=0000 ss=0000
```

The register contents are displayed in a "register modify" command format. This allows you to save the output of the **reg** command to a command file which may later be used to restore the register contents. (Refer to the **po** (port options) command description in the *Terminal Interface: User's Reference* for more information on command files.)

You can also display page registers. Refer to the "70136 Emulator Specific Command Syntax" appendix for more information on the register names and classes.

Combining Commands

More than one command may be entered in a single command line if the commands are separated by semicolons (;). For example, you could execute the next instruction(s) and display the registers by entering the following.

```
M>s;reg

00000:00403  -                MOV DS0,AW | MOV AW,#0080
 PC = 00000:00408
 reg ps=0000 pc=0408 psw=f002 aw=0080 bw=4ec0 cw=0000 dw=ff80 sp=0fff bp=1566
 reg ix=fffd iy=0180 ds0=0060 ds1=0000 ss=0000
```

The sample above shows you that MOV DS0,AW and MOV AW,#0080H are executed by step command. Refer to the Note above.

Using Macros

Suppose you want to continue stepping through the program, displaying registers after each step. You could continue entering **s** command followed by **reg** command, but you may find this tiresome. It is easier to use a macro to perform a sequence of commands which will be entered again and again.

Macros allow you to combine and store commands. For example, to define a macro which will display registers after every step, enter the following command.

```
M>mac st={s;reg}
```

Once the **st** macro has been defined, you can use it as you would any other command.

```
M>st
```

```
 # s ; reg
00000:00408  -                MOV DS1,AW | MOV SS,AW | MOV SP,#0100
PC = 00000:0040f
reg ps=0000 pc=040f psw=f002 aw=0080 bw=4ec0 cw=0000 dw=ff80 sp=0100 bp=1566
reg ix=fffd iy=0180 ds0=0060 ds1=0080 ss=0080
```

Command Recall

The command recall feature is yet another, easier way to enter commands again and again. You can press <CTRL>**r** to recall the commands which have just been entered. If you go past the command of interest, you can press <CTRL>**b** to move forward through the list of saved commands. To continue stepping through the sample program, you could repeatedly press <CTRL>**r** to recall and <RETURN> to execute the **st** macro.

Repeating Commands

The **rep** command is also helpful when entering commands repetitively. You can repeat the execution of macros as well commands. For example, you could enter the following command to cause the **st** macro to be executed four times.

```
M>rep 4 st
```

```
 # s ; reg
00000:0040f  -                MOV DS1:0000,#00
PC = 00000:00415
reg ps=0000 pc=0415 psw=f002 aw=0080 bw=4ec0 cw=0000 dw=ff80 sp=0100 bp=1566
reg ix=fffd iy=0180 ds0=0060 ds1=0080 ss=0080
# s ; reg
00000:00415  -                NOP
PC = 00000:00416
reg ps=0000 pc=0416 psw=f002 aw=0080 bw=4ec0 cw=0000 dw=ff80 sp=0100 bp=1566
reg ix=fffd iy=0180 ds0=0060 ds1=0080 ss=0080
# s ; reg
00000:00416  -                MOV AL,DS1:0000
PC = 00000:0041a
reg ps=0000 pc=041a psw=f002 aw=0000 bw=4ec0 cw=0000 dw=ff80 sp=0100 bp=1566
reg ix=fffd iy=0180 ds0=0060 ds1=0080 ss=0080
# s ; reg
00000:0041a  -                CMP AL,#00
PC = 00000:0041c
reg ps=0000 pc=041c psw=f046 aw=0000 bw=4ec0 cw=0000 dw=ff80 sp=0100 bp=1566
reg ix=fffd iy=0180 ds0=0060 ds1=0080 ss=0080
```

Command Line Editing

The terminal interface supports the use of HP-UX **ksh(1)**-like editing of the command line. The default is for the command line editing feature to be disabled to be compatible with earlier versions of the interface. Use the **cl** command to enable command line editing.

```
M>cl -e
```

Refer to "Command Line Editing" in the *HP 64700-Series Emulators Terminal Interface Reference* for information on using the command line editing feature.

Modifying Memory

The preceding step and register commands show the sample program is executing Scan loop, where it continually reads the command input byte to check if a command had been entered. Use the **m** (memory) command to modify the command input byte.

```
M>m 800=41
```

To verify that 41H has been written to 800H, enter the following command.

```
M>m -db 800
000800@p..000800@p  41
```

When memory was displayed in byte format earlier, the display mode was changed to "byte". The display and access modes from previous commands are saved and they become the defaults.

Specifying the Access and Display Modes

There are a couple different ways to modify the display and access modes. One is to explicitly specify the mode with the command you are entering, as with the command **m -db 800**. The **mo** (display and access mode) command is another way to change the default mode. For example, to display the current modes, define the display mode as "word", and redisplay 800H, enter the following commands.

```
M>mo
mo -ab -db

M>mo -dw
M>m 800
000800@p..000800@p  0041
```

To continue the rest of program.

```
M>r
U>
```

Display the **Msg_Dest** memory locations (destination of the message, 801H) to verify that the program moved the correct ASCII bytes. At this time we want to see correct byte value, so "-db" option (display with byte) is used.

```
U>m  -db 801..820
000801@p..000810@p  43 6f 6d 6d 61 6e 64 20 41 20 65 6e 74 65 72 65
000811@p..000820@p  64 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Searching Memory for Data

The **ser** (search memory for data) command is another way to verify that the program did what it was supposed to do.

```
U>ser 800..820="Command A entered "
pattern match at address: 000801@p
```

If any part of the data specified in the **ser** command is not found, no match is displayed (No message displayed).

Breaking into the Monitor

You can use the break command (**b**) command to generate a break to the background monitor. While the break will occur as soon as possible, the actual stopping point may be many cycles after the break request (depend on the type of instruction being executed and whether the processor is in a hold state).

```
U>b
M>
```

Using Software Breakpoints

You can stop program execution at specific address by using **bp** (software breakpoint) 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.

If the breakpoint interrupt instruction (F1 hex) was not inserted as the result of **bp** command (in other words, it is part of the user program), the "Undefined software breakpoint" message is displayed.

Caution 

Software breakpoints should not be set, enabled, disabled, or removed 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 code (F1 hex), you cannot define software breakpoints in target ROM.
You can, however, copy target ROM into emulation memory (see the "Target ROM Debug Topics" section of the "In-Circuit Emulation" chapter).

Note

Software breakpoint will be ignored, when software breakpoint and other emulation break (for example, break command (**b**), simple trigger command (**tg**), etc.) occur at the same time. Refer to *HP 64700 Emulators Terminal Interface: User's Reference* manual.

Displaying and Modifying the Break Conditions

Before you can define software breakpoints, you must enable software breakpoints with the **bc** (break conditions) command. To view the default break conditions and change the software breakpoint condition, enter the **bc** command with no option. This command displays current configuration of break conditions.

```
M>bc
```

```
bc -d bp #disable
bc -e rom #enable
bc -d bnct #disable
bc -d cmbt #disable
bc -d trig1 #disable
bc -d trig2 #disable
```

To enable the software break point feature enter

```
M>bc -e bp
```

Defining a Software Breakpoint

Now that the software breakpoint feature is enabled, you can define software breakpoints. Enter the following command to break on the address of the **Cmd_I** (address 438H) label.

```
M>bp 438
M>bp
bp 000438@p #enabled
```

Run the program, and verify that execution broke at the appropriate address.

```
M>r 0:400
U>m 800=43
!ASYNC_STAT 615! Software breakpoint: 00000:000438
M>st
# s;reg
00000:00438  -          MOV CW,#0012
PC = 00000:0043b
reg ps=0000 pc=043b psw=f002 aw=0043 bw=0000 cw=0012 dw=ff80 sp=0100 bp=0000
reg ix=0012 iy=0021 ds0=0060 ds1=0080 ss=0080
```

When a breakpoint is hit, it becomes disabled. You can use the **-e** option to the **bp** command to re-enable the software breakpoint.

```
M>bp
### BREAKPOINT FEATURE IS ENABLED ###
bp 000438@p #disabled
M>bp -e 438
M>bp
### BREAKPOINT FEATURE IS ENABLED ###
bp 000438@p #enabled
M>r
U>m 800=43
!ASYNC_STAT  615! Software breakpoint: 000438@p
M>bp
### BREAKPOINT FEATURE IS ENABLED ###
bp 000438@p #disabled
```

Refer to the "ADDRESS" and "ADDRESS_EXPRESSION" section in the "70136 Emulator Specific Command Syntax" appendix.

Predefined Trace Labels

Three trace labels are predefined in the 70136 emulator. You can view these labels by entering the **tlb** (trace label) command with no options.

```
M>tlb
```

```
#### Emulation trace labels
tlb addr 0..23
tlb data 24..39
tlb stat 40..53
```

Predefined Status Equates

Common values for the 70136 status trace signals have been predefined. You can view these predefined equates by entering the **equ** command with no options.

```
M>equ
```

```
### Equates ###
equ bs16    = 0xx xx1x xxxx xxxxy   ;bus size 16
equ bs8     = 0xx xx0x xxxx xxxxy   ;bus size 8
equ coproc  = 0x0 xxxx x101 x0xxy   ;co-processor access
equ cprd    = 0x0 xxxx x101 x01xy   ;co-processor read
equ cpwr    = 0x0 xxxx x101 x00xy   ;co-processor write
equ exec    = 0x0 xxxx x0xx xxxxy   ;executed code
equ extaddr = 0xx 1xxx xxxx xxxxy   ;extended address mode
equ fetch   = 0x0 xxxx 1100 x11xy   ;program fetch
equ grd     = 0xx xxxx 0xxx x1xxy   ;guarded access
equ haltack = 0x0 xxxx x111 x00xy   ;halt acknowledge
equ holdack = 0x1 xxxx xxxx xxxxy   ;hold acknowledge
equ intack  = 0x0 xxxx x100 x01xy   ;interrupt acknowledge
equ io      = 0x0 xxxx x110 x0xxy   ;I/O access
equ iord    = 0x0 xxxx x110 x01xy   ;I/O read
equ iowr    = 0x0 xxxx x110 x00xy   ;I/O write
equ mem     = 0x0 xxxx 1110 x1xxy   ;memory access
equ memforcp= 0x0 xxxx 1101 x1xxy   ;memory access for cp
equ memrd   = 0x0 xxxx 1110 x11xy   ;memory read
equ memrdcp = 0x0 xxxx 1101 x11xy   ;memory read for cp
equ memwr   = 0x0 xxxx 1110 x10xy   ;memory write
equ memwrcp = 0x0 xxxx 1101 x10xy   ;memory write for cp
equ mon     = 0xx x0xx xxxx xxxxy   ;monitor cycle
equ nmladdr = 0xx 0xxx xxxx xxxxy   ;normal address mode
equ read    = 0x0 xxxx x1xx xx1xy   ;read cycle
equ write   = 0x0 xxxx x1xx xx0xy   ;write cycle
equ wrrom   = 0xx xxx0 xxxx xx0xy   ;write to ROM
```

These equates may be used to specify values for the **stat** trace label when qualifying trace conditions.

In the 70236 emulator, the common values for the status trace signals have been predefined as follows:

```
### Equates ###
equ bs16    = 0xxx xx1x xxxx xxxxy   ;bus size 16
equ bs8     = 0xxx xx0x xxxx xxxxy   ;bus size 8
equ coproc  = 0xx0 xxxx x101 00xxy   ;co-processor access
equ cprd    = 0xx0 xxxx x101 001xy   ;co-processor read
equ cpwr    = 0xx0 xxxx x101 000xy   ;co-processor write
equ dma     = 0xx0 xxxx x110 11xxy   ;DMA cycle
equ dmard   = 0xx0 xxxx x110 111xy   ;DMA read cycle
equ dmawr   = 0xx0 xxxx x110 110xy   ;DMA write cycle
equ dmac    = 1xxx xxxx xxxx xxxxy   ;DMA cascade
equ eio     = 0xx0 xxxx x110 00xxy   ;external I/O access
equ eiord   = 0xx0 xxxx x110 001xy   ;external I/O read
equ eiowr   = 0xx0 xxxx x110 000xy   ;external I/O write
equ exec    = 0xx0 xxxx x0xx xxxxy   ;executed code
equ extaddr = 0xxx 1xxx xxxx xxxxy   ;extended address mode
equ fetch   = 0xx0 xxxx 1100 011xy   ;program fetch
equ grd     = 0xxx xxxx 0xxx xxxxy   ;guarded access
equ haltack = 0xx0 xxxx x111 000xy   ;halt acknowledge
equ holdack = 0xx1 xxxx xxxx xxxxy   ;hold acknowledge
equ intacki = 0xx0 xxxx x100 101xy   ;interrupt acknowledge (ICU)
equ intacks = 0xx0 xxxx x100 001xy   ;interrupt acknowledge (SLAVE)
equ iio     = 0xx0 xxxx x110 10xxy   ;internal I/O access
equ iiord   = 0xx0 xxxx x110 101xy   ;internal I/O read
equ iiowr   = 0xx0 xxxx x110 100xy   ;internal I/O write
equ mem     = 0xx0 xxxx 1110 01xxy   ;memory access
equ memforcp= 0xx0 xxxx 1101 01xxy   ;memory access for cp
equ memrd   = 0xx0 xxxx 1110 011xy   ;memory read
equ memrdcp = 0xx0 xxxx 1101 011xy   ;memory read for cp
equ memwr   = 0xx0 xxxx 1110 010xy   ;memory write
equ memwrcp = 0xx0 xxxx 1101 010xy   ;memory write for cp
equ mon     = 0xxx x0xx xxxx xxxxy   ;monitor cycle
equ nmladdr = 0xxx 0xxx xxxx xxxxy   ;normal address mode
equ read    = 0xx0 xxxx x1xx xx1xy   ;read cycle
equ refresh = 0xx0 xxxx x100 111xy   ;refresh cycle
equ write   = 0xx0 xxxx x1xx xx0xy   ;write cycle
equ wrrom   = 0xxx xxx0 x1xx xx0xy   ;write to ROM
```

Specifying a Simple Trigger

The **tg** analyzer command is a simple way to specify a condition on which to trigger the analyzer. Suppose you wish to trace the states of the program after the read of a "B" (42 hex) command from the command input byte. Enter the following commands to set up the trace, run the program, issue the trace, and display the trace status. (Note that the analyzer is to search for a lower byte read of 42H because the address is even.)

```
M>tg addr=800 and data=0xx42
```

If you wish to trace the odd address and the data, enter the following command to set up the trace (Note that the data value should be entered like as **0xx42 or 42xx** when using the 70136 emulator.): tg addr=801 and data=42xx

```
M>t
emulation trace started

M>r 0:400
U>ts
--- Emulation Trace Status ---
New User trace running
Arm ignored
Trigger not in memory
Arm to trigger ?
States ? (512) ?..?
Sequence term 1
Occurrence left 1
```

The trace status shows that the trigger condition has not been found. You would not expect the trigger to be found because no commands have been entered. Modify the command input byte to "B"(42H) and display the trace status again.

```
U>m 800=42
U>ts
---Emulation Trace Status ---
New User trace complete
Arm ignored
Trigger in memory
Arm to trigger ?
States 512 (512) 0..511
Sequence term 2
Occurrence left 1
```

The trace status shows that the trigger has been found, and that 512 states have been stored in trace memory. Enter the following command to display the first 20 states of the trace.

```
U>tl -t 20
```

Line	addr,H	70136 mnemonic,H		count,R	seq
0	000800	xx42 memory read	N	---	+
1	00041a	INSTRUCTION--opcode unavailable		0.040 uS	.
2	00041e	413c prefetch	N	0.080 uS	.
3	00041c	INSTRUCTION--opcode unavailable		0.040 uS	.
4	000420	0674 prefetch	N	0.080 uS	.
5	00041e	CMP AL,#41		0.120 uS	.
6	000422	423c prefetch	N	0.120 uS	.
7	000420	BE/Z 000428		0.080 uS	.
8	000424	0a74 prefetch	N	0.040 uS	.
9	000422	CMP AL,#42		0.080 uS	.
10	000426	10eb prefetch	N	0.120 uS	.
11	000424	BE/Z 000430		0.080 uS	.
12	000428	12b9 prefetch	N	0.040 uS	.
13	000430	12b9 prefetch	N	0.120 uS	.
14	000432	be00 prefetch	N	0.120 uS	.
15	000434	0012 prefetch	N	0.120 uS	.
16	000430	MOV CW,#0012		0.080 uS	.
17	000436	06eb prefetch	N	0.040 uS	.
18	000433	MOV IX,#0012		0.080 uS	.
19	000438	12b9 prefetch	N	0.080 uS	.

Line 0 in the trace list above shows the state which triggered the analyzer. The trigger state is always on line 0.

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

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.

To list the next lines of the trace, enter the following command.

```
U>tl
```

```
  Line   addr,H  70136 mnemonic,H                        count,R  seq
-----   ------  ----------------------------------  ---------  ---
   20   000436  BR      SHORT 00043e                    0.040 uS   .
   21   00043a    be00  prefetch                   N   0.080 uS   .
   22   00043e    01bf  prefetch                   N   0.160 uS   .
   23   000440    f300  prefetch                   N   0.160 uS   .
   24   000442    32a4  prefetch                   N   0.120 uS   .
   25   00043e  MOV     IY,#0001                        0.040 uS   .
   26   000444    b9c0  prefetch                   N   0.080 uS   .
   27   000446    0021  prefetch                   N   0.120 uS   .
   28   000441  REP/E/Z MOVBKB                         0.080 uS   .
   29   000448    cf2b  prefetch                   N   0.040 uS   .
   30   000612    xx43  memory read               N   0.200 uS   .
   31   000801    43xx  memory write              N   0.160 uS   .
   32   000613    6fxx  memory read               N   0.200 uS   .
   33   000802    xx6f  memory write              N   0.200 uS   .
   34   000614    xx6d  memory read               N   0.160 uS   .
   35   000803    6dxx  memory write              N   0.200 uS   .
   36   000615    6dxx  memory read               N   0.200 uS   .
   37   000804    xx6d  memory write              N   0.200 uS   .
   38   000616    xx61  memory read               N   0.160 uS   .
   39   000805    61xx  memory write              N   0.200 uS   .
```

For a Complete Description

For a complete description of the HP 64700 Series analyzer, refer to the *HP 64700 Emulators Terminal Interface: Analyzer User's Guide*.

Copying Memory

The **cp** (copy memory) command gives you the ability to copy the contents of one range of memory to another. This is a handy feature to test things like the relocatability of programs, etc. To test if the sample program is relocatable within the same segment, enter the following command to copy the program to an unused, but mapped, area of emulation memory. After the program is copied, run it from its new start address to verify that the program is indeed relocatable.

```
U>cp 500=400..44d
U>r 0:500
U>
```

The prompt shows that the emulator is executing user code, so it looks as if the program is relocatable. You may want to issue a simple trace to verify that the program works while running from its new location.

```
U>tg any
U>t
Emulation trace started

U>tl
```

```
  Line   addr,H  70136 mnemonic,H                        count,R  seq
-----   ------  ----------------------------------  ---------  ---
    0   00051a    003c  prefetch                  N        ---   +
    1   000516  INSTRUCTION--opcode unavailable         0.080 uS   .
    2   00051c    f874  prefetch                  N   0.040 uS   .
    3   000800    xx00  memory read              N   0.200 uS   .
    4   00051a  CMP     AL,#00                        0.080 uS   .
    5   00051e    413c  prefetch                  N   0.040 uS   .
    6   00051c  BE/Z    000516                        0.080 uS   .
    7   000520    0674  prefetch                  N   0.040 uS   .
    8   000516    a026  prefetch                  N   0.200 uS   .
    9   000518    0000  prefetch                  N   0.120 uS   .
   10   00051a    003c  prefetch                  N   0.120 uS   .
   11   000516  MOV     AL,DS1:0000                   0.080 uS   .
   12   00051c    f874  prefetch                  N   0.040 uS   .
   13   000800    xx00  memory read              N   0.200 uS   .
   14   00051a  CMP     AL,#00                        0.080 uS   .
   15   00051e    413c  prefetch                  N   0.040 uS   .
   16   00051c  BE/Z    000516                        0.080 uS   .
   17   000520    0674  prefetch                  N   0.040 uS   .
   18   000516    a026  prefetch                  N   0.200 uS   .
   19   000518    0000  prefetch                  N   0.120 uS   .
```

Testing for Coverage

For each byte of emulation memory, there is an additional bit of emulation RAM used by the emulator to provide coverage testing. When the emulator is executing the target program and an access is made to a byte in emulation memory, the corresponding bit of coverage memory is set. With the **cov** command, you can see which bytes in a range of emulation memory have (or have not) been accessed.

For example, suppose you want to determine how extensive some test input is in exercising a program (in other words, how much of the program is covered by using the test input). You can run the program with the test input and then use the **cov** command to display which locations in the program range were accessed.

The examples which follow use the **cov** command to perform coverage testing on the sample program. Before performing coverage tests, reset all coverage bits to non-accessed by entering the following command.

```
U>cov -r
```

Run the program from the start address (00000:00400H) and use the **cov** command to display how much of the program is accessed before any commands are entered (refer to the "ADDRESS" and "ADDRESS_EXPRESSION" section in the "70136 Emulator Specific Command Syntax" appendix).

```
U>r 400
R>cov -a 400..44d@e
# coverage list  -  list of address ranges accessed
0000400@e..0000423@e

percentage of memory accessed:  % 46.2
```

Now enter the sample program commands "A", "B", and an invalid command ("C" will do); display the coverage bits for the address range of the sample program after each command. You can see that more of the sample program address range is covered after each command is entered.

```
                                U>m 800=41
                                U>cov -a 400..44d@e
# coverage list  -  list of address ranges accessed
0000400@e..0000425@e
0000428@e..0000433@e
000043e@e..000044d@e

percentage of memory accessed:  % 84.6
                                U>m 800=42
                                U>cov -a 400..44d@e
# coverage list  -  list of address ranges accessed
0000400@e..000043b@e
000043e@e..000044d@e

percentage of memory accessed:  % 97.4
                                U>m 800=43
                                U>cov -a 400..44d@e
# coverage list  -  list of address ranges accessed
0000400@e..000044d@e

percentage of memory accessed:  % 100.0
```

Resetting the Emulator

To reset the emulator, enter the following command.

```
U>rst
R>
```

The emulator is held in a reset state (suspended) until a **b** (break), **r** (run), or **s** (step) command is entered. A CMB execute signal will also cause the emulator to run if reset.

Note

When the emulator is held in a reset state,the emulator is set to normal address mode.

The **-m** option to the **rst** command specifies that the emulator begin executing in the monitor after reset instead of remaining in the suspended state.

```
R>rst -m
M>
```

3

Emulation Topics

Introduction

Many of the topics described in this chapter involve the commands which are unique to the 70136 emulator such as the **cf** command which allows you to specify emulator configuration.
A reference-type description of the 70136 emulator configuration items can be found in the "70136 Emulator Specific Command Syntax" appendix.

This chapter will:

- Describe how to run in real-time and how to break on an analyzer trigger. These topics are related to program execution in general.
- Describe how to locate the monitor, These topics are related to the monitor options.
- Describe how to do other things which do not fall into the categories mentioned above: how to specify a run from reset.

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.

Execution Topics

The descriptions in this section are of emulation tasks which involve program execution in general.

Restricting the Emulator to Real-Time Runs

By default, the emulator is not restricted to real-time runs. However, you may wish to restrict runs to real-time to prevent accidental breaks that might cause target system problems. Use the **cf** (configuration) command to enable the **rrt** configuration item.

```
R>cf rrt=en
```

When runs are restricted to real-time and the emulator is running user code, the system refuses all commands that cause a break except **rst** (reset), **r** (run), **s**(step), and **b** (break to monitor).

Because the emulator contains dual-port emulation memory, commands which access emulation memory are allowed while runs are restricted to real-time.

But, when the emulator should break to the monitor to read page registers (**"cf pgrd=en"**), the memory command which need to Logical to Extended Address Conversion is not allowed even if accessing emulation memory; see the "CONFIG_ITEMS" section of the "70136 Emulator Specific Command Syntax" appendix.

The following commands are not allowed when runs are restricted to real-time:

- **reg** (register display/modification).
- **m** (memory display/modification) commands that access target system memory.
- **io** (I/O display/modification).

The following command will disable the restriction to real-time runs and allow the system to accept commands normally.

```
R>cf rrt=dis
```

Setting Up to Break on an Analyzer Trigger

The analyzer may generate a break request to the emulation processor. To set up to break on an analyzer trigger, follow the steps below.

Specify the Signal Driven when Trigger is Found

Use the **tgout** (trigger output) command to specify which signal is driven when the analyzer triggers. Either the "trig1" or the "trig2" signal can be driven on the trigger.

```
R>tgout trig1
```

Enable the Break Condition

Enable the "trig1" break condition.

```
R>bc -e trig1
```

After you specify the trigger to drive "trig1" and enable the "trig1" break condition, set up the trace, issue the **t** (trace) command, and run the program.

Making Coordinated Measurements

Coordinated measurements are measurements made between multiple HP 64700 Series emulators which communicate via the Coordinated Measurement Bus (CMB). Coordinated measurements can also include other instruments which communicate via the BNC connector. A trigger signal from the CMB or BNC can break emulator execution into the monitor, or it can arm the analyzer. An analyzer can send a signal out on the CMB or BNC when it is triggered. The emulator can send an EXECUTE signal out on the CMB when you enter the **x** (execute) command.

Coordinated measurements can be used to start or stop multiple emulators, start multiple trace measurements, or to arm multiple analyzers.

As with the analyzer generated break, breaks to the monitor on CMB or BNC trigger signals are interpreted as a "request to break". The emulator looks at the state of the CMB READY (active high) line to determine if it should break. It does not interact with the EXECUTE (active low) or TRIGGER (active low) signals.

For information on how to make coordinated measurements, refer to the *HP 64700 Emulators Terminal Interface: Coordinated Measurement Bus User's Guide* manual.

Monitor Option Topics

The monitor is a program which is executed by the emulation processor. It allows the emulation system controller to access target system resources. For example, when you enter a command that requires access to target system resources (display target memory, for example), the system controller writes a command code to a communications area and breaks the execution of the emulation processor into the monitor. The monitor program then reads the command from the communications area and executes the processor instructions which access the target system. After the monitor has performed its task, execution returns to the target program.

It does not take up any processor address space and does not need to be linked to the target program. The monitor resides in dedicated background memory.

Note

Halt instructions will cause "processor halted" emulation status (the "h>" prompt is shown).

In this status, the 70136 emulator does not break to the monitor. If you want to continue emulation session, you should enter the "**rst**" command to reset the emulator first.

The 70236 and 70236A emulator can break to the monitor in "processor halted" emulation status. Refer to "CONFIG_ITEMS" section in the "70136 Emulator Specific Command Syntax" appendix.

Background Monitor

When the emulator is powered up or initialized, the background monitor is selected by default.

Locating the Background Monitor

The default emulator configuration locates the monitor at 0FF000 hex. You can relocate the monitor to any 4K byte boundary. Use the **cf** (configuration) command with the **loc** configuration item to specify the location of the monitor.

```
R>cf loc=20000
```

Any valid physical address may be specified when relocating the monitor; however, the monitor address range will be placed on the 4K byte boundary at or below the address specified (for example, the monitor is placed at 20000 hex after the command shown above).

Foreground monitor

The default emulator configuration selects the background monitor. You can change the emulator configuration to select the foreground monitor. When you select the foreground monitor, processor address space is taken up. The foreground monitor takes up 4K bytes of memory. Use the **cf** command to select the foreground monitor.

```
R>cf mon=fg..2000
```

2000 defines an hexadecimal address (on a 4K byte boundary) where the monitor will be located. (Note: this will not load the monitor, it only specifies its location.) A memory mapper term is automatically created when you execute the **cf mon=fg** command to reserve 4K bytes of memory space for the monitor. The memory map is reset any time **cf mon=bg** is entered. It is only reset when the **cf mon=bg** command is entered if the emulator is not already configured to use the background monitor.

Note

The foreground monitor provided with the 70136 emulator should not be located at a base address 0 or 0ff000 hex; because the 70136 microprocessor's vector table is located.
Also, the foreground monitor can not be located at the base address over 100000 hex.

Note

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

Other Topics

This section describes how other emulation tasks, which did not fit into the previous groupings, are performed.

Selecting Accept Or Ignore Target System Reset

The 70136 emulator can respond or ignore target system reset while running in user program or waiting for target system reset (refer to "**cf trst**" configuration setting in "70136 Emulator specific Command Syntax" appendix).

While running in background monitor, the 70136 emulator ignores target system reset completely independent on this setting.

You can ignore reset from target system completely by specifying "**cf trst=dis**". In this configuration emulator ignore any reset from target system. Specifying "**cf trst=en"**, 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

4

In-Circuit Emulation Topics

Introduction

Many of the topics described in this chapter involve the commands which relate to using the emulator in-circuit, that is, connected to a target system.

This chapter will:

- Describe the issues concerning the installation of the emulator probe into target systems.
- Show you how to install the emulator probe.
- Describe how to use software breakpoints with ROMed code, how to perform coverage testing on ROMed code, and how to test patches to ROMed code. These topics relate to the debugging of target system ROM.
- Describe some of restrictions and considerations.

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 Emualtion and Analysis* manual and the "Getting Started" chapter of this manual.

Installing the Emulator Probe into a Target System

The 70136 emulator probe has a 68-pin PLCC connector; the 70236 and 70236A emulator probe has a 132-pin PGA connector. The 70236 and 70236A emulator is shipped with a pin protector over the target system probe. This guard is designed to prevent impact damage to the pins and should be left in place while you are not using the emulator.

Caution

DAMAGE TO THE EMULATOR CIRCUITRY MAY RESULT IF THESE PRECAUTIONS ARE NOT OBSERVED. The following precautions should be taken 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, operators should take precautionary measures 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.

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.

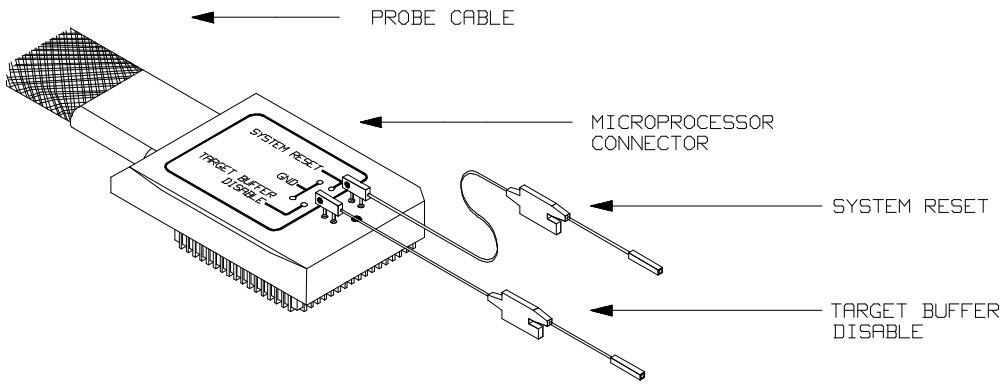
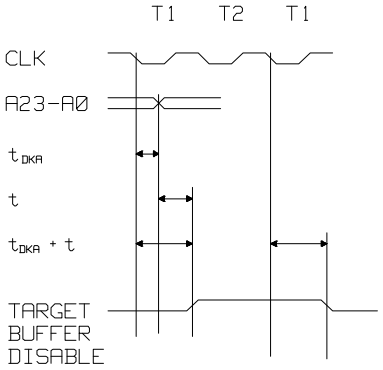


Figure 4-1. Auxiliary Output Lines (70136 Emulator)

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 70136/70236/70236A 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, proceeded 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.

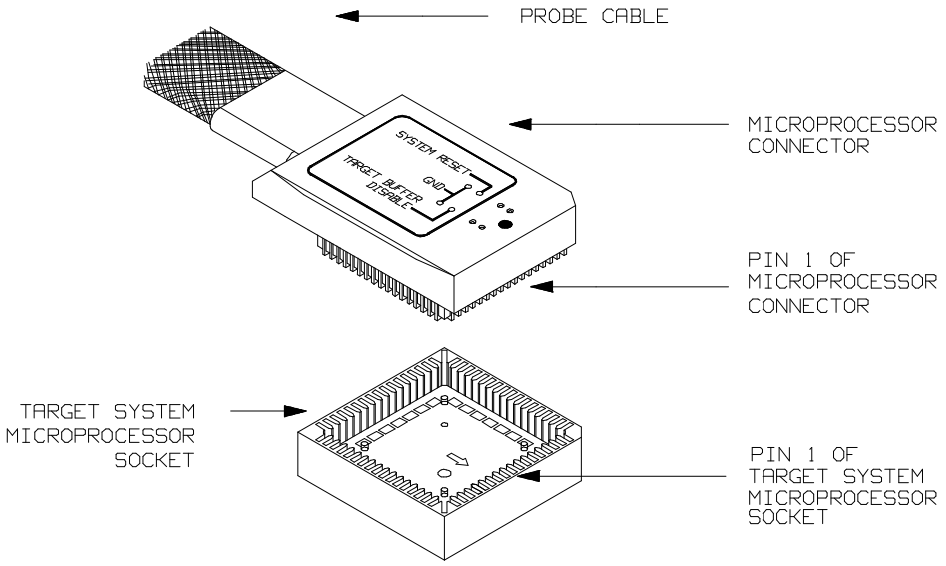


Figure 4-2. Installing into a 70136 PLCC type socket

Installing into a PGA Type Socket

The 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, proceeded 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 4-3), attached to the end of the probe cable, into the target system microprocessor socket.

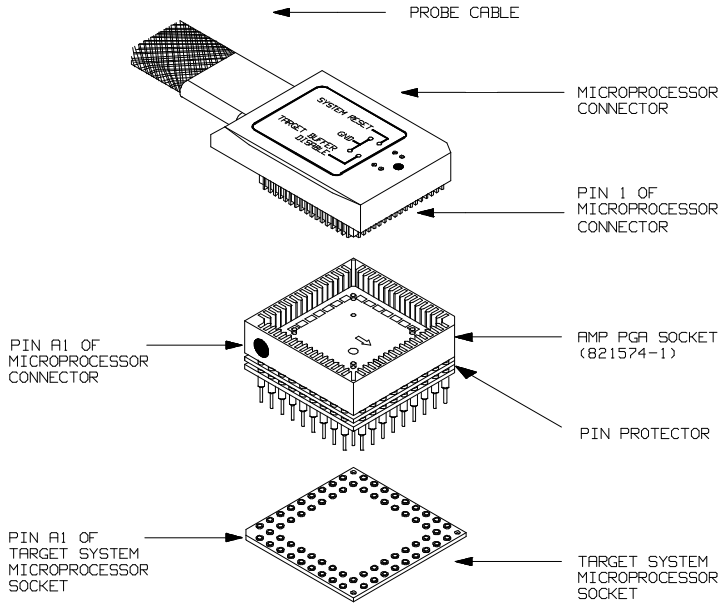


Figure 4-3. Installing into a 70136 PGA type socket

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, proceeded 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 4-4), attached to the end of the probe cable, into the target system microprocessor socket.

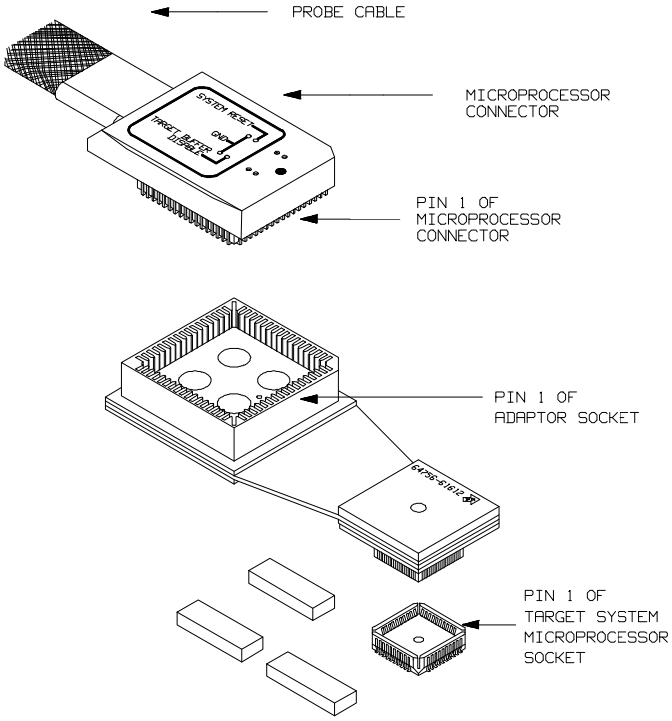


Figure 4-4. Installing into a 70136 QFP type socket

Installing into a 70236/70236A PGA Type Socket

To connect the microprocessor connector to the target system, proceeded 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 4-5).

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.

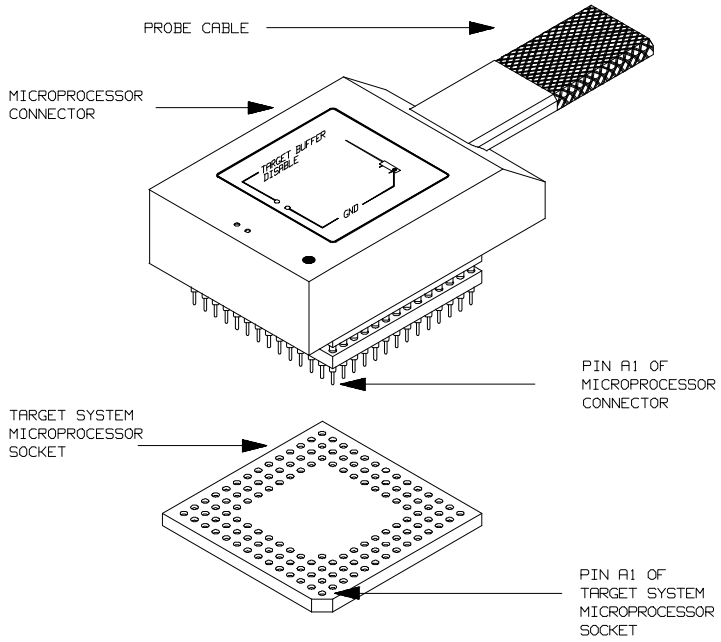


Figure 4-5. Installing into a 70236 PGA type socket

Execution Topics

The descriptions in this section are of emulation tasks which involve program execution in general.

Specifying the Emulator 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. Use the **cf** (configuration) command and the **clk** configuration item to specify that the emulator use a target system clock.

```
R>cf clk=ext
```

To reconfigure the emulator to use its internal clock, enter the following command.

```
R>cf clk=int
```

Emulator Probe Signal Topics

The descriptions in this section are of emulation tasks which involve emulator probe signals while in background or while accessing emulation memory.

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 lines while emulation memory is being accessed. Use the **cf** (configuration) command with the **rdy** configuration item to cause emulation memory accesses to honor target system ready signals.

```
R>cf rdy=lk
```

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).

To reconfigure so that emulation memory accesses do not honor target system ready signals, enter the following command.

```
R>cf rdy=unlk
```

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).

Target ROM Debug Topics

The descriptions in this section are of emulation tasks which involve debugging target ROM. The tasks described below are made possible by the **cim** (copy target system memory image) command.
The **cim** command allows you to read the contents of target memory into the corresponding emulation memory locations. Moving target ROM contents into emulation memory is the key which allows you to perform the tasks described below.
For example, if target ROM exists at locations 400H through 0A38H, you can copy target ROM into emulation memory with the following commands.

```
R>map 400..0bff erom
R>cim 400..0a38@e
```

Note that the target ROM location must be defined as the extended address (refer to the "ADDRESS" section the "70136 Emulator Specific Command Syntax" appendix).

Using Software Breakpoints with ROMed Code

You cannot define software breakpoints in target ROM memory. However, you can copy target ROM into emulation memory which does allow you to use software breakpoints.

Once target ROM is copied into emulation memory, software breakpoints may be used normally at addresses in these emulation memory locations.

```
R>bc -e bp
R>bp 440
```

Coverage Testing ROMed Code

Coverage testing (as described in the "Getting Started" chapter) can only be performed on emulation memory. However, if you wish to perform coverage tests on code in target system ROM, you can copy target ROM into emulation memory and perform the coverage tests on your ROMed code.

Once target ROM is copied into emulation memory, coverage testing may be done normally at addresses in these emulation memory locations (refer to the "ADDRESS" section the "70136 Emulator Specific Command Syntax" appendix).

```
U>cov -a 400..0a38@e
```

Modifying ROMed Code

Suppose that, while debugging your target system, you begin to suspect a bug in some target ROM code. You might want to fix or "patch" this code before programming new ROMs. This can also be done by copying target system ROM into emulation memory with the **cim** (copy target memory image) command. Once the contents of target ROM are copied into emulation memory, you can modify emulation memory to "patch" your suspected code.

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.
R/$\overline{W}$,M/$\overline{IO}$ BUSST0	Setting the "cf cyc=dis", always high impedance except accessing target. When accessing target by background monitor, same as foreground. Setting the "cf cyc=en", always high level except accessing target. When accessing target by background monitor, same as foreground.
BUSST1	Setting the "cf cyc=dis", always high impedance except accessing target. When accessing target by background monitor, same as foreground. Setting the "cf cyc=en", always low level except accessing target. When accessing target by background monitor, same as foreground.
$\overline{UBE}$	Setting the "cf cyc=dis", always high impedance except accessing target. When accessing target by background monitor, same as foreground. Setting the "cf cyc=en", 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.
R/$\overline{W}$,M/$\overline{IO}$, IORD,IOWR, MWR	Setting the "cf cyc=dis", always high impedance except accessing target. When accessing target by background monitor, same as foreground.
	Setting the "cf cyc=en", always high level except accessing target. When accessing target by background monitor, same as foreground.
$\overline{MRD}$	Setting the "cf cyc=dis", always high impedance except accessing target. When accessing target by background monitor, same as foreground.
	Setting the "cf cyc=en" same as foreground except for emulation memory write. When accessing emulation memory write, low.
BUSST2-0, UBE,BCYST, DSTB,BUFEN	Setting the "cf cyc=dis", always high impedance except accessing target. When accessing target by background monitor, same as foreground.
	Setting the "cf cyc=en", Same as foreground.
Other	Same as foreground

Electrical Characteristics (70136)

The AC characteristics of the 70136 emulator are listed in the following table

Table 4-1 70136 AC Electrical Specifications

Characteristic	Symbol	uPD70136 16MHz		HP 64756F			Unit
				Worst Case		Typical	
		Min	Max	Min	Max		
CLK Cycle Time	t_{CYK}	62.5	500	62.5	500		ns
CLK High Level Width	t_{KKH}	25		25			ns
CLK Low Level Width	t_{KKL}	25		25			ns
CLK Rise Time	t_{KR}		5		5		ns
CLK Fall Time	t_{KF}		5		5		ns
RESET Release Delay Time (Vdd)	t_{DURST}	1000		1000			ns
RESET Setup Time (CLK)	t_{SRSTK}	10		25			ns
RESET Hold Time (CLK)	t_{HKRST}	15		13			ns
RESET High Level Width	t_{WRSTH}	375		375			ns
CLK Low To $\overline{\text{BCYST}}$ Delay Time	t_{DKBC}	5	40	8	52		ns
$\overline{\text{BCYST}}$ High Level Width(*1)	t_{BCBCH}	52.5		54.5			ns
$\overline{\text{BCYST}}$ Low Level Width	t_{BCBCL}	52.5		54.5			ns
CLK Low To Address Delay Time	t_{DKA}	5	40	8	54.1		ns

Table 4-1 70136 AC Electrical Specification(Cont'd)

CLK Low To Status Delay Time	t_{DKST}	5	40	8	54.1		ns
CLK To Float Delay Time	t_{FK}	0	50	1	35.35		ns
CLK High To $\overline{\text{DSTB}}$ Low Delay Time	t_{DKDS}	5	40	8	52		ns
$\overline{\text{DSTB}}$ High Level Width	t_{DSDSH}	21.25		23.25			ns
$\overline{\text{DSTB}}$ Low Level Width (*1)	t_{DSDSL}	52.5		54.5			ns
CLK To $\overline{\text{DSTB}}$ High Delay Time	t_{DKDSH}	5	40	8	52		ns
Address/Status To $\overline{\text{DSTB}}$ Low Delay Time	t_{DADSL}	16.25		12.15			ns
$\overline{\text{DSTB}}$ High To Address/Status Hold Time	t_{HDSHA}	16.25		12.15			ns
$\overline{\text{DSTB}}$ High To Data Delay Time	t_{DDSHD}	16.25		12.15			ns
Address/Status To Data Delay Time	t_{DAD}	16.25		12.15			ns
CLK High To Data Delay Time	t_{DKD}	5	40	8	49.1		ns
Data Setup Time (CLK)	t_{SDK}	7		17.1			ns
Data Hold Time (CLK)	t_{HKD}	10		18.1			ns
Data Hold Time ($\overline{\text{DSTB}}$)	t_{HDSD}	0		2.2			ns
Data Hold Time (Address/Status)	t_{HASD}	0		2.2			ns
READY Setup Time (CLKOUT)	t_{SRYK}	7		21.2			ns
READY Hold Time (CLKOUT)	t_{HKRY}	15		23.1			ns
$\overline{\text{BS8}}$/BS16 Setup Time (CLKOUT)	t_{SBSK}	7		21.2			ns
$\overline{\text{BS8}}$/BS16 Hold Time (CLKOUT)	t_{HKBS}	15		23.1			ns

Table 4-1 70136 AC Electrical Specification(Cont'd)

HLDRQ Setup Time (CLK)	t_{SHQK}	7		21.2			ns
HLDRQ Hold Time (CLK)	t_{HKHQ}	15		23.1			ns
$\overline{\text{NMI}}$,INTPn(n=0-7),$\overline{\text{CPBUSY}}$ Setup Time (CLKOUT)	t_{SIK}	10		19.1			ns
$\overline{\text{NMI}}$,INTPn(n=0-7),$\overline{\text{CPBUSY}}$ Hold Time (CLKOUT)	t_{HKI}	10		13.1			ns

*1 No wait State

Electrical Characteristics (70236)

The AC characteristics of the 70236 emulator are listed in the following table

Table 4-2 70236 AC Electrical Specifications

Characteristic	Symbol	uPD70236 16MHz		HP 64757F			Unit
				Worst Case		Typical	
		Min	Max	Min	Max		
External Clock Cycle Times	t_{CYX}	31.25	250	31.25	250		ns
External Clock High Level Width	t_{KKH}	13		13			ns
External Clock Low Level Width	t_{KKL}	13		13			ns
External Clock Rise Time	t_{XKR}		5		5		ns
External Clock Fall Time	t_{XKF}		5		5		ns
CLKOUT Cycle Time	t_{CYK}	62.5	500	62.5	500		ns
CLKOUT High Level Width	t_{KKH}	24.25		26.25			ns
CLKOUT Low Level Width	t_{KKL}	24.25		26.25			ns
CLKOUT Rise Time	t_{KR}		7		5		ns
CLKOUT Fall Time	t_{KF}		7		5		ns
CLKOUT Delay Time (External Clock)	t_{DXK}	10	35	15	65		ns
PCLKOUT Cycle Time	t_{PCYK}	125	1000	125	1000		ns
PCLKOUT High Level Width	t_{PKH}	55.5		57.5			ns

Table 4-2 70236 AC Electrical Specification(Cont'd)

PCLKOUT Low Level Width	t_{PKL}	55.5		57.5			ns
PCLKOUT Rise Time	t_{PKR}		7		5		ns
PCLKOUT Fall Time	t_{PKF}		7		5		ns
Input Rise Time (*1)	t_{IR}		12		12		ns
Input Fall Time (*1)	t_{IF}		10		10		ns
Output Rise Time (*1)	t_{OR}		12		5		ns
Output Fall Time (*1)	t_{OF}		10		5		ns
$\overline{\text{RESET}}$ Setup Time	t_{SRSTK}	30			58.6		ns
$\overline{\text{RESET}}$ Hold Time	t_{HKRST}	15			12.2		ns
RESETOUT Delay Time (CLKOUT)	t_{DKRO}	0	40	-2.8	42.8		ns
$\overline{\text{RESET}}$ Low Level width	t_{WRSTL}	375		375			ns
$\overline{\text{BCYST}}$ Low Level width	t_{BCBCL}	52.5		57.5			ns
$\overline{\text{BCYST}}$ High Level width(*2)	t_{BCBCH}	52.5		52.5			ns
CLKOUT Low To $\overline{\text{BCYST}}$ Delay Time	t_{DKBC}	5	40	2.2	42.8		ns
CLKOUT Low To Address Delay Time	t_{DKA}	5	40	2.2	42.8		ns
CLKOUT Low To Status Delay Time	t_{DKST}	5	40	2.2	42.8		ns
CLKOUT To Float Delay Time	t_{FK}	0	30	-3.1	33.1		ns
CLKOUT High To $\overline{\text{DSTB}}$ Low Delay Time	t_{DKDS}	5	40	2.2	52.8		ns
$\overline{\text{DSTB}}$ High Level Width	t_{DSDSH}	21.25		21.25			ns

Table 4-2 70236 AC Electrical Specification(Cont'd)

$\overline{\text{DSTB}}$ Low Level Width (*2)	t_{DSDSL}	52.5		52.5			ns
CLKOUT To $\overline{\text{DSTB}}$ High Delay Time	t_{DKDSH}	5	40	2.2	52.8		ns
Address/Status To $\overline{\text{DSTB}}$ Low Delay Time	t_{DADSL}	16.25		13.45			ns
$\overline{\text{DSTB}}$ High To Address/Status Hold Time	t_{HDSHA}	16.25		13.45			ns
$\overline{\text{DSTB}}$ High To Oputput Set Time	t_{DLZ}	15.62		11.92			ns
$\overline{\text{DSTB}}$ Low To Output Floating Time	t_{DHZ}		0		3.1		ns
$\overline{\text{DSTB}}$ High To Data Delay Time	t_{DDSHD}	16.25		13.45			ns
Address/Status To Data Delay Time	t_{DAD}	16.25		13.45			ns
CLKOUT High To Data Delay Time	t_{DKD}	5	40	2.2	43.1		ns
Data Setup Time (CLKOUT)	t_{SDK}	10		30.9			ns
Data Hold Time (CLKOUT)	t_{HKD}	7		4			ns
Data Hold Time ($\overline{\text{DSTB}}$)	t_{HDSD}	0		-3			ns
Data Hold Time (Address/Status)	t_{HASD}	0		-3			ns
Data Hold Time (R/$\overline{\text{W}}$)	t_{HRWD}	0		-3			ns
READY Setup Time (CLKOUT)	t_{SRYK}	7		35.6			ns
READY Hold Time (CLKOUT)	t_{HKRY}	15		12			ns
$\overline{\text{BS8}}$/BS16 Setup Time (CLKOUT)	t_{SBSK}	7		35.6			ns
$\overline{\text{BS8}}$/BS16 Hold Time (CLKOUT)	t_{HKBS}	10		7			ns
HLDRQ Setup Time (CLKOUT)	t_{SHQK}	7		35.6			ns

Table 4-2 70236 AC Electrical Specification(Cont'd)

HLDRQ Hold Time (CLKOUT)	t_{HKHQ}	15		12			ns
CLKOUT High To HLDAK Delay Time	t_{DKHA}	5	40	2.2	42.8		ns
Output Float To HLDAK Delay Time	t_{DFHA}	16.25		13.45			ns
$\overline{NMI}$,INTPn(n=0-7),$\overline{CPBUSY}$ Setup Time (CLKOUT)	t_{SIK}	10		29.6			ns
$\overline{NMI}$,INTPn(n=0-7),$\overline{CPBUSY}$ Hold Time (CLKOUT)	t_{HKI}	10		7			ns
CLKOUT To $\overline{IOWR}$ Delay Time	t_{DKIW}	0	40	-2.8	42.8		ns
CLKOUT To $\overline{IORD}$ Delay Time	t_{DKIR}	0	40	-2.8	42.8		ns
CLKOUT To $\overline{MRD}$ Delay Time	t_{DKMR}	0	40	-2.8	42.8		ns
CLKOUT To $\overline{MWR}$ Delay Time	t_{DKMW}	0	40	-2.8	42.8		ns
Address/Status To $\overline{MRD}$, $\overline{IORD}$ Low Delay Time	t_{DARL}	16.25		13.45			ns
Data Hold Time ($\overline{MRD}$, $\overline{IORD}$)	t_{HRD}	0		-2.8			ns
Address/Status To $\overline{MWR}$, $\overline{IOWR}$ Low Delay Time	t_{DAWL}	16.25		13.45			ns
$\overline{MWR}$, $\overline{IOWR}$ Low Level Width (*2)	t_{WWL}	52.5		52.5			ns
$\overline{MWR}$ High To Address/Status Hold Time	t_{HMWHA}	16.25		13.45			ns
TCTLn(n=0-2) Setup Time (CLKOUT)	t_{SGK}	50		69.6			ns
TCTLn(n=0-2) Hold Time (CLKOUT)	t_{HKG}	100		97.2			ns
TCTLn(n=0-2) High Level Width	t_{GGH}	50		50			ns

Table 4-2 70236 AC Electrical Specification(Cont'd)

TCTLn(n=0-2) Low Level Width	t_{GGL}	50		50			ns
TOUTn(n=0-2) Delay Time (CLKOUT)	t_{DKTO}		100		102.8		ns
TCLK Rise Time	t_{TKR}		15		15		ns
TCLK Fall Time	t_{TKF}		15		15		ns
TCLK High Level Width	t_{TKTKH}	30		30			ns
TCLK Low Level Width	t_{TKTKL}	45		45			ns
TCLK Cycle Time	t_{CYTK}	100		100			ns
TCTLn(n=0-2) Setup Time (TCLK)	t_{SGTK}	50		52.8			ns
TCTLn(n=0-2) Hold Time (TCLK)	t_{HTKG}	100		102.8			ns
TOUTn(n=0-2) Delay Time (TCLK)	t_{DTKTO}		100		119.6		ns
TOUTn(n=0-2) Delay Time (TCTL)	t_{DGTO}		100		119.6		ns
RxD Setup Time (SCU Interanl Clock)	t_{SRX}	1000		1019			ns
RxD Hold Time (SCU Interanl Clock)	t_{HRX}	1000		997			ns
TOUT1 High To TxD Delay Time	t_{DTX}		500		502.8		ns
CLKOUT To CONTROL1 DelayTime(*3)	t_{DKCT1}		40		42.8		ns
CLKOUT To CONTROL2 DelayTime(*4)	t_{DKCT2}		40		42.8		ns
CLKOUT Low To $\overline{MWR}$, $\overline{IOWR}$ High Delay Time	t_{DKRH}		40		42.8		ns
CLKOUT Low To $\overline{MWR}$, $\overline{IOWR}$ Low Delay Time	t_{DKRL}		40		42.8		ns

Table 4-2 70236 AC Electrical Specification(Cont'd)

CLKOUT High To $\overline{\text{DMAKn}}$(n=0-3) Delay Time	t_{DKHDA}		40		42.8		ns
$\overline{\text{IORD}}$, $\overline{\text{IOWR}}$ Low Delay Time (MDAAKn(n=0-3))	t_{DDARW}	16.25		13.45			ns
$\overline{\text{DMAAKn}}$(n=0-3) High Delay Time ($\overline{\text{IORD}}$)	t_{DRHDAW}	16.25		13.45			ns
$\overline{\text{IORD}}$ High Delay Time ($\overline{\text{MWR}}$) $\overline{\text{MRD}}$ High Delay Time ($\overline{\text{IOWR}}$)	t_{DWHRH}	5	40	2.2	42.8		ns
$\overline{\text{TC}}$ Delay Time (CLKOUT)	t_{DKTCL}		40		42.8		ns
$\overline{\text{TC}}$ OFFDelay Time (CLKOUT)(*5)	t_{DKTCF}		40		42.8		ns
$\overline{\text{TC}}$ Pullup Delay Time (CLKOUT)(*5)	t_{DKTCH}		40		42.8		ns
$\overline{\text{END}}$ Setup Time(CLKOUT)	t_{SEDK}	35		54.6			ns
$\overline{\text{END}}$ Low Level Width	t_{EDEDL}	100		100			ns
$\overline{\text{IORD}}$, $\overline{\text{MRD}}$ Low Level Width	t_{RR}	85		82.2			ns
$\overline{\text{IOWR}}$, $\overline{\text{MWR}}$ Low Level Width (Extended Write)	t_{WW1}	85		82.2			ns
$\overline{\text{IOWR}}$, $\overline{\text{MWR}}$ Low Level Width (Normal Write)	t_{WW2}	22.5		19.7			ns
$\overline{\text{TC}}$ Low Level Width	t_{TCTCL}	47.5		47.5			ns
$\overline{\text{DMARQn}}$(n=0-3) Setup Time (CLKOUT)	t_{SDQK}	20		39.6			ns
CLKOUT HighTo $\overline{\text{DMAAKn}}$(n=0-3) Delay Time	t_{DKLDA}	0	40	-2.8	42.8		ns

Table 4-2 70236 AC Electrical Specification(Cont'd)

$\overline{\text{INTP}}$n(n=0-7) Low Level Width	t_{IPIPL}	100		100			ns

*1 Except CLKOUT,PLKOUT
*2 No wait State
*3 $\overline{\text{MWR}}$,$\overline{\text{IOWR}}$ during DMA cycl
*4 $\overline{\text{BEFEN}}$,$\overline{\text{INTAK}}$,$\overline{\text{REFRQ}}$
*5 TC must be pulled up by 2.2K

Electrical Characteristics (70236A)

The AC characteristics of the 70236A emulator are listed in the following table

Table 4-3 70236A AC Electrical Specifications

Characteristic	Symbol	uPD70236A 16MHz		HP 64757G			Unit
				Worst Case		Typical (*1)	
		Min	Max	Min	Max		
External Clock Cycle Times	t_{CYX}	25	250	25	250		ns
External Clock High Level Width	t_{KKH}	11		11			ns
External Clock Low Level Width	t_{KKL}	11		11			ns
External Clock Rise Time	t_{XKR}		3		3		ns
External Clock Fall Time	t_{XKF}		3		3		ns
CLKOUT Cycle Time	t_{CYK}	50	500	50	500		ns
CLKOUT High Level Width	t_{KKH}	20		15			ns
CLKOUT Low Level Width	t_{KKL}	20		15			ns
CLKOUT Rise Time	t_{KR}		5		5		ns
CLKOUT Fall Time	t_{KF}		5		5		ns
CLKOUT Delay Time (External Clock)	t_{DXK}	10	35	15	65		ns
PCLKOUT Cycle Time	t_{PCYK}	100	1000	100	1000		ns
PCLKOUT High Level Width	t_{PKH}	45		45			ns

Table 4-3 70236A AC Electrical Specification(Cont'd)

PCLKOUT Low Level Width	t_{PKL}	45		45			ns
PCLKOUT Rise Time	t_{PKR}		5		5		ns
PCLKOUT Fall Time	t_{PKF}		5		5		ns
Input Rise Time (*2)	t_{IR}		10		10		ns
Input Fall Time (*2)	t_{IF}		10		10		ns
Output Rise Time (*2)	t_{OR}		10		5		ns
Output Fall Time (*2)	t_{OF}		10		5		ns
$\overline{\text{RESET}}$ Setup Time	t_{SRSTK}	25		45.6			ns
$\overline{\text{RESET}}$ Hold Time	t_{HKRST}	12		12.2			ns
RESETOUT Delay Time (CLKOUT)	t_{DKRO}	0	40	-2.8	42.8		ns
$\overline{\text{RESET}}$ Low Level width	t_{WRSTL}	300		300			ns
$\overline{\text{BCYST}}$ Low Level width	t_{BCBCL}	43		40		43	ns
$\overline{\text{BCYST}}$ High Level width(*3)	t_{BCBCH}	43		40		46	ns
CLKOUT Low To $\overline{\text{BCYST}}$ Delay Time	t_{DKBC}	5	25	2.2	27.8	18	ns
CLKOUT Low To Address Delay Time	t_{DKA}	5	25	2.2	27.8	12	ns
CLKOUT Low To Status Delay Time	t_{DKST}	5	25	2.2	27.8	13	ns
CLKOUT To Float Delay Time	t_{FK}	0	25	-3.1	28.1	14	ns
CLKOUT High To $\overline{\text{DSTB}}$ Low Delay Time	t_{DKDS}	5	25	2.2	34.8	16	ns
$\overline{\text{DSTB}}$ High Level Width	t_{DSDSH}	18		15		20	ns

Table 4-3 70236A AC Electrical Specification(Cont'd)

$\overline{\text{DSTB}}$ Low Level Width (*3)	t_{DSDSL}	43		40		42	ns
CLKOUT To $\overline{\text{DSTB}}$ High Delay Time	t_{DKDSH}	5	25	2.2	34.8	22	ns
Address/Status To $\overline{\text{DSTB}}$ Low Delay Time	t_{DADSL}	13		10.2		28	ns
$\overline{\text{DSTB}}$ High To Address/Status Hold Time	t_{HDSHA}	13		10.2		21	ns
$\overline{\text{DSTB}}$ High To Oputput Set Time	t_{DLZ}	12.5		10.8		12	ns
$\overline{\text{DSTB}}$ Low To Output Floating Time	t_{DHZ}		0		3.1	2	ns
$\overline{\text{DSTB}}$ High To Data Delay Time	t_{DDSHD}	13		10.2		29	ns
Address/Status To Data Delay Time	t_{DAD}	13		10.2		25	ns
CLKOUT High To Data Delay Time	t_{DKD}	5	25	2.2	28.1	11	ns
Data Setup Time (CLKOUT)	t_{SDK}	10		19.9		12	ns
Data Hold Time (CLKOUT)	t_{HKD}	7		4		2	ns
Data Hold Time ($\overline{\text{DSTB}}$)	t_{HDSD}	0		-3		-5	ns
Data Hold Time (Address/Status)	t_{HASD}	0		-3		-5	ns
Data Hold Time (R/$\overline{\text{W}}$)	t_{HRWD}	0		-3		-4	ns
READY Setup Time (CLKOUT)	t_{SRYK}	5		22.6		18	ns
READY Hold Time (CLKOUT)	t_{HKRY}	12		12		6	ns
$\overline{\text{BS8}}$/BS16 Setup Time (CLKOUT)	t_{SBSK}	5		22.6		19	ns
$\overline{\text{BS8}}$/BS16 Hold Time (CLKOUT)	t_{HKBS}	7		12		6	ns
HLDRQ Setup Time (CLKOUT)	t_{SHQK}	5		22.6		19	ns

Table 4-3 70236A AC Electrical Specification(Cont'd)

HLDRQ Hold Time (CLKOUT)	t_{HKHQ}	12		12		7	ns
CLKOUT High To HLDAK Delay Time	t_{DKHA}	5	25	2.2	27.8		ns
Output Float To HLDAK Delay Time	t_{DFHA}	10		7.2			ns
$\overline{\text{NMI}}$,INTPn(n=0-7),$\overline{\text{CPBUSY}}$ Setup Time (CLKOUT)	t_{SIK}	7		16.6			ns
$\overline{\text{NMI}}$,INTPn(n=0-7),$\overline{\text{CPBUSY}}$ Hold Time (CLKOUT)	t_{HKI}	7		7			ns
CLKOUT To $\overline{\text{IOWR}}$ Delay Time	t_{DKIW}	0	25	-2.8	27.8	18	ns
CLKOUT To $\overline{\text{IORD}}$ Delay Time	t_{DKIR}	0	25	-2.8	27.8	19	ns
CLKOUT To $\overline{\text{MRD}}$ Delay Time	t_{DKMR}	0	25	-2.8	27.8	19	ns
CLKOUT To $\overline{\text{MWR}}$ Delay Time	t_{DKMW}	0	25	-2.8	27.8	20	ns
Address/Status To $\overline{\text{MRD}}$, $\overline{\text{IORD}}$ Low Delay Time	t_{DARL}	13		10.2		26	ns
Data Hold Time ($\overline{\text{MRD}}$, $\overline{\text{IORD}}$)	t_{HRD}	0		-2.8		-4	ns
Address/Status To $\overline{\text{MWR}}$, $\overline{\text{IOWR}}$ Low Delay Time	t_{DAWL}	13		10.2		25	ns
$\overline{\text{MWR}}$, $\overline{\text{IOWR}}$ Low Level Width (*3)	t_{WWL}	43		40		45	ns
$\overline{\text{MWR}}$ High To Address/Status Hold Time	t_{HMWHA}	13		10.2			ns
TCTLn(n=0-2) Setup Time (CLKOUT)	t_{SGK}	50		69.6			ns
TCTLn(n=0-2) Hold Time (CLKOUT)	t_{HKG}	100		97.2			ns
TCTLn(n=0-2) High Level Width	t_{GGH}	50		50			ns

Table 4-3 AC Electrical Specification(Cont'd)

TCTLn(n=0-2) Low Level Width	t_{GGL}	50		50			ns
TOUTn(n=0-2) Delay Time (CLKOUT)	t_{DKTO}		100		102.8		ns
TCLK Rise Time	t_{TKR}		15		15		ns
TCLK Fall Time	t_{TKF}		15		15		ns
TCLK High Level Width	t_{TKTKH}	30		30			ns
TCLK Low Level Width	t_{TKTKL}	45		45			ns
TCLK Cycle Time	t_{CYTK}	100		100			ns
TCTLn(n=0-2) Setup Time (TCLK)	t_{SGTK}	50		52.8			ns
TCTLn(n=0-2) Hold Time (TCLK)	t_{HTKG}	100		102.8			ns
TOUTn(n=0-2) Delay Time (TCLK)	t_{DTKTO}		100		119.6		ns
TOUTn(n=0-2) Delay Time (TCTL)	t_{DGTO}		100		119.6		ns
RxD Setup Time (SCU Interanl Clock)	t_{SRX}	1000		1019			ns
RxD Hold Time (SCU Interanl Clock)	t_{HRX}	1000		997			ns
TOUT1 High To TxD Delay Time	t_{DTX}		500		502.8		ns
CLKOUT To CONTROL1 DelayTime(*4)	t_{DKCT1}		35		37.8		ns
CLKOUT To CONTROL2 DelayTime(*5)	t_{DKCT2}		35		37.8		ns
CLKOUT Low To $\overline{MWR}$, $\overline{IOWR}$ High Delay Time	t_{DKRH}		35		37.8		ns
CLKOUT Low To $\overline{MWR}$, $\overline{IOWR}$ Low Delay Time	t_{DKRL}		35		37.8		ns

Table 4-3 70236A AC Electrical Specification(Cont'd)

CLKOUT High To $\overline{\text{DMAKn}}$(n=0-3) Delay Time	t_{DKHDA}		35		37.8		ns
$\overline{\text{IORD}}$, $\overline{\text{IOWR}}$ Low Delay Time (MDAAKn(n=0-3))	t_{DDARW}	13		10.2			ns
$\overline{\text{DMAAKn}}$(n=0-3) High Delay Time ($\overline{\text{IORD}}$)	t_{DRHDAW}	13		10.2			ns
$\overline{\text{IORD}}$ High Delay Time ($\overline{\text{MWR}}$) $\overline{\text{MRD}}$ High Delay Time ($\overline{\text{IOWR}}$)	t_{DWHRH}	5	35	2.2	37.8		ns
$\overline{\text{TC}}$ Delay Time (CLKOUT)	t_{DKTCL}		35		37.8		ns
$\overline{\text{TC}}$ OFFDelay Time (CLKOUT)(*6)	t_{DKTCF}		35		37.8		ns
$\overline{\text{TC}}$ Pullup Delay Time (CLKOUT)(*6)	t_{DKTCH}		35		37.8		ns
$\overline{\text{END}}$ Setup Time(CLKOUT)	t_{SEDK}	35		54.6			ns
$\overline{\text{END}}$ Low Level Width	t_{EDEDL}	100		100			ns
$\overline{\text{IORD}}$, $\overline{\text{MRD}}$ Low Level Width	t_{RR}	65		62.2			ns
$\overline{\text{IOWR}}$, $\overline{\text{MWR}}$ Low Level Width (Extended Write)	t_{WW1}	65		62.2			ns
$\overline{\text{IOWR}}$, $\overline{\text{MWR}}$ Low Level Width (Normal Write)	t_{WW2}	15		12.2			ns
$\overline{\text{TC}}$ Low Level Width	t_{TCTCL}	38		38			ns
$\overline{\text{DMARQn}}$(n=0-3) Setup Time (CLKOUT)	t_{SDQK}	12		33.6			ns
CLKOUT HighTo $\overline{\text{DMAAKn}}$(n=0-3) Delay Time	t_{DKLDA}	0	25	-2.8	27.8		ns

Table 4-3 70236A AC Electrical Specification(Cont'd)

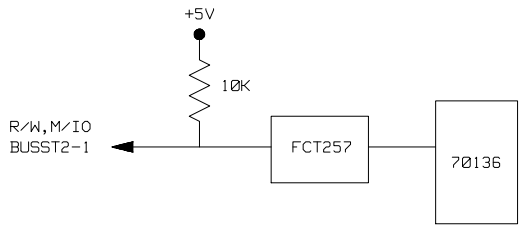
$\overline{\text{INTPn}}$(n=0-7) Low Level Width	t_{IPIPL}	100		100			ns
$\overline{\text{MRD}}$ High Level Width	t_{MRMRH}	18		15			ns
$\overline{\text{MRD}}$ High To Data Setup Time	t_{DMRHLZ}	12.5		10.8			ns
$\overline{\text{MRD}}$ High To Data Delay Time	t_{DMRHD}	13		10.2			ns
CLKOUT To Cascade Address Delay Time	t_{DKCA}	5	25	2.2	27.8		ns

*1 Typical outputs measured with 20pF load
*2 Except CLKOUT,PLKOUT
*3 No wait State
*4 $\overline{\text{MWR}}$,$\overline{\text{IOWR}}$ during DMA cycl
*5 $\overline{\text{BEFEN}}$,$\overline{\text{INTAK}}$,$\overline{\text{REFRQ}}$
*6 TC must be pulled up by 1.1K

Target System Interface (70136)

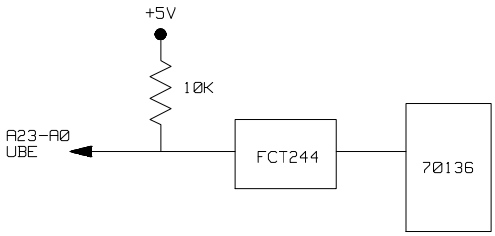
R/$\overline{W}$ M/$\overline{IO}$
BUSST2-1

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

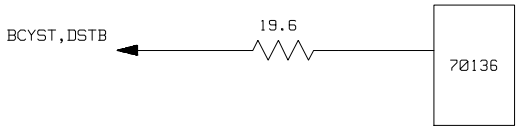


A23-A0
$\overline{UBE}$

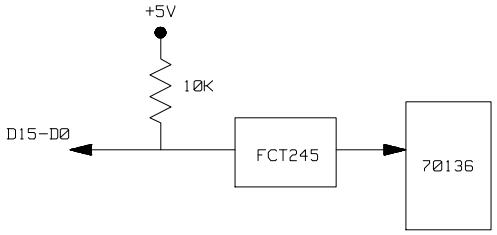
These singals are connected to 70136 through FCT244 and 10K ohm pull-up register.



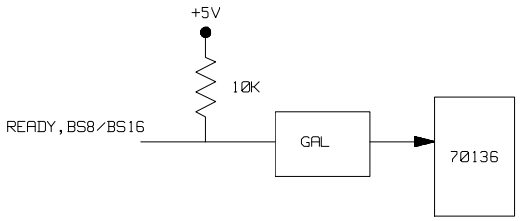
$\overline{\text{BCYST}}$ $\overline{\text{DSTB}}$ These singals are connected to 70136 through 19.6 ohm.



D15-D0 These singals are connected to 70136 through FCT245 and 10K ohm pull-up register.

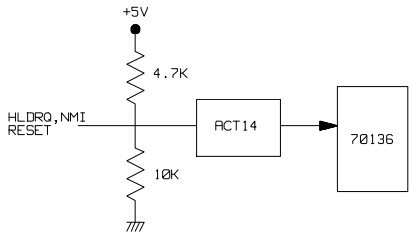


$\overline{\text{READY}}$
$\overline{\text{BS8}}$/BS16 These singals are connected to 70136 through GAL and 10K ohm pull-up register.



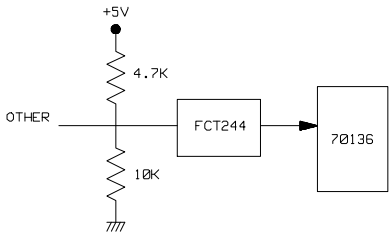
HLDRQ
NMI
RESET

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



OHTER

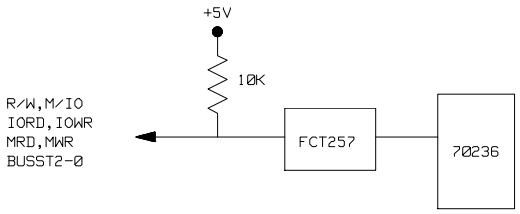
These singals are connected to 70136 through FCT244 and 4.7K ohm pull-up and 10K ohm pull-down registers.



Target System Interface (70236/70236A)

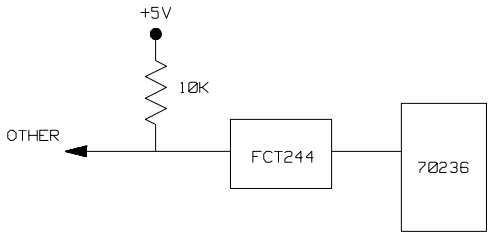
R/$\overline{W}$ M/$\overline{IO}$
IORD IOWR
MRD MWR
BUSST2-0

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



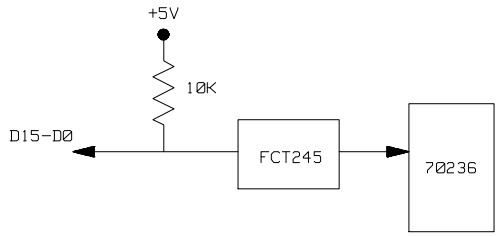
OTHER(INPUT)

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



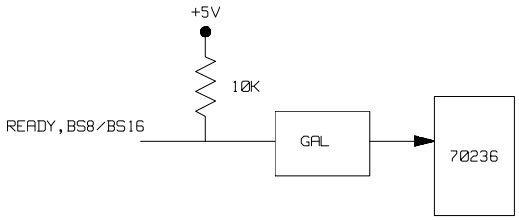
D15-D0

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



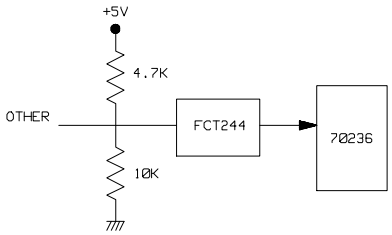
$\overline{\text{READY}}$
$\overline{\text{BS8}}$/BS16

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



OTHER(OUTPUT)

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



A

70136 Emulator Specific Command Syntax

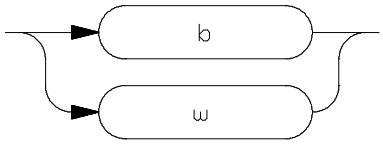
The following pages contain descriptions of command syntax specific to the 70136 emulator. The following syntax items are included (several items are part of other command syntax):

- <ACCESS_MODE>. May be specified in the **mo** (display and access mode), **m** (memory), and **io** (I/O port) commands. The access mode is used when the **m** or **io** commands modify target memory or I/O locations.
- <ADDRESS>. May be specified in emulation commands which allow addresses to be entered.
- <ADDRESS_EXPRESSION>. May be specified in emulation commands which allow address expressions to be entered.
- <CONFIG_ITEMS>. May be specified in the **cf** (emulator configuration) and **help cf** commands.
- <DISPLAY_MODE>. May be specified in the **mo** (display and access mode), **m** (memory), **io** (I/O port), and **ser** (search memory for data) commands. The display mode is used when memory locations are displayed or modified.
- <REG_NAME> and <REG_CLASS>. May be specified in the **reg** (register) command.

ACCESS_MODE

Summary Specify cycles used by monitor when accessing target system memory or I/O.

Syntax



Function The **<ACCESS_MODE>** specifies the type of microprocessor cycles that are used by the monitor program to access target memory or I/O locations. When a command requests the monitor to read or write to target system memory or I/O, the monitor program will look at the access mode setting to determine whether byte or word instructions should be used.

Parameters

b — **Byte.** Selecting the byte access mode specifies that the emulator will access target memory using upper and lower byte cycles (one byte at a time).

w — **Word.** Selecting the word access mode specifies that the emulator will access target memory using word cycles (one word at a time) at an even address.
When the emulator read or write odd number of byte data, the emulator will read or write the last byte data using byte cycle.
At an odd address, the emulator will access target memory using byte cycles.

Defaults The <**ACCESS_MODE**> is **b** at power up initialization. Access mode specifications are saved; that is, when a command changes the access mode, the new access mode becomes the current default.

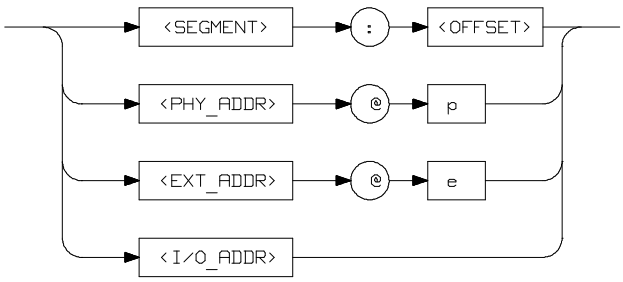
Related Commands **mo** (specify display and access modes)

ADDRESS

Address Syntax

Address specifications used in emulation commands.

Syntax



Function

The **<ADDRESS>** parameter 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 A-1) is converted to a segment:offset address and an extended address in memory commands (see table A-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

**** 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 (0-0FFFFF hex) with "@p" function code is a physical address in the 70136 address range. In run commands (see table A-1), the emulation system converts this physical address to a segment:offset address as specified by the **rad** (run address default) configuration item (see the **<CONFIG_ITEM>** description).

<EXT_ADDR> This expression (0-0FFFFFF hex) with "@e" function code is an extended address in the 70136 address range. In memory commands (see table A-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 A-1).

Defaults

If no number base is specified, values entered are interpreted as hexadecimal numbers.

Related Commands

<CONFIG_ITEMS> (70136 specific items specified with the **cf** command)

ADDRESS_ EXPRESSION

Summary Specify address expression used in emulation commands.

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

Table A-1. Address Expression Syntax

Command group	Terminal command	<EXT_ADDR> (@e)	<PHY_ADDR> (@p)	:<OFFSET>	No function code
Memory commands	cp,dump m,ser	OK	OK(*1)	OK(*1)	same as <PHY_ADDR>
	cim	OK	ERROR	ERROR	not accepted
	cov	OK	ERROR	ERROR	not accepted
Run commands (*2)	r,rx,s (rad=maxseg)	ERROR	OK	OK	same as <PHY_ADDR>
	r,rx,s (rad=minseg)	ERROR	OK	OK	same as <PYH_ADDR>
	r,rx,s (rad=curseg)	ERROR	ERROR	OK	<OFFSET> (0-0FFFFH)
I/O command	io	ERROR	ERROR	ERROR	OK (0-0FFFFH)
Map command	map	OK	ERROR	ERROR	same as <EXT_ADDR>
Breakpoints command	bp	OK	OK(*1)	OK(*1)	same as <PHY_ADDR>
Symbols command	sym	OK	OK	OK	same as <PYH_ADDR>

*1 : Emulator breaks to the monitor on accesses to emulation memory
(refer to "cf pgrd" command in CONFIG_ITEMS section of this appendix.)

*2 : Refer to the "cf rad" command in CONFIG_ITEMS section of this appendix.

Memory Commands

The following commands are included in memory commands (refer to *HP 64700 Emulators Terminal Interface: User's Reference* manual).

cp (Copy memory blocks)
dump (Dump memory to a host file)
m (Display/modify memory locations)
ser (Search memory for values)

You can use the following address expression in memory commands (refer to **cf pgrd** command in this appendix).

commands: **cp,dump,m,ser**

	<EXT_ADDR>	<PHY_ADDR>	:<OFFSET>
cf pgrd=en	OK	OK*	OK*
cf pgrd=dis	OK	OK	OK

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

When you set **cf pgrd** equal to **en**, the emulator should break to the monitor to get the current value of page register to convert physical address to extended address using in emulation system.

When you set **cf pgrd** equal to **dis**, 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 use **"cf pgrd = dis"** configuration setting when you only use normal address mode in your program or the value of page registers is not changed after initalizing while executing your program.

Note

When program execution should take place in real-time (refer to **cf rrt** command in this appendix) and the emulator should break to the monitor to read page registers (**cf pgrd** = **en**), the commands showing above which need physical to extended address conversion are not allowed in running user program (the "U>" prompt is shown). If you entered, the following error message will be shown:

```
!ERROR 145! Can not get extended address
```

The following commands are also included in memory commands.

cim (Copy target system memory to emulation memory)
cov (Measure percentage of memory locations accessed)

You can use the following address expression in using above commands.

commands:**cim,cov**

<EXT_ADDR>	<PHY_ADDR>	:<OFFSET>
OK	ERROR	ERROR

You should use the extended address expression to use the above commands.

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

Command : **cim**

```
!ERROR   736! Memory not mapped as emulation:
```

Command : **cov**

```
!ERROR   143! Physical address can not be
used
```

Load/Dump Commands

When you download programs into memory using **load** command, the emulator will interpret an address in the absolute file owing to the following configuration setting (refer to the **cf lad** command in CONFIG_ITEMS section in this appendix).

```
commands;load
```

configuration	address mode
cf lad = ext	extended address
cf lad = phy	physical address

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

```
commands;dump
```

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

(Refer to the "ADDRESS" section in this appendix and *HP 64700 Emulators Terminal Interface: User's Reference* manual.)

Note

When you download the host file made by **dump** command before, you should set the same **cf lad** configuration that you enter the **dump** command.
Otherwise, the memory image is not the same as when you enter the **dump** command to make the host file.

Note

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

Run Commands

The following commands are included in run commands (refer to *HP 64700 Emulators Terminal Interface: User's Reference* manual).

r (Run the emulator from current PC or specified location)
rx (Specify starting address for emulator run upon CMB execution)
s (Step the emulation processor one or more instructions)

You can use the following address expression in run commands.

commands:**r,rx,s**

	<EXT_ADDR>	<PHY_ADDR>	:<OFFSET>	No function code
cf rad = maxseg	ERROR	OK	OK	SAME AS <PHY_ADDR>
cf rad = minseg	ERROR	OK	OK	SAME AS <PHY_ADDR>
cf rad = curseg	ERROR	ERROR	OK	<OFFSET>

Refer to the "Default Physical to Logical Run Address Conversion" section in this appendix.

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   144! Extended address can not be used.
```

Default Physical to Logical Run Address Conversion

The run and step commands allow you to enter addresses in either logical form (segment:offset, e.g., 0F000:0FFFF) or physical form (e.g., 0FFFFF@p; @p is the function code as physical address; refer to the "ADDRESS" section in this appendix).

When a physical address (with @p) is entered with either a run or step command, the emulator must convert it to a logical (segment:offset) address. By default, a physical run address is converted such that the low 16 bits of the address become the offset value. Use the **cf** (configuration) command with the **rad** (run address default conversion) configuration item to specify 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.

```
R>cf rad=maxseg
# logical_addr = (phys_addr >> 4):(phys_addr & 0xf)
```

To reconfigure so that the low 16 bits of the physical address become the offset value, enter the following command. The physical address is right-shifted 4 bits and ANDed with 0F000H to yield the segment value.

```
R>cf rad=minseg
# logical_addr = ((phys_addr >> 4) & 0xf000):(phys_addr  & 0xffff)
```

To configure that the value (0 through 0FFFF hex ; with no function code) entered with either a run or step command becomes the offset value, enter the following command. In this configuration, the current segment value is not changed.

```
R>cf rad=curseg
# logical_addr = (current segment):(entered value)
```

If you use logical addresses other than the three methods shown above, you must enter run and step addresses in **:<OFFSET>** form.

I/O Command

The following command is included in I/O command.

io (Display/modify I/O locations)

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

Note

You should not change the value of the 70136 internal I/O registers with using **io** command. You should use the **reg** command to change the value of internal I/O registers.

Map Command

The following command is included in map command.

map (Map emulation and target system memory)

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, and display the memory map (The extended address expression should be used in **map** command).

```
R>map 0..7ff@e erom 16
R>map 800..9ff@e eram 8
R>map other tram
R>map
# remaining number of terms   : 14
# remaining emulation memory  : 1e600h bytes
map  0000000@e..00007ff@e  erom 16   # term 1
map  0000800@e..00009ff@e  eram  8   # term 2
map  other tram
```

As you can see, from 0 hex through 7ff hex is mapped as emulation ROM with 16-bit data bus; from 800 hex through 9ff hex is mapped as emulation RAM with 8-bit data bus; 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 the **ebs** and **tbs** command descriptions in "CONFIG_ITEMS" section in this appendix.

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 (refer to *HP 64700 Emulators Terminal Interface: User's Reference* manual).

bp (Insert or modify software breakpoints)

You can use the following address expression in breakpoints command (refer to **cf pgrd** command in this appendix).

```
commands:bp
```

	<EXT_ADDR>	<PHY_ADDR>	:<OFFSET>
cf pgrd=en	OK	OK*	OK*
cf pgrd=dis	OK	OK	OK

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

When you set **cf pgrd** equal to **en**, the emulator should break to the monitor to get the current value of page register to convert physical address to extended address using in emulation system.

When you set **cf pgrd** equal to **dis**, 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.

Symbols Command

The following command is included in symbols command.

sym (Manage the emulator symbol table)

You can use following address expression in **sym** command. If you define symbols with the following address expression, the symbol has the following address information.

Address expression (in sym command)	address information (in the symbol)
<EXT_ADDR> ("@e")	extended address
<PHY_ADDR> ("@p")	physical address
:<OFFSET>	:<OFFSET>
No function code	physical address

(Refer to the "ADDRESS" section in this appendix and *HP 64700 Emulators Terminal Interface: User's Reference* manual.)

Caution

The emulation system uses 24-bit extended address information to trace the execution of program (refer to the **tlb** command in "Getting Started" chapter).

When you specify trace condition in analyzer commands with using the symbol which has physical or :<offset> address information, the trace condition may be ignored in executed in extended address mode.

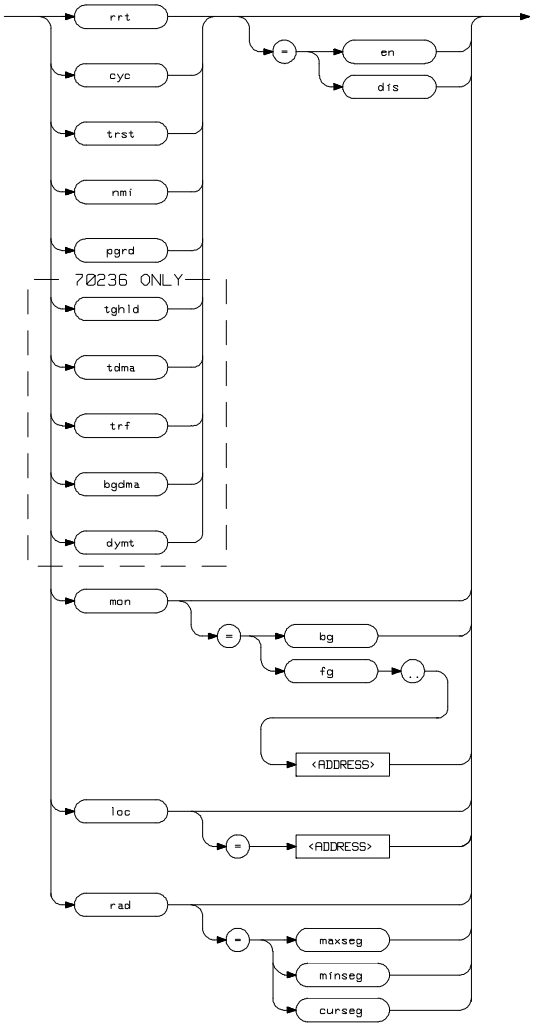
Related Commands

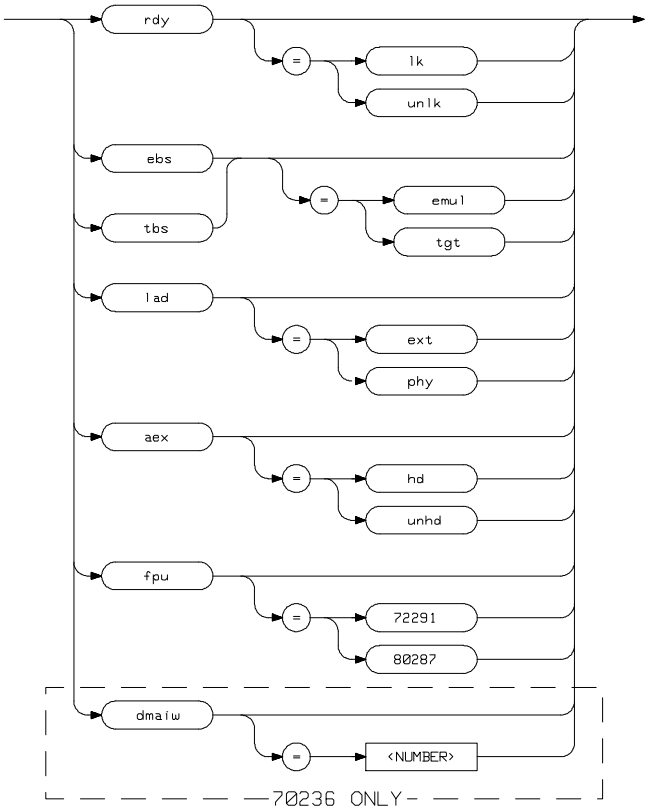
<CONFIG_ITEMS> (70136 specific items specified with the **cf** command)

CONFIG_ITEMS

Summary 70136 emulator configuration items.

Syntax





Function The **<CONFIG_ITEMS>** are the 70136 specific configuration items which can be displayed/modified using the **cf** (emulator configuration) command. If the "=" portion of the syntax is not used, the current value of the configuration item is displayed.

Parameters

clk Clock Source. This configuration item allows you to specify whether the emulator clock source is to be internal (**int**, provided by the emulator) or external (**ext**, provided by the target system).

The internal clock speed of 70216, 70236 and 70236A emulator is 16 MHz (system clock).

The 70136 emulator will operate at external clock speed from 2-16 MHz (entered clock).
The 70236 emulator will operate at external clock speed from 4-32 MHz (entered clock).
The 70236A emulator will operate at external clock speed from 4-40 MHz (entered clock).

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.

---

rrt Restrict to Real-Time Runs. This configuration item allows you to specify whether program execution should take place in real-time or whether commands should be allowed to cause breaks to the monitor during program execution.

To restrict execution to real-time, set **rrt** equal to **en**.

To allow breaks to the monitor during program execution, set **rrt** equal to **dis**. When runs are restricted to real-time, commands which access target system resources (display registers, step, or display/modify target system memory or I/O) are not allowed.

mon Monitor Options. This configuration item is used to select the type of monitor to be used by the emulator.

If **bg** (background monitor) is selected, all monitor functions are performed in background. If **fg** (foreground monitor) is selected, all monitor functions are performed in foreground. (Breaks to the monitor still put the emulator into the background mode, but the monitor program returns to foreground before performing any functions.)

Note

You should not use the physical or segment:offset address expression to locate the foreground monitor.
Refer to the "Using the Optional Foreground Monitor" appendix in this manual

---

cyc
(70136 Emulator)

Visible/Hidden Background Cycles. This configuration allows you to select whether or not the 70136 emulator will drive the bus status lines (M/IO, BUSST1, BUSST0, R/W) on all background monitor cycles to the target system.

Note

All address bus (A23 to A0), BCYST, and DSTB are always driven to the target system on all background monitor cycles independent on this configuration item.
All data bus (D15 to D0) are never driven to the target system on all background monitor cycles.

Setting **cyc** equal to **en** specifies that the emulator will drive the bus status lines to the target system. All cycles appear to the target system as memory read cycles (M/IO = 1, BUSST1 = 0, BUSST0 = 1, R/W = 1) from the address range of the monitor. It is possible to place the monitor at different locations if read cycles from the current range cause an undesired interaction (see the **loc** configuration item).

Setting **cyc** equal to **dis** specifies that the bus status lines (M/IO , BUSST1, BUSST0, R/W) are not driven to the target system.

cyc (70236/70236A Emulator)

Visible/Hidden Background Cycles. This configuration allows you to select whether or not the 70236 emulator will drive the bus status lines (M/IO, R/W, BUSST2, BUSST1, BUSST0, UBE, BCYST, DSTB) on all background monitor cycles to the target system.

Note

All address bus (A23 to A0), AEX, BUSLOCK, REFRQ, and HLDAK are always driven to the target system on all background monitor cycles independent on this configuration item.
All data bus (D15 to D0) are never driven to the target system on all background monitor cycles.

Note

The emulator will drive all bus lines on all DMA and refresh cycles in the background monitor to the target system.

Setting **cyc** equal to **en** specifies that the emulator will drive the bus status lines to the target system. All cycles appear to the target system as read cycles for memory (M/IO = 1, R/W = 1) from the address range of the monitor. It is possible to place the monitor at different locations if read cycles from the current range cause an undesired interaction (see the **loc** configuration item).

Setting **cyc** equal to **dis** specifies that the bus status lines (M/IO, R/W, BUSST2, BUSST1, BUSST0, UBE, BCYST, DSTB) are not driven to the target system.

loc Monitor Location. This configuration item allows you to specify the location of the background monitor program. The monitor may be located on any 4K byte boundary. If the <ADDRESS> specified is not on a 4K boundary, the 4K boundary below the address is used. The location of background monitors may be important because background cycles of the 70136 emulator can be visible to the target system.

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.

---

rad Physical to Logical Run Address Conversion. This configuration item allows you to specify the default method in which the emulation system will convert physical addresses specified in run and step commands to logical addresses.

Setting **rad** equal to **maxseg** specifies that the low nibble of the physical address become the offset value; the high four nibbles become the segment value.

Setting **rad** equal to **minseg** specifies that the low four nibbles of the physical address become the offset value; the high nibble and three hex zeros will become the segment value.

Setting **rad** equal to **curseg** specifies that the value which is entered in a run or step command is placed in the 70136 PC register. The segment value which is in the 70136 PS register is not changed.

Refer to the "Default Physical to Logical Run Address Conversion" section in this appendix.

trst The 70136 emulator can respond or ignore target system reset while running in user program or waiting for target system reset.

While running in background monitor, the 70136 emulator ignores target system reset completely independent on this setting.

Specifying "**cf trst=en"**, 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.

You can ignore reset from target system completely by specifying "**cf trst=dis**". In this configuration emulator ignore any reset from target system.

Note

When you use the **r rst** (run from reset) command in-circuit to run form processor reset after the target reset input, you should use **"cf trst=en"** configuration setting.

nmi Enable/disable user NMI. This configuration item allows you to specify whether user NMI is accepted or ignored by the emulator. To accept user NMI, set **nmi** equal to **en**. To ignore user NMI, set **nmi** to **dis**. When **nmi** is set to **dis**, the emulator ignores user NMI input.

Note

You should not use step command when 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 in this configuration setting.

rdy Allow Target Ready Signals to Insert Wait States. This configuration item allows you to specify whether the emulator should honor target system ready signals on accesses to emulation memory. Setting **rdy** equal to **lk** specifies that target ready signals be honored on emulation memory accesses. Setting **rdy** equal to **unlk** specifies that target ready signals be ignored on emulation memory accesses.

ebs Setting **ebs** equal to **tgt** specifies that the bus size of emulation memory is defined from the BS8/BS16 input of the target system.

Setting **ebs** equal to **emul** specifies that the bus size of emulation memory is selected from the setting of the map configuration.
Refer to the **map** command description in "ADDRESS_EXPRESSION" section of this appendix.

tbs

Setting **tbs** equal to **tgt** specifies that the bus size of target memory is defined from the $\overline{BS8}$/BS16 input of the target system.

Setting **tbs** equal to **emul** specifies that the bus size of target memory is selected from the setting of the map configuration.

Refer to the **map** command description in "ADDRESS_EXPRESSION" section of this appendix.

pgrd

Allow to break to the monitor to read page registers in Physical to Extended Address Conversion. 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 physical address specified in the following commands to extended address.

cp (Copy memory blocks)
dump (Dump memory to a host file)
m (Display/modify memory locations)
ser (Search memory for values)
bp (Insert or modify software breakpoints)

Setting **pgrd** equal to **en** specifies that the emulator should break to the monitor to get the current value of page registers on accesses to emulation/target memory.

Setting **pgrd** equal to **dis** 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 **reg** command (refer to *HP 64700 Emulators Terminal Interface: User's Reference* manual).

Note

You may use **"cf pgrd = dis"** configuration setting not to break to the monitor to read page registers when you only use the normal address mode in your program or the value of page registers is not changed after initalizing while executing your program.

Note

When program execution should take place in real-time (refer to **cf rrt** command in this appendix) and the emulator should break to the monitor to read page registers (**pgrd** = **en**), the commands showing above which need physical to extended address conversion are not allowed. Refer to Table A-1 in this appendix.

lad Specify the Load Address of an absolute file. If you specify **ext** for this configuration item, emulator will interpret address in absolute file as extended address.

If you specify **phy** for this configuration item, emulator will interpret address in absolute file as physical address.

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

Setting **aex** equal to **hd** 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 while the background monitor cycles with this configuration.

Setting **aex** equal to **unhd** specifies that the emulator will drive the **AEX** signal with the level dependent on the address mode in the background monitor cycles. When you use the extended address in using the memory commands (see table A-1) in background monitor, the **AEX** signal will be driven to high level with this configuration.

fpu

Select the assembler mnemonics for FPU (Floating Point Unit) to display memory.

Setting **fpu** equal to **72291** specifies that mnemonics for NEC uPD72291 floating point processor will be used to display memory.

Setting **fpu** equal to **80287** specifies that mnemonics for Intel 80287 numeric processor extension will be used to display memory.

tghld

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

Setting **tghld** equal to **dis** specifies that the emulator ignores HLDRQ signal from target system completely in background.

Setting **tghld** equal to **en** specifies that the emulator accepts HLDRQ signal. When the HLDRQ is accepted, the emulator will respond as actual microprocessor.

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

Setting **tdma** equal to **en** specifies that the analyzer will trace the internal DMA cycles.

Setting **tdma** equal to **dis** specifies that the analyzer will not trace the internal DMA cycles.

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

Setting **trf** equal to **en** specifies that the analyzer will trace the 70236 refresh cycles.

Setting **trf** equal to **dis** specifies that the analyzer will not trace the refresh cycles.

dmaiw **(70236 /70236AEmulator only)** Wait states for internal DMA cycles. When you want to trace internal DMA cycles correctly with using the emulator, you must set the number of wait states 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). Refer to the **cf tdma** command in this section.

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

Setting **bgdma** equal to **en** specifies that the internal DMA is allowed while in background.

Setting **bgdma** equal to **dis** specifies that the internal DMA is not allowed while in background.

dmyt **(70236Emulator only)** Trace dummy cycles during HALT acknowledge. This question allows you to specify whether or not the analyzer trace the emulation processor's dummy cycles during HALT acknowledge.
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 occurred by the breaks during the emulation processor is halted.

Setting **dmyt** equal to **dis** specifies that the analyzer will not trace the dummy cycles during HALT acknowledge.

Setting **dmyt** equal to **en** specifies that the analyzer will trace the dummy cycles during HALT acknowledge.

Defaults

The default values of the 70136 emulator configuration items are listed below.

```
cf clk=int
cf rrt=dis
cf mon=bg
cf cyc=en
cf loc=0ff000
cf rad=minseg
cf trst=en
cf nmi=en
cf rdy=unlk
cf ebs=emul
cf tbs=tgt
cf pgrd=en
cf lad=ext
cf aex=unhd
cf fpu=72291
```

The default values of the 70236 emulator configuration items are also listed below.

```
cf clk=int
cf rrt=dis
cf mon=bg
cf cyc=en
cf loc=0ff000
cf rad=minseg
cf trst=en
cf nmi=en
cf rdy=unlk
cf ebs=emul
cf tbs=tgt
cf pgrd=en
cf lad=ext
cf aex=unhd
cf fpu=72291
cf tghld=dis
cf tdma=en
cf trf=en
cf dmaiw=7
cf bgdma=en
cf dmyt=dis
```

Related Commands

help

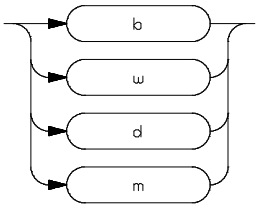
You can get an on line help information for particular configuration items by typing:

```
R>help cf <CONFIG_ITEM>
```

DISPLAY_MODE

Summary Specify the memory display format or the size of memory locations to be modified.

Syntax



Function The **<DISPLAY_MODE>** specifies the format of the memory display or the size of the memory which gets changed when memory is modified.

Parameters

b	**Byte**. Memory is displayed in a byte format, and when memory locations are modified, bytes are changed.
w	**Word**. Memory is displayed in a word format, and when memory locations are modified, words are changed.
d	**Double Word**. Memory is displayed in a double word format, and when memory locations are modified, double words are changed.
m	**Mnemonic**. Memory is displayed in mnemonic format; that is, the contents of memory locations are inverse-assembled into mnemonics and operands. When memory locations are modified, the last non-mnemonic display mode specification is used. You cannot specify this display mode in the **ser** (search memory for data) command.

Defaults

The <**DISPLAY_MODE**> is **b** at power up initialization. Display mode specifications are saved; that is, when a command changes the display mode, the new display mode becomes the current default.

Related Commands

mo (specify access and display modes)

m (memory display/modify)

io (I/O display/modify)

ser (search memory for data)

REGISTER NAMES and CLASSES (70136 Emulator)

The following register names and classes are used with the display/modify registers commands in 70136 emulator.

BASIC(*) class

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

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 Emulator)

The following register names and classes are used with the display/modify registers commands in 70236 emulator.

BASIC(*) class

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

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	Boud 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
sbcr	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, interruptis 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(Write only)
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)

Related Commands

reg (register display/modify)

Notes

B

Using the Optional Foreground Monitor

By using and modifying the optional Foreground Monitor, you can provide an emulation environment which is customized to the needs of a particular target system.

The monitor programs named **FM70136.S** and **FM70236.S** are for the HP 64853 Cross Assembler/Linker.

Note

Use the appropriate monitor; "FM70136.S" for the 70136 emulator and "FM70236.S" for the 70236 and 70236A emulator. "FM70136.S" foreground monitor program is used in this example. If your emulator is for the 70236 or 70236A, read this appendix by replacing "FM70136" with "FM70236".

Comparison of Foreground and Background Monitors

An emulation monitor is required to service certain requests for information about the target system and the emulation processor. For example, when you request a register display, the emulation processor is forced into the monitor. The monitor code has the processor dump its registers into certain emulation memory locations, which can then be read by the emulator system controller without further interference.

Background Monitors

A *background* monitor is an emulation monitor which overlays the processor's memory space with a separate memory region.

Usually, a background monitor will be easier to work with in starting a new design. The monitor is immediately available upon powerup, and you don't have to worry about linking in the monitor code or allocating space for the monitor to use the emulator. No assumptions are made about the target system environment; therefore, you can test and debug hardware before any target system code has been written. All of the processor's address space is available for target system use, since the monitor memory is overlaid on processor memory, rather than subtracted from processor memory. Processor resources such as interrupts are not fully taken by the background monitor.

However, all background monitors sacrifice some level of support for the target system. For example, when the emulation processor enters the monitor code to display registers, it will not respond to target system interrupt requests. This may pose serious problems for complex applications that rely on the microprocessor for real-time, non-intrusive support. Also, the background monitor code resides in emulator firmware and can't be modified to handle special conditions.

Foreground Monitors

A *foreground* monitor may be required for more complex debugging and integration applications. A foreground monitor is a block of code that runs in the same memory space as your program. You link this monitor with your code so that when control is passed to your program, the emulator can still service real-time events, such as interrupts or watchdog timers. For most multitasking, interrupt intensive applications, you will need to use a foreground monitor.

You can tailor the foreground monitor to meet your needs, such as servicing target system interrupts. However, the foreground monitor does use part of the processor's address space, which may cause problems in some target systems. You must also properly configure the emulator to use a foreground monitor (see the "Emulation topics" chapter and the examples in this appendix).

An Example Using the Foreground Monitor

In the following example, we will illustrate how to link a foreground monitor. By using the emulation analyzer, we will also show how the emulator switches from state to state using a foreground monitor.

For this example, We will locate the monitor at 1000 hex; the sample program will be located at 400 hex with its data at 600 hex and its common at 800 hex.

Modify EQU Statement

To use the monitor, you must modify the EQU statement near the top of the monitor listing to point to the segment start address where the monitor will be loaded. In this example, the monitor will be located at 1000 hex, so the modified EQU statement looks like this:

```
MONSEGMENT      EQU     0100H
```

Notice that the EQU statement is indented from the left margin; if it is not indented, the assembler will attempt to interpret the EQU as a label and will generate an error when processing the address portion of the statement. You can load the monitor at any base address on a 4k byte boundary.

Note

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

Also, the foreground monitor can not be located at the base address over 100000 hex.

Assemble and Link the Monitor

You can assemble and link the foreground monitor program with the following commands in using the HP 64853 Cross Assembler/Linker:

```
$ asm -o FM70136.S > FM70136.LIS <RETURN>
$ lnk <RETURN>
object files  FM70136.R  <RETURN>
library files  <RETURN>
Load addresses: PROG,DATA,COMN  0,0,0
<RETURN>
more files (y or n)  n  <RETURN>
absolute file name  FM70136.X  <RETURN>
```

If you haven't already assembled and linked the sample program, do that now. Refer to the "Getting Started" chapter for instructions on assembling and linking the sample program.

Initialize the Emulator

To initialize the emulator to a known state for this example, type:

```
M> init -p
```

```
                Copyright (c) Hewlett-Packard Co. 1987
All Rights Reserved.  Reproduction, adaptation, or translation without prior
written permission is prohibited, except as allowed under copyright laws.

  HP64700 Series Emulation System
    Version:   A.02.02 24Jan90

  HP64756 NEC 70136 emulator
  HP64740 Emulation Analyzer with External State/Timing Analyzer
```

Configure the Emulator

You need to tell the emulator that you will be using a foreground monitor and allocate the memory space for the monitor. This is all done with one configuration command. To locate the monitor on a 4k boundary starting at 1000 hex, type:

```
R> cf mon=fg..001000
```

To see the new memory mapper term allocated for the foreground monitor, type:

```
R> map
```

```
# remaining number of terms   : 15
# remaining emulation memory  : 1f000h bytes
map  0001000@e..0001fff@e  eram 16       # term 1
map  other tram
```

Notice that a 4k byte block from 1000 through 1fff hex was mapped.

Now, you need to map memory space for the sample program. Let's map the memory from 0 through 5ff hex to emulation ROM and map the memory from 600 through 9ff hex to emulation RAM in this example.

```
R> map 0..5ff erom
R> map 600..9ff eram
```

Load the Program Code

Now it's time to load the sample program and monitor. In the example shown, we're loading the program from a host with the emulator in Transparent Configuration. If you're using the standalone configuration with a data terminal, you will need to enter the data using the **m** command. (You can get the data from your assembly listings.) Load the program by typing:

```
R> load -hbs "transfer -tb FM70136.X"
```

##############

Load the Sample Program

Assuming the sample program has been assembled and linked as shown in Chapter 2, you can load the sample program by typing:

```
R> load -hbs "transfer -tb cmd_rds.X"
```

#####

Disable Tracing Refresh Cycle (70236/70236A Emulator only)

If you wish to disable the analyzer from tracing refresh cycles, you can use the **cf trf** command ; the refresh cycles are not detected by the analyzer. Type:

```
M> cf trf=dis
```

Set Analyzer Master Clock Qualifiers

We want to view the transitions made between the different emulator states: reset to break, break to run, run to break. Since the foreground monitor is actually entered via a few cycles in the emulator's built-in background monitor, we need to be able to view the background states. We can do this by modifying the emulation analyzer's master clock qualifier to include tracing of background code. To see the initial clock qualifier, type:

```
M> tck
```

```
tck -r L -u -s S
```

Modify this as follows:

```
M> tck -r L -ub -s S
```

Now, reset the processor so we can make the first measurement from a known state:

```
M> rst
```

Reset to Break

We want to see the monitor's transition from the reset state to running in the foreground monitor. Since the foreground monitor occupies the address range from 1000 through 1fff hex, we can simply trigger on any access to that range:

```
R> tg addr=1000..1fff
```

We also want to see the states leading up to the transition between reset and foreground monitor execution. We can position the trigger so that there are 20 states **before** the trigger as follows:

```
R> tp -b 20
```

Start the measurement:

```
R> t
```

```
Emulation trace started
```

Now, break the emulator into the monitor:

```
R> b
```

Display 20 disassembled states of the trace from the top of the trace:

```
M> tl -td 20
```

```
  Line   addr,H  70136 mnemonic,H                          count,R  seq
-----   ------  ------------------------------------  ---------  ---
   -3   0ffff4    ffff  prefetch                    N        ---   .
   -2   000008    0310  memory read                 M   0.160 uS   .
   -1   00000a    0100  memory read                 M   0.200 uS   .
    0   001310    c62e  prefetch                    M   0.320 uS   +
    1   001312    0e06  prefetch                    M   0.200 uS   .
    2   0000fe    f002  undefined                   M   0.160 uS   .
    3   0000fc    ffff  undefined                   M   0.200 uS   .
    4   0000fa    0000  undefined                   M   0.200 uS   .
    5   001314    0002  prefetch                    M   0.120 uS   .
    6   001316    a32e  prefetch                    M   0.120 uS   .
    7   001310  MOV     PS:020e,#00                     0.080 uS   .
    8   001318    00e6  prefetch                    M   0.040 uS   .
    9   00120e    xx00  memory write                M   0.200 uS   .
   10   001316  MOV     PS:00e6,AW                      0.040 uS   .
   11   00131a    892e  prefetch                    M   0.080 uS   .
   12   0010e6    0011  memory write                M   0.160 uS   .
   13   00131c    ec16  prefetch                    M   0.160 uS   .
   14   00131e    8b00  prefetch                    M   0.120 uS   .
   15   001320    25c4  prefetch                    M   0.120 uS   .
   16   00131a  MOV     PS:00ec,DW                      0.080 uS   .
```

At state -2, the processor began executing code, actually starting to read the data; it executed in the background monitor. To see the transition from background execution to foreground monitor program execution, type:

```
M> tl 140..160
```

```
  Line   addr,H  70136 mnemonic,H                          count,R  seq
-----   ------  ------------------------------------  ---------  ---
  140   00142c  NOP                                     0.080 uS   .
  141   00142d  NOP                                     0.160 uS   .
  142   001434    0200  prefetch                    M   0.120 uS   .
  143   00142e  illegal opcode, data = 0f cf            0.160 uS   .
  144   0000fa    0500  undefined                   M   0.240 uS   .
  145   0000fc    0100  undefined                   M   0.160 uS   .
  146   0000fe    f002  undefined                   M   0.200 uS   .
  147   001500    8c2e  prefetch                    N   0.200 uS   .
  148   001502    fa16  prefetch                    N   0.120 uS   .
  149   001504    2e00  prefetch                    N   0.120 uS   .
  150   001506    2689  prefetch                    N   0.120 uS   .
  151   001500  MOV     PS:00fa,SS                      0.080 uS   .
  152   001508    00ee  prefetch                    N   0.040 uS   .
  153   0010fa    0000  memory write                N   0.120 uS   .
  154   001505  MOV     PS:00ee,SP                      0.080 uS   .
  155   00150a    892e  prefetch                    N   0.080 uS   .
  156   0010ee    0100  memory write                N   0.160 uS   .
  157   00150c    e81e  prefetch                    N   0.120 uS   .
  158   00150e    2e00  prefetch                    N   0.120 uS   .
  159   001510    0e89  prefetch                    N   0.160 uS   .
  160   00150a  MOV     PS:00e8,BW                      0.040 uS   .
```

The foreground monitor start at states 147.

Monitor to User Program

We can look at the transition from the foreground monitor to running the user program by triggering the trace on a user program address. Type:

```
M> tg addr=400
```

We will leave the trigger position where it was for the last measurement (20 states are retained before the trigger position). Start the measurement:

```
M> t
```

```
Emulation trace started
```

Now, run the sample program:

```
M> r 400
```

Display trace states from -15 to +5 in disassembled form as follows:

```
U> tl -d -15..5
```

```
   Line   addr,H  70136 mnemonic,H                        count,R  seq
  -----   ------  ---------------------------------  ---------  ---
    -15   0010f8    0000  memory read                 N   0.160 uS   .
    -14   001970    fa16  prefetch                    N   0.120 uS   .
    -13   001972    2e00  prefetch                    N   0.160 uS   .
    -12   001974    268b  prefetch                    N   0.120 uS   .
    -11   00196e  MOV     SS,PS:00fa                      0.040 uS   .
    -10   001976    00ee  prefetch                    N   0.080 uS   .
     -9   0010fa    0100  memory read                 N   0.120 uS   .
     -8   001973  MOV     SP,PS:00ee                      0.080 uS   .
     -7   001978    00cf  prefetch                    N   0.040 uS   .
     -6   0010ee    0ff4  memory read                 N   0.200 uS   .
     -5   001978  RETI                                    0.080 uS   .
     -4   00197a    0000  prefetch                    N   0.040 uS   .
     -3   001ff4    0400  memory read                 N   0.200 uS   .
     -2   001ff6    0000  memory read                 N   0.160 uS   .
     -1   001ff8    f002  memory read                 N   0.200 uS   .
      0   000400    60b8  prefetch                    N   0.200 uS   +
      1   000402    8e00  prefetch                    N   0.120 uS   .
      2   000404    b8d8  prefetch                    N   0.120 uS   .
      3   000400  MOV     AW,#0060                        0.080 uS   .
      4   000406    0080  prefetch                    N   0.040 uS   .
      5   000403  MOV     DS0,AW                          0.080 uS   .
```

At state -5 in the trace listing, the processor executed the **RETI** instruction to transfer execution to the user program at state 0.

Note

As you can see in the trace list, user stack pointer is used when context is changed from foreground monitor to user program, or from user program to foreground monitor program. If you are configuring the emulator to background monitor, the user stack is not used.

User Program Run to Break

You can trace the execution from the user program run to the foreground monitor due to a break condition by setting as below.

```
U> tg stat=mon
```

Start the measurement:

```
U> t
```

```
Emulation trace started
```

Satisfy the trigger condition by breaking the emulator into the monitor:

```
U> b
```

Now, display states -10 through 10 of the trace list in disassembled format:

```
M> tl -10..10
```

```
   Line   addr,H  70136 mnemonic,H                          count,R  seq
 -----   ------  ------------------------------------  ---------  ---
   -10   000416  MOV     AL,DS1:0000                     0.080 uS   .
    -9   00041c    f874  prefetch                    N   0.040 uS   .
    -8   000800    xx00  memory read                N   0.200 uS   .
    -7   00041a  CMP     AL,#00                          0.040 uS   .
    -6   00041e    413c  prefetch                    N   0.080 uS   .
    -5   00041c  BE/Z    000416                          0.080 uS   .
    -4   000420    0674  prefetch                    N   0.040 uS   .
    -3   000416    a026  prefetch                    N   0.200 uS   .
    -2   000418    0000  prefetch                    N   0.120 uS   .
    -1   00041a    003c  prefetch                    N   0.120 uS   .
     0   000008    0310  memory read                 M   0.240 uS   +
     1   00000a    0100  memory read                 M   0.200 uS   .
     2   001310    c62e  prefetch                    M   0.320 uS   .
     3   001312    0e06  prefetch                    M   0.160 uS   .
     4   0008fe    f046  undefined                   M   0.200 uS   .
     5   0008fc    0000  undefined                   M   0.200 uS   .
     6   0008fa    0416  undefined                   M   0.160 uS   .
     7   001314    0002  prefetch                    M   0.120 uS   .
     8   001316    a32e  prefetch                    M   0.160 uS   .
     9   001310  MOV     PS:020e,#00                     0.040 uS   .
    10   001318    00e6  prefetch                    M   0.080 uS   .
```

At state 0 of the trace list, the processor entered the background state to make the transition. And actual foreground monitor program start at after several background monitor execution. To see the starting point of foreground monitor, type:

```
M> tl 140..160
```

```
  Line   addr,H  70136 mnemonic,H                        count,R  seq
-----   ------  ----------------------------------  ---------  ---
  140   00142b  NOP                                    0.120 uS   .
  141   001432    a02e  prefetch                  M   0.120 uS   .
  142   00142c  NOP                                    0.040 uS   .
  143   00142d  NOP                                    0.200 uS   .
  144   001434    0200  prefetch                  M   0.120 uS   .
  145   00142e  illegal opcode, data = 0f cf           0.120 uS   .
  146   0008fa    0500  undefined                  M   0.240 uS   .
  147   0008fc    0100  undefined                  M   0.200 uS   .
  148   0008fe    f046  undefined                  M   0.200 uS   .
  149   001500    8c2e  prefetch                   N   0.160 uS   .
  150   001502    fa16  prefetch                   N   0.160 uS   .
  151   001504    2e00  prefetch                   N   0.120 uS   .
  152   001506    2689  prefetch                   N   0.120 uS   .
  153   001500  MOV     PS:00fa,SS                     0.080 uS   .
  154   001508    00ee  prefetch                   N   0.040 uS   .
  155   0010fa    0080  memory write               N   0.120 uS   .
  156   001505  MOV     PS:00ee,SP                     0.080 uS   .
  157   00150a    892e  prefetch                   N   0.040 uS   .
  158   0010ee    0100  memory write               N   0.200 uS   .
  159   00150c    e81e  prefetch                   N   0.120 uS   .
  160   00150e    2e00  prefetch                   N   0.120 uS   .
```

At state 149, the foreground monitor program starts.

Single Step and Foreground Monitors

To use the "step" command to step through processor instructions with the foreground monitor listed in this chapter, you must modify the processor's interrupt vector table. The entry that you **must** modify is the "BRK flag" interrupt vector, located at 4H thru 7H. The "BRK flag" interrupt vector must point to the identifier UEE_BRK_FLAG in the foreground monitor.

Extended Address Mode

To use the foreground monitor in the extended mode, in defalt, you can not use page register 0, page register 64 and other one page register to locate the foreground monitor.

You **must** modify the processor's interrupt vector indicated "FGMON_VECNO" in the foreground monitor source.

You must set common stack area for the nomarl and exteded address mode, because the foreground moniter temporay move into the normal mode.

Limitations of Foreground Monitors

Synchronized measurements

You cannot perform synchronized measurements over the CMB when using a foreground monitor. If you need to make such measurements, set the foreground/background configuration option to **cf mon=bg**.

Notes

C

70136 Emulator Specific Error Messages

The following pages document the error messages which are specific to the 70136 emulator. The cause of the error is described, as well as the action you must take to remedy the situation.

Message 140 : Second term is smaller than first term

Cause

This error occurs when you attempt to enter the address range with the first term is smaller than the second term.

Action

Enter the address range with the first term is not smaller than the second term.

Message 141 : Range terms must be the same type

Cause

This error occurs when you attempt to enter the address range with the address expression of the first term is different from the second term.

Action

Use the same address expression to enter the address range. Refer to "ADDRESS" section of the appendix A of this manual for more information.

Message 142 : Range terms must be in the same segment

Cause

This error occurs when you attempt to enter the address range with :<OFFSET> address expression and the value of first term is different from the value of second term.

Action

Use the same value to enter the address range with :<OFFSET> address expression.

Message

143 : Physical address can not be used

Cause

This error occurs when you attempt to enter the address with physical address expression in the emulation commands which physical address expression cannot be used in.

Action

Use the other address expression which can be used in the emulation commands. Refer the "ADDRESS_EXPRESSION" section of the Appendix A in this manual.

Message 144 : Extended address can not be used

Cause

This error occurs when you attempt to enter the address with extended address expression in the emulation commands which extended address expression cannot be used in.

Action

Use the other address expression which can be used in the emulation commands. Refer the "ADDRESS_EXPRESSION" section of the Appendix A in this manual.

Message 145 : Can not get extended address

Cause

This error occurs when program execution should take place in real-time (refer to **cf rrt** command in Appendix A), the emulator should break to the monitor to read page registers (refer to **cf pgrd** command in Appendix A), and the emulation commands which need physical to extended address conversion are entered in running user program (the "U>" prompt is shown).

Action

Use the extended address expression not to need the physical to extended address conversion in the emulation commands. Refer the "ADDRESS_EXPRESSION" section of the Appendix A in this manual.

If you want to use the "**bp**" command, you should break to the monitor to set software breakpoints correctly. Refer the "Using Software Breakpoints" section of the chapter 2 in this manual.

Message 146 : I/O address range overflow

Cause

This error occurs when you attempt to enter the I/O address which is over I/O address range (0-0FFFF hex) after executing the I/O command.

Action

Use the I/O address which is not over the I/O address range after executing the I/O command.

Message 150 : DMA controller is 71071 mode (70236/70236A Emulator only)

Cause

This error occurs when you attempt to access the DMA37 class registers (refer to "REGISTER NAMES and CLASSES" section in Appendix A) and the 70236/70236A internal DMA Control Unit is uPD71071 mode.

Action

Change the mode of 70236/70236A internal DMA Control Unit to uPD71037 mode.

Message 151 : DMA controller is 71037 mode (70236/70236 Emulator only)

Cause

This error occurs when you attempt to access the DMA71 class registers (refer to "REGISTER NAMES and CLASSES" section in Appendix A) and the 70236/70236A internal DMA Control Unit is uPD71037 mode.

Action

Change the mode of 70236/70236A internal DMA Control Unit to uPD71071 mode.

Message 152 : Device not enable (70236/70236A Emulator only)

Cause

This error occurs when you attempt to access registers in the 70236/70236A internal peripheral (ICU, TCU, SCU, and DMAU) and the internal peripheral is disabled.

Action

Enable the internal peripheral to modify the OPSEL register (on-chip peripheral selection register) with using "reg" command (refer to "REGISTER NAMES and CLASSES" section in Appendix A).

Notes

Index